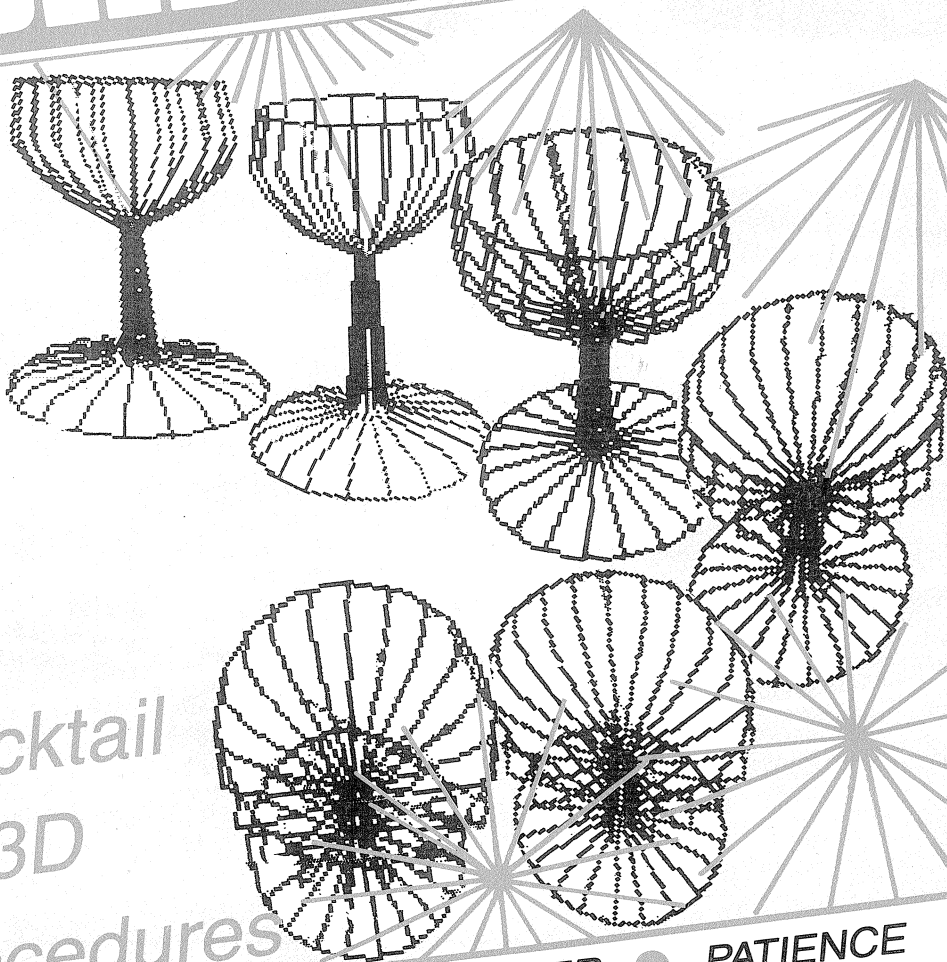


Vol.6 · No.9 · March 1988

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES



Cocktail
of 3D
Procedures

- VIDEO CASSETTE CATALOGUER ● PATIENCE
- PROGRAMMING IN C ● SPRITE ANIMATOR

FEATURES

Instant Listing of Functions and Procedures	9
Now C Here	11
A Cocktail of 3D Procedures	16
Video Cassette Cataloguer	19
Barry Christie Visuals - A Sprite Animator	24
First Course - Character Control	30
The Master Pages - Talking to the ADFS (Part 2)	41
MOS Plus Review	44
Master Hints	46
Circuit Analysis (Part 2)	49
Exploring Assembler (Part 8)	54
Workshop - Using Printers	58
BEEBUG Education	60
Patience	63

REVIEWS

System Gamma	6
The C Programming Language	15
Master Emulation ROM	28
MOS Plus Review	44
Printer ROMs	47

REGULAR ITEMS

Editor's Jottings	4
News	4
Supplement	33-40
Master Hints	46
Hints and Tips	67
Postbag	69
Subscriptions & Back Issues	70
Magazine Disc/Tape	71

HINTS & TIPS

GENERAL

Invisible Calculations
Colour View
Basic Priority on the B+
Power-up Configuration
Double Stepping

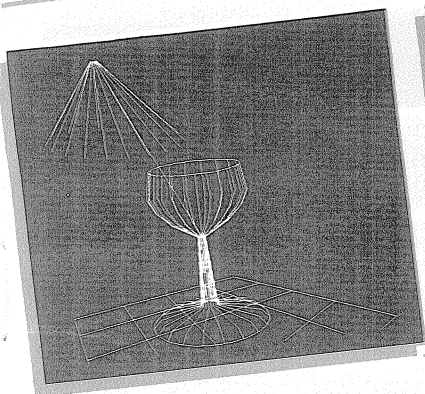
MASTER

Automatic Backup
Simple Auto-Save

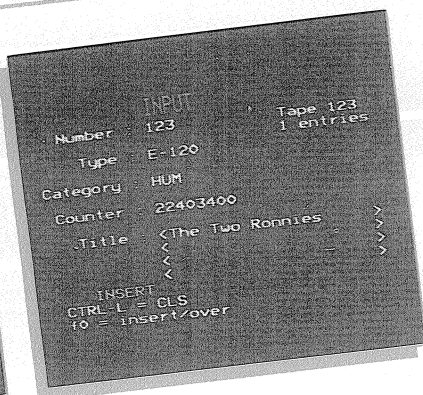
PROGRAM INFORMATION

All programs listed in BEEBUG magazine are produced direct from working programs. They are listed in LISTO1 format with a line length of 40. However, you do not need to enter the space after the line number when typing in programs, as this is only included to aid readability. The line length of 40 will help in checking programs listed on a 40 column screen.

Programs are checked against all standard Acorn systems (model B, B+, Master, Compact and Electron; Basic I and Basic II; ADFS, DFS and Cassette filing systems; and the Tube). We hope that the classification symbols for programs, and also reviews, will clarify matters with regard to compatibility. The complete set of icons is given



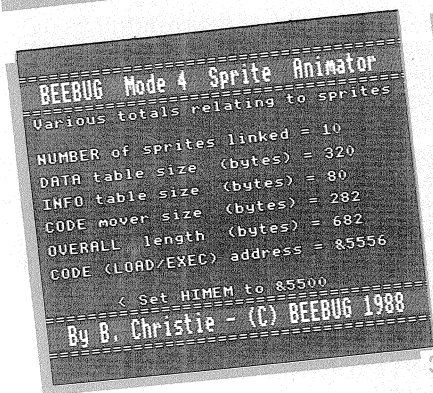
1.



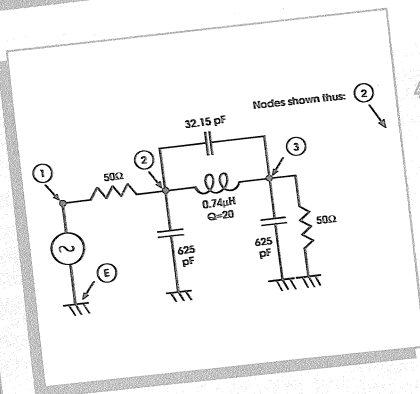
1. Cocktail of 3D Procedures

2. Video Cassette Cataloguer

3. Sprite Animator



3.

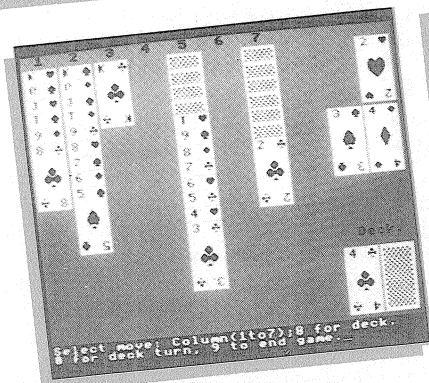


4. Circuit Analysis

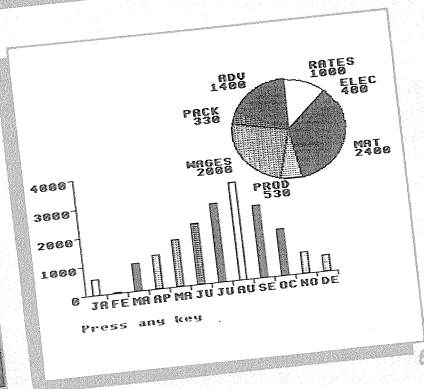
5. Patience

6. System Gamma Reviewed

4.



5.



6.

below. These show clearly the valid combinations of machine (version of Basic) and filing system for each item, and Tube compatibility. A single line through a symbol indicates partial working (normally just a few changes will be needed); a cross shows total incompatibility. Reviews do not distinguish between Basic I and II.

Computer System

Master (Basic IV)



Compact (Basic VI)



Model B (Basic II)



Model B (Basic I)



Electron



Filing System

ADFS



DFS



Cassette



Tube Compatibility

Tube



Editor's Jottings

PROGRAMMING IN C

This issue of BEEBUG sees the start of a series of introductory articles (four to five) which will be devoted to the programming language C. Given the amount of recent publicity with regard to this language, and its availability on the whole of the BBC micro range, including the Archimedes, this is perhaps no more than is reasonable. The object of this series is not to provide a comprehensive text book on C, but to provide a flavour of C for those who are considering this language as one which they might wish to use themselves.

In fact, this obscures a rather more fundamental question as far as BEEBUG is concerned. In all the time that we have been supporting the BBC micro (in its various guises), we have never devoted any part of the magazine to languages other than Basic and Assembler (unless in the form of reviews of compilers and interpreters for languages such as Forth, Prolog, Pascal, Comal, etc).

So the question is quite simple. Should BEEBUG, in the future, be prepared to devote further pages to articles about C and to programs written in C. If it were a matter of simply increasing the number of pages then the answer might be easier to find, but if that were the case then there might well be many other topics which might warrant some or better coverage in BEEBUG (hardware projects, for example). And any decision must be in the best interests of the majority of BEEBUG members.

If you have views on this subject then we would be pleased to hear from you (as we would on any topic related to the magazine). It is not our intention to initiate major changes, more to determine the direction which BEEBUG should take in the future.

LOOKING AHEAD WITH BEEBUG

The next issue of BEEBUG will mark the end of volume six, by which time we shall be looking forward to the first issue of volume seven. That will be a total of sixty issues of BEEBUG all told. As usual, we shall be publishing a complete index to the whole of volume 6, and this will be distributed free to all members with the first issue of volume 7 (the May issue).

..News..News..

MICROLINK FOR FREE

Anyone with a 1200/75 baud modem and telephone connection can try out Microlink's electronic mail service free of charge using Dial-a-Demo. Simply dial 01-583 1275 and when the >PAD prompt appears just key CALL 72 followed by Return. When asked to sign on, key ID MAG111 (that's three letters and three digits) and press Return again. The password DATABASE followed by Return will bring up the menu. There are four sections to browse through (communication, information, services and leisure) each describing a different aspect of the system. For further information telephone (0625) 878888.

ACCOUNTING FOR THE ARCHIMEDES

Minerva, of System Delta and System Gamma fame, has released an entire suite of five *Business Accounting* modules for the Archimedes. The programs are fully integrated with facilities for exporting data to spreadsheet and graphics programs. No price has yet been announced. Minerva are on (0392) 37756.

ARCHIMEDES OFF TO A FLYING START

Mitre Software is a company new to the Acorn world but already well established on PCs and compatibles. Now *Flying Start II*, a database package, is available for the Master 512 and for the Archimedes under the PC emulator, and a native mode version (for the Archimedes) is expected to be released shortly. *Flying Start* is a program which lets the user create records, files, lists, reports and labels for later retrieval and analysis quickly and easily. The package costs £69.95 (inc. VAT) for either system, and can be obtained from Mitre Software Ltd, International House, 26 Creechurch Lane, London EC3A 5BA or telephone 01-283 4646.

NEW GAMES FOR THE BEEB

New Games (or nearly new games) are still being released for the BBC micro. Bug-Byte's *Tutankhamun's Revenge* is described as an arcade adventure game allowing you, in the role of Howard Carter (the English explorer who discovered the tomb), to explore and collect treasure. A screen designer is also provided to

News..News..News..News..News..

add new rooms and pyramids to those already present. Cost of Tutankhamun's Revenge is £7.95 on dual 40/80 track disc. This new game follows Bug-Byte's release earlier this year of arcade adventure Plan B2, a sequel to the best selling Plan B. Plan B2 costs just £2.95 on cassette.

Play It Again Sam 2 from Superior Software is yet another games compilation featuring this time Repton 3, Galaforce, Codename:Droid and Craze Rider. Prices range from £9.95 for a dual cassette, through £11.95 for 5.25" disc to £14.95 for 3.5" disc. Bug-Byte are on 01-439 0666 and Superior are on (0532) 459453, or try your local Acorn dealer (BEEBUG keeps plenty of these in stock).

NO JACK OF ALL TRADES, BUT DABHAND AT SOME

Yet another new product from DABS Press has just been released in the form of *ViewSheet and ViewStore, A Dabhand Guide*. Written by Acorn User Editor Graham Bell, this mammoth 350 page tome offers to explain all you are ever likely to want to know about these two View family products for the BBC micro and Master. The book also contains listings of a number of utility programs designed to make use of ViewStore and ViewSheet even easier. The book costs just £12.95, from shops or direct from DABS Press, 76 Gardener Road, Prestwich, Manchester M25 7HU. Telephone 061-773 2413 for further information.

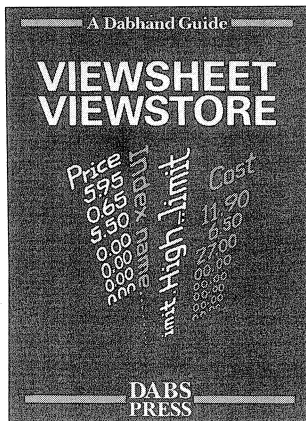
HELP FOR USERS

A new service aimed particularly at computer users, as opposed to programmers, has been established by *Micro Advisory Services* of Richmond, London. The intention is to help BBC micro users to exploit to the full the potential of their machines in serious and practical applications. If you are unsure as to the best

configuration, hardware or software combination, then Micro Advisory Services may be able to help. For more information telephone 01-878 7044 or write to the company at 314 Upper Richmond Road West, London SW14 7JN.

HIGH-RES GRAPHICS DUMP

Silicon Vision, producers of Real Time Graphics and Real Time Solids Modeller, has announced a new high resolution graphics dump for Epson compatible printers, called *Super-Dump*. This achieves a resolution of 1920x1024 (sic) on most printers. Like the high-res dump published in BEEBUG (Vol.6 No.6) this works from spooled files of VDU codes to achieve its aims. Silicon Vision are at 47 Dudley Gardens, Harrow, Middx HA2 0DQ, or telephone 01-422 2274.



POLYTECHNIC VIDEOTEX DATABASE

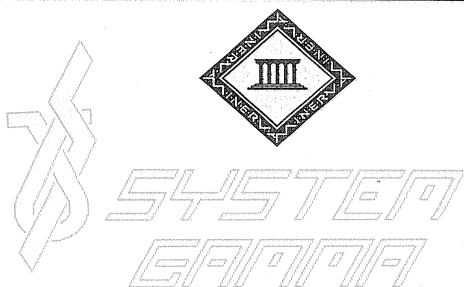
Oxford Polytechnic is operating an educational Videotex service running at 1200/75 baud. The service offers information on staff research, educational telesoftware, curriculum development, primary education, applications for the disabled, plus news and comment on the Polytechnic world. The Videotex service can be accessed on Oxford (0865) 819937. For more information contact the Database Manager, Computer Education Unit,

Oxford Polytechnic, Lady Spencer-Churchill College, Wheatley, Oxford OX9 1HX.

FIND OUT ABOUT BARTG

BARTG's (*British Amateur Radio Teleprinter Group*) latest quarterly journal (Datcom) has just been published. BARTG covers all aspects of amateur data communications (RTTY, Packet, Amtor and FAX) with UK membership costing £8. Members receive copies of Datcom four times a year. For more information contact John and Pat Beedie, GW6MOK and GW6MOJ, Ffynnonlas, Salem, Llandeilo, Dyfed SA19 7NP.

B



Geoff Bains casts an eye at Minerva's new package System Gamma for creating graphic displays from your database.

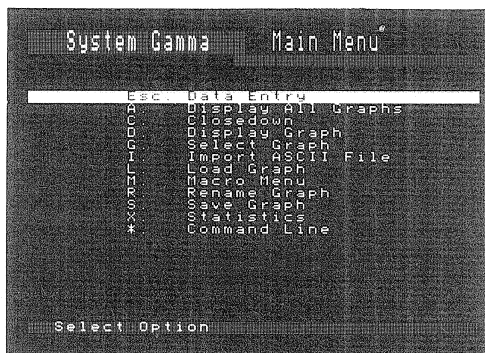
When Minerva launched the System Delta database language/package last year (see BEEBUG Vol.5 No.4) the one thing missing was a good way of displaying the data sorted and searched for. True, the data could be got at quite easily and used with a Basic program to draw a graph or chart. However, that relies on the same programming skills which System Delta was designed to render unnecessary. Now Minerva has followed the tradition of System Delta with a data presentation package called System Gamma.

Product System Gamma
Supplier Minerva Systems
 69 Sidwell Street,
 Exeter,
 Devon EX4 6PH.
 Tel. (0392) 37756
Price £49.95

Again the package is a programming language with a series of star commands to manipulate data and create displays on the screen. As with System Delta, when you purchase System Gamma you are not actually told how to use the programming language. It is not easy to guess it all either, as there are a great many star commands (such as *SGOPT, *SGdefine, *SGxscale, *SGfont, *SGaxis and *SGgraph), not to mention a myriad *FX163,18,n commands, which are near to impossible to fathom without help.

All the information to actually program your own graphics presentation package is contained in the System Gamma Programmer's Reference Guide. Unfortunately, as with System Delta before it, this will set you back a further £19.95.

I can't help feeling this is both a bit of a con and extremely silly. If the guide is written and printed, it costs very little more to include it (in presumably large quantities) with the software. This habit of Minerva's is not one we should look to encourage.



The documentation you do receive with System Gamma is the manual for an application already written. To be fair, this program does

System Gamma * Data Entry Mode Memory: []

Current Graph: SALES86 Current Font: Normal Type: List

Label	X value	Y value	Colour	Highlight
JAN	5.00	560.00	Auto	Off
FEB	7.00	40.00	Auto	Off
MAR	12.00	970.00	Auto	Off
APR	11.00	120.00	Auto	Off
MAY	16.00	160.00	Auto	Off
JUN	20.00	220.00	Auto	Off
JUL	28.00	220.00	Auto	Off
AUG	25.00	340.00	Auto	Off
SEP	17.00	720.00	Auto	Off
OCT	12.00	720.00	Auto	Off
NOV	18.00	750.00	Auto	Off
DEC	18.00	600.00	Auto	Off

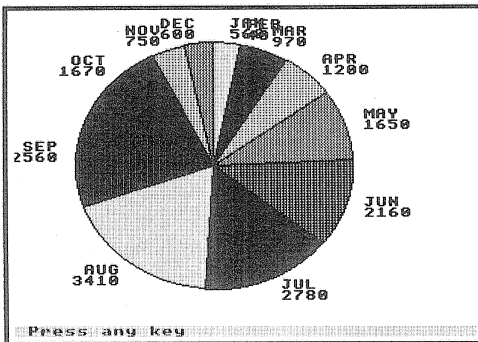
Outline: On Percent: Off Labels: On Values: On

allow you to present displays of data entered by hand or imported from ASCII files, or from System Delta, in just about any way you would want. This program

will suit the majority of people as it is. However, if you want to import data from something a little more exotic, you will have to write your own program or modify the ones given - and we are back to the Programmer's Reference Guide again.

System Gamma is supplied in a 32K EPROM which plugs into a normal 8/16K ROM socket thanks to a small carrier board of the Computer Concepts type. The ROM is compatible with all types of BBC micro but requires shadow RAM for extra memory and the Acornsoft GXR graphics extension ROM for some of its fill patterns. These can both be either built in from the start (as in the Master) or additional items to your model B micro.

selected colour and highlighted or not - the way the highlighting is done depends on the graph type but all are effective.

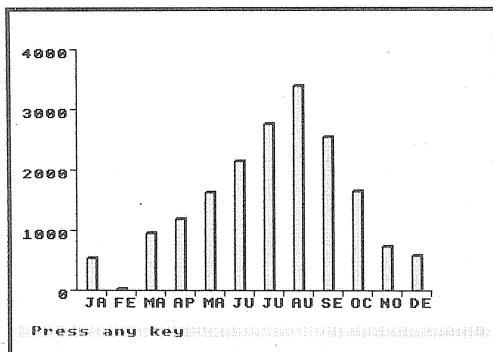


The type of graph is also specified on this table along with the font used for the axis and entry labels (normal, small or thin), and flags are provided to

Unfortunately, although the programming language is happy with, say, an Aries board for shadow RAM and a plug-in GXR ROM, some of the opening sections of the applications program are not - they require a little rejigging to get the program to run. It is not hard to do but rather defeats the purpose of ready-written software. In addition, the different on-screen fonts are not always available with a model B system.

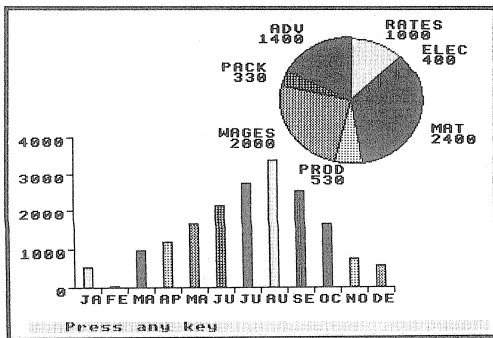
display an outline, percentages, values and labels, and are toggled with Ctrl-letter keypresses. The whole table is easily and

The applications program provides graphics displays of any data in a number of different graph forms. These are simple histogram, '3D' histogram, pie chart, scatter diagram and line graph. The whole program is centered around a rather peculiar menu-and-additional-single-keypress driven module which will be a familiar style to existing Minerva software users.



quickly drawn up for any particular graph required. This data can then be saved to disc or altered with the simple and effective editor.

Data for the graphs is entered into a simple table providing for a label and two values for each entry. Each



entry's section of the resulting graph can now also be defined in a specified or automatically

Data can also be 'imported' into the program from other software (most likely a database or spreadsheet) as long as it is in an ASCII file and takes the form <label>, <x-value>, <y-value>, <label>, <x-value>, <y-value>, etc. Many programs can produce such data, and it is relatively simple to write a short Basic program to convert other software's data to this format or to generate such data to start with. For System Delta data a

special program is provided for stripping the required data from the Delta index cards.

Once the data is entered it can be instantly displayed in the type of graph format chosen, and some further manipulation performed on the screen. The graph can be changed in size (and to some extent shape), and shifted around the screen. It can be printed out or the screen saved to disc for further additions later.

For graphs which plot two values for each entry (scatter and line, not histogram or pie) the line of best fit can be instantly plotted on the graph, and the equation is given on the prompt line at the bottom of the screen. Complete statistical analysis of each graph can also be performed by the program. The minimum, maximum, line of best fit, mean, standard deviation and correlation coefficient are also displayed nearly instantaneously and can be output to a printer if required.

Several graphs can be in memory at a time, and with more than one the ability to shrink graphs and move them around the screen becomes a vital one. More than one graph can be drawn on the same screen, at any size or relative position. Again, the multiple graph screen can be printed or saved to disc.

For that final touch of professional presentation, text can be added to the complete screen in any position inside boxes of various designs and in the same range of fonts as are provided for axis labelling. Lines can be drawn

to divide up the display and graphs boxed and shaded to give a professional look to the whole screen. Again the finished screen can be saved or printed.

Of course, the same set of instructions will often be used to produce similar graphs from different sets of data or to update a graph of cumulative data.

To make this much easier, System Gamma is also provided with a macro facility. All the operations of the system can be defined in a simple program (which basically uses the System Gamma star

command language, without the stars) and the program saved to disc.

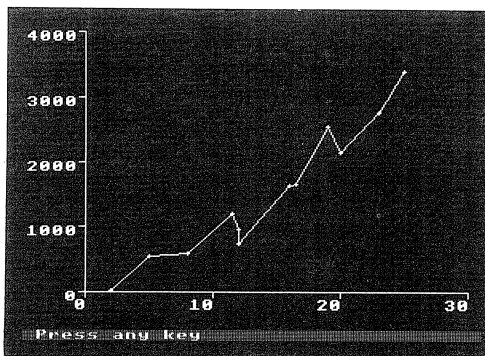
Useful additions such as prompted waits for a keypress are included, so a complete 'slide show' presentation of many graphs can be put together for near-automatic display.

There is nothing that System Gamma does which any reasonably competent Basic programmer with a little elementary geometry couldn't manage on their own. However, this package does it all for you and at a

speed and with a flexibility which are hard to match.

The lack of inclusion of a real programmer's manual is a grave omission, but even without it System Gamma provides a powerful all-purpose data presentation package which will prove well worth the price to anyone struggling with the Beeb for business.

B



INSTANT LISTING OF FUNCTIONS AND PROCEDURES

Graham Crossley has developed a highly useful and practical utility which allows any function or procedure in your current program to be listed simply by referencing its name. Now you can forget all about line numbers.

INTRODUCTION

Several 'professional' computers allow function and procedure definitions within a program to be listed individually simply by typing the name of the required section of code after the LIST command. The utility presented here allows the BBC computer user to enjoy the same flexibility as those more expensive machines when editing and debugging a Basic program.

USING THE UTILITY

Type in the listing, taking particular care with the machine code, and then save it away before running it. If no errors are reported (the assembled code includes a checksum) you will be asked if the code is to be saved (using the name NEWLIST), after which the new list command is installed. At other times the file NEWLIST need only be *RUN to install the utility. In order to list a particular function or procedure in any program in memory simply type:

*LINE <fn/proc name> Return

and the desired section of code will be listed instantly. Function and procedure names may be abbreviated and ended with a full stop in which case the first function or procedure within the program whose name matches the characters entered will be listed. An error message is produced if the name entered cannot be found.

IMPROVEMENT

The best way to use a utility such as this, and the way that I use it, is to have it residing in sideways ROM or RAM along with other useful routines. This makes it permanently available

and immune from Break or Ctrl-Break. Details of how to achieve this can be found in the Advanced User guides. Articles describing the techniques involved have also been published in BEEBUG (see Vol.5 Nos.2-4 & Vol.6 No.1)

HOW IT WORKS

Use has been made of the *LINE operating system command which allows the user to pass a parameter string, in this case the name of a function or procedure, to a resident machine code routine. The call is vectored through the USERV vector at &200/201 the contents of which are changed to the address of the new list command. On entry the X and Y registers point to the address in memory of the parameter string.

A resident Basic program is searched for a DEF token (&DD) and a match check carried out between its name and the name entered at the keyboard. Should the names not match, the program is searched for the next DEF token. Assuming a match is found the line number is stored and the search continues, this time for the line where the function or procedure ends. This will be the occurrence of either:

1. ENDPROC.
2. End of function (=).
3. New DEF statement.
4. End of program.

The start and end line numbers of a desired function or procedure can easily be found, but the problem is how to pass these to the Basic interpreter along with the LIST command. The solution used here is to force the LIST command, together with start/end line numbers and Return, into the keyboard buffer as if they had been actually typed in. But what about the line numbers being in 2 byte format? These need to be converted to the normal decimal numbers that would normally be typed in at the keyboard. This is achieved by counting the number of 10000s, 1000s, 100s, 10s and 1s in the original two-byte number, converting the counts to ASCII codes (count OR &30) and placing them in the keyboard buffer. The conversion uses the subtraction method of counting and is actually used within the Basic ROM.

```

10 REM Program FN/PROC Lister
20 REM Version B1.2
30 REM Author Graham Crossley
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 MODE 7:ON ERROR GOTO 170
110 PROCinitialise
120 PROCassemble
130 PROCcheckcode
140 PROCsavecode
150 END
160 :
170 ON ERROR OFF:MODE 7
180 REPORT:PRINT" at line ";ERL
190 END
200 :
1000 DEF PROCinitialise
1010 pointer=&70:length=&72:source=&73
1020 dest=&75:line=&77:result=&79
1030 temp=&7A:command=&7C:osbyte=&FFF4
1040 ENDPROC
1050 :
1060 DEF PROCassemble
1070 FOR pass=0 TO 2 STEP 2
1080 P%=&900
1090 [:OPT pass
1100 .setup
1110 SEI
1120 LDA #list MOD256:STA &200
1130 LDA #list DIV256:STA &201
1140 CLI
1150 .exit
1160 RTS
1170 :
1180 .list
1190 STX command:STY command+1
1200 .checkforname
1210 LDY #&00:LDA #&0D
1220 CMP (command),Y:BEQ exit
1230 LDA #&83:JSR osbyte
1240 STX pointer:STY pointer+1
1250 LDY #&01:LDA (pointer),Y
1260 CMP #&FF:BEQ exit
1270 :
1280 .begin
1290 CLD
1300 JSR checkfortoken
1310 BEQ deffound
1320 .nextline
1330 JSR adjustpointer
1340 BNE begin
1350 BRK
1360 EQUB 29
1370 EQU$ "No such FN/PROC..."
1380 BRK
1390 :
1400 .deffound
1410 INY:LDA (pointer),Y
1420 CMP #ASC" ":BEQ deffound
1430 .checknames

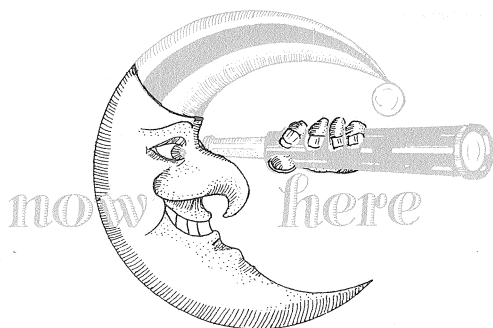
```

```

1440 INY:LDX #&00
1450 .nameloop
1460 LDA (pointer),Y
1470 CMP #&0D:BEQ nextline
1480 CMP #ASC" ":BEQ nextline
1490 CMP #ASC"(":BEQ nextline
1500 STY temp
1510 PHA:TXA:TAY:PLA
1520 CMP (command),Y:BNE nextline
1530 INY:LDA (command),Y
1540 CMP #&0D:BEQ chknamelen
1550 CMP #ASC" ":BEQ matchok
1560 INX:LDY temp:INY:BNE nameloop
1570 :
1580 .chknamelen
1590 LDY temp:INY:LDA (pointer),Y
1600 CMP #ASC" ":BEQ matchok
1610 CMP #ASC"(":BEQ matchok
1620 CMP #ASC"(":BEQ matchok
1630 CMP #&0D:BNE nextline
1640 :
1650 .matchok
1660 LDY #&02:LDA (pointer),Y
1670 STA source:STA dest
1680 DEY:LDA (pointer),Y
1690 STA source+1:STA dest+1
1700 :
1710 .listto
1720 JSR adjustpointer
1730 BEQ convertsource
1740 JSR checkfortoken
1750 BEQ convertsource
1760 STY temp:LDY #&02
1770 LDA (pointer),Y:STA dest
1780 DEY:LDA (pointer),Y
1790 STA dest+1:LDY temp
1800 LDA (pointer),Y
1810 CMP #&E1:BEQ convertsource
1820 CMP #ASC"=:BNE listto
1830 :
1840 .convertsource
1850 LDA #&15:LDX #&00:JSR osbyte
1860 LDA #&8A:LDY #ASC"L":JSR osbyte
1870 LDY #ASC" ":JSR osbyte
1880 LDA source:STA line
1890 LDA source+1:STA line+1
1900 JSR convert
1910 LDA #&8A:LDX #&00
1920 LDY #ASC" ":JSR osbyte
1930 :
1940 .convertdest
1950 LDA dest:STA line
1960 LDA dest+1:STA line+1:JSR convert
1970 LDA #&8A:LDX #&00
1980 LDY #&0D:JMP osbyte
1990 :
2000 .convert
2010 LDX #&04
2020 .con1
2030 LDA #&00:STA result:SEC
2040 .con2

```

Continued on page 62



If programming in the language C has aroused your interest then our new series by Dec McSweeney starting this month could be just what you are looking for.

The language C has been around for some time now, available for "real computers" like the IBM, DEC and NCR ranges. Not having any of these beasts to hand, I was delighted to see several versions of the language become available for the humble Beeb. This article is the first in a series aimed at the experienced BBC Basic programmer who wants to get to grips with C. In the next few issues we will develop a number of programs which will illustrate the crucial differences between C and BBC Basic.

For more details, the reader is referred to "The C programming language" by Kernighan and Ritchie (reviewed elsewhere in this issue). This book is the standard work on the subject and was used in the preparation of this series. The review of Acornsoft C (BEEBUG Vol.6 No.5) contained background information about compiled languages, and C in particular, which I will not reproduce here.

Most of the information in this series applies to any implementation of C on any machine, though it is the BEEBUG version I have used to test out each program. To save space, examples of C statements are not usually presented in the context of a complete program, but if you have access to a C compiler try inserting the examples in this skeleton (note that C does not use line numbers):

```

/* skeleton program for C examples */
/*      explanations later      */
#include <h.stdio>
main(){
/*      Insert main routines here      */

}

/*      Other functions follow      */

```

A full explanation of the above follows the final example in this article. If you have the BEEBUG version of C, you will need to use a word processor to enter the programs. Some implementations include a built-in editor.

THE BASIC ELEMENTS

Let's look first at some general points regarding C syntax. A program in C consists of a set of functions; the top-level or harness function must be called **main**. The arguments or parameters of a function are enclosed in *normal* brackets (). Most C statements end with a semi-colon - exceptions you'll meet this month are the control flow statements **if** and **while** - and may be grouped together in braces {}. Comments reside between */** and **/*. Reserved words, that is words which are part of the C language, are written by convention in lower case.

VARIABLES

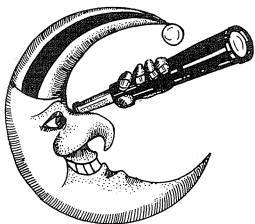
In C, as in Basic, data may be held in variables. The first major difference between C and Basic is that C requires the declaration of all variables before they are used. Rather than assuming the data type from the variable's name (e.g. **integer%**, **string\$**), the type must be declared, for example:

```

int number; /* integer - 16 bits */
char alpha; /* single byte (1 char)*/
float realnumber; /* 32 bits */

```

In the above, we have declared an integer called **number**, which can hold any value between -32768 and +32767; a single byte called **alpha**, which can hold any ASCII character; and a larger variable called **realnumber** with a capacity similar to Basic's real numbers. Much larger integers may be stored in type **long** (32 bits). This is not the whole story, but it's quite enough for now.



PROGRAM CONTROL

A program in C comprises a set of functions. All C programs must contain a function called `main()`; this may be the only

function in the program or it may call other functions, which call other functions in turn (or themselves, recursively), and so on. The language has no in-built facilities for input and output, but standard library functions are provided for such hardware-dependent activities (for example: `scanf()`, which is equivalent to INPUT, and `printf()`, the analogue of PRINT). The beginner may like to think of these as part of the language, as they are available universally.

As in Basic, control passes from one statement to the next in sequence, starting with the first statement in `main`. Naturally, there are a number of ways to disrupt this orderly flow, including `if` and `for`. There is also a `goto` statement, but don't let me catch you using it in such a structured language as C!

The `if` statement looks much like its Basic counterpart;

```
if (condition)
    statement-1;
else
    statement-2;
statement-3;
```

The condition in brackets is evaluated. If it is TRUE, `statement-1` is executed; if it is FALSE, `statement-2` is executed. In either case control then passes to `statement-3`. Several statements may be grouped together using braces, and multiple `if` statements may be nested. A simple example of an `if` might be:

```
byte = getchar();
if(byte >= 32)
    printf("printable");
else
    printf("control character");
```

`getchar()` is a library function returning a single character from the input stream, just as `GET$` does in Basic. The function `printf()` is like the

Basic PRINT statement, but rather too complicated to describe fully in one go. Bear with me.

FOR LOOPS

The `for` loop is also similar to the equivalent Basic construction. An example such as:

```
10 FOR N% = 29 TO 40 STEP 2
20 PRINT"Aargh!";
30 NEXT
```

could appear in C as:

```
int n;
for(n = 29; n <= 40; n = n+2)
    printf("Aargh!");
```

The bracketed expressions correspond to the start-value, end-value and step portions of the Basic FOR statement, but the loop is executed while the central expression (`n <= 40`) is true, and this need not relate to the variables in the start or end expressions. All three parts are optional. Note the absence of a NEXT statement. The loop consists of the statement following the `for`. However, several statements may be grouped in braces thus:

```
int n,n2;
for(n = 29 ; n <= 40 ; n = n+2) {
    n2 = n*n;
    printf("%d - Aargh!", n2);
}
```

The `printf` function is used in a more complex form here. The string (in quotes) contains formatting information about the variable list which follows; in this case `%d` specifies a decimal digit. Key in the above (inserting it into the main part of the skeleton program, say), and then try replacing the `%d` with `%o` (octal - base 8), `%h` (hexadecimal) or `%f` (floating point). You'd better change the declaration (`int n2`; to `float n2`;) for the last one.

WHILE LOOPS

Users of BBC Basic will be familiar with the REPEAT-UNTIL construction. The nearest equivalent in C is the `do-while` statement. An example:

```
10 N%=0
20 REPEAT
30 PRINT N%
40 N%=N%+1
50 UNTIL N%>10
```

which becomes in C:

```
int n = 0;
do{
    printf("%d", n);
    n += 1;
}
while(n <= 10);
```

As you can see in this example, it is possible to assign values to variables when they are declared. The last statement, `n += 1`, is equivalent to `n = n+1`. This shorthand is available for all the arithmetic operators, '+', '-', '*', and '/', as well as others we have yet to meet. Again, note the use of braces to enclose the body of the loop. Note, too, the semicolon at the end of the while statement, which distinguishes the **do-while** from the true **while** described below. As with the Basic original, the loop is always executed at least once. Replace line 10 in the Basic program with:

```
10 N%=12
```

and the first line of the C example with:

```
int n = 12;
```

Despite the fact that `n` (or `N%`) starts off exceeding the terminating value, the loop is still executed once. If you want to avoid that possibility, then C offers an alternative form of the **while** as in the variation below. Examine this carefully and you'll see that the loop in the C program is never executed, because the test fails before the first pass through the loop.

```
/* This loop is never executed */
int n = 12;
while(n <= 10){
    printf("%d",n);
    n += 1;
}
```

Note the use of braces again to enclose the statements within the loop. **do-while** is exactly equivalent to **REPEAT-UNTIL**, but is less useful, and less frequently used than **while**.

Incrementing and decrementing counters is such a common activity in C that two special unary operators, `++` and `--`, have been provided. These can

be used either before the variable (prefix) or after it (postfix), thus:

`while(n++ <= 10)` means 'test the value of `n`, then add 1 to `n`'.

`while(++n <= 10)` means 'add 1 to `n`, then test'.

Try using these constructions in the examples above. Here is another example, this time in the form of a complete program:

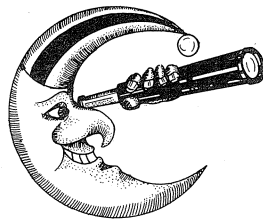
```
#include <h.stdio> /* compiler directive
                    - see below */

main()
{
    int n = 10;
    while(--n > 0)
    {
        printf("d...", n);
    }
    printf("LIFTOFF!\n");
}
```

If you substitute `n--` for `--n` you'll once again see the difference between postfixing and prefixing the `++` operator.

FUNCTIONS

Functions in C are very like their equivalent in BBC Basic. Parameters, or arguments, are passed in brackets. The function itself uses *local* copies of the parameters ("call by value"), and returns a single value. The syntax might look confusing to a Basic programmer. A function like `printf()` may have many arguments, but returns a meaningless value, while `getchar()` has no arguments though the brackets are still needed, returning a single character. A function is assumed to return an integer unless declared otherwise.





Variables declared external to a function may be changed from within a function by passing a pointer to the variable rather than the variable itself -

similar to the idea of passing an address to a Basic function. Pointers are an important feature of C, and we'll cover them in detail later.

To close this introduction and illustrate the points covered, we will now write a program to calculate and print the square and square root of a number entered by the user.

```
/* c.sqrt calculate the square and */
/*      square root of a number */

#include <h.stdio>
main()
{
    double n;
    double sqrt();
    n=0;
    printf("give me a number; ");
    scanf("%f", &n);
    printf("You entered %3.0f", n);
    printf(" square = %5.0f", n*n);
    printf(" root = %2.4f\n", sqrt(n));
}

double sqrt(num)
double num;
{
    double wrk,ans;
    wrk=ans=num/2.0;
    while(wrk > 0.001)
    {
        ans=(ans+(num/ans))/2.0;
        wrk=((num/ans)-ans);
        if(wrk<0.0)
        {
            wrk *= -1.0;
        }
    }
    return(ans);
}
```

The first two lines are comments; the next is a compiler directive, which tells the compiler to copy in a file called h.stdio (supplied with

Beebug C). This contains the input-output routines we will be using. Then comes our main routine, in which we declare our variable n and our function sqrt() as type double (for double precision), print a message, accept a number and call the sqrt function. Note the declaration of the function sqrt as type double. This is necessary because a function is assumed to return an integer unless otherwise specified.

The function scanf() returns the number entered by the user. This function resembles INPUT in the same way that printf() resembles PRINT, and the arguments are analogous - first a control string, then a list of variables. The argument specifying the destination, &n, is actually a pointer to the receiving variable. This is always so with scanf() - and a common source of program errors!

Three calls to printf() follow, showing how an expression like n*n or a function returning a value may replace a variable of the same type (this is generally true throughout the language). The \n in the control string causes a newline character to be printed.

After the brace terminating main(), the sqrt function is defined. This has an argument, locally known as num, which must be specified outside the function braces {}. Within the braces we define our local variables. The actual computation is an iteration, and the limit of accuracy is plus or minus 0.001. Think of the while loop as a REPEAT-UNTIL construction, and the logic is easier to follow. The return() statement at the bottom of the function specifies the value passed back to main just like the terminating = of a BBC Basic function.

As it stands, the program will run once and stop. See if you can tweak it to loop until the user enters 0, or to deliver the square roots of a range of numbers (hint: use a for loop, or while(n++<100)).

Swots among you might like to validate the input, which must of course be positive. C you next month!

B

THE C PROGRAMMING LANGUAGE

This month we commence a short series on programming in the language 'C'. Anyone who wishes to pursue C more seriously will need an adequate reference, and the book most often recommended is 'The C Programming Language' by B.W.Kernighan and D.M.Ritchie. Ray Hughes gives a run down of what you can expect to find in this acclaimed book.

THE C PROGRAMMING LANGUAGE

by B.W.Kernighan & D.M.Ritchie
published by Prentice Hall
at £23.95 (£22.75 to BEEBUG members).

This is an excellent introduction to the world of C programming, containing 228 pages of highly readable information laid out in several distinct sections.

A TUTORIAL

This is included to enable programmers new to C to pick up the language as quickly as possible. However, as the book says, it is not a beginner's guide to programming, and its readers are expected to understand already such concepts as variables, assignments, loops and functions etc. It gets into coding straight away with a good old "Hello world" type program, which is probably the first piece of code written in any language by first time users. Throughout the book, there are also question and answer sessions to allow the reader to fully check that he/she understands the current chapter before moving on to the next. The tutorial in its 30 odd pages manages to cover the main core of the language.

LANGUAGE & PROGRAM STRUCTURE

This section of the book deals with the more complex ideas that need to be covered for a full understanding of the facilities that can be made available to the C programmer. This is in fact the main body of the book and covers in order the following topics :

Types, Operations and Expressions
(Char, int, float, short, long, double etc)

Control Flow
(While, wend, for, next, do-while, case, etc)

Functions and Program Structure
(main() recursion, macros etc)

Pointers and Arrays
(pointers, multi-dimensional arrays etc)

Structures
(lookup tables, fields, Unions etc)

Input/Output
(getchar, putchar, scanf, file access etc)

This then ends the main body of the book. However, the last 85 pages cover other aspects of the language that serious users will want to know about. There is a special chapter on the UNIX system interface, but this may not be of much use to Acorn machine owners, except those lucky Archie owners who may one day get access to UNIX via the proposed ARX operating system.

Next there is the C reference manual which is a full and concise description of all keywords etc, with brief examples of their use. Finally, we have an excellent index covering all items mentioned in the preceding pages.

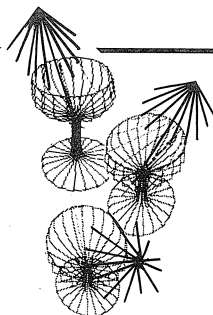
Throughout the book the authors describe and recommend standard programming techniques, and extensive use is made of examples to demonstrate, in particular, structured programming methods. Many short, and some not so short programs are listed in the book, many of which I am sure budding programmers will take and later modify to their own requirements.

IN CONCLUSION

Although reviewed many times since it was first published in 1978, this may be the first review of K&R read by many BEEBUG readers, since C has only recently been released for the Acorn machines. Now nearly a decade old, this book has remained throughout that time the most important reference work for C programmers. It has no equal in what it attempts to achieve, or the deceptive ease with which this is accomplished.

Both the Acorn and BEEBUG implementations of C refer the reader to Kernighan & Ritchie for a detailed guide to the C language. It is certainly not a cheap book (although BEEBUG members are entitled to the usual discount), and compared to some language manuals seems quite thin at 228 pages. However, the information contained within those pages is excellent, and the book really should be considered an absolute must by any prospective C programmer who wants to exploit the language fully. **B**

Cocktail of 3D Procedures



If you have ever wanted to produce 3D wire frame graphics this program is for you. David Lowndes Williams offers procedures to create the 3D equivalents of Basic's DRAW and MOVE instructions.

Two procedures PROCmove(x,y,z) and PROCdraw(x,y,z) form the basis of this article, and provide the 3D equivalents of the MOVE x,y and DRAW x,y commands in Basic. In order to explain how they work the chosen 3D coordinate system needs explaining. Figure 1 shows that the x and y-axes are exactly as they are normally, i.e. in the plane of the screen. The z-axis is perpendicular to the screen, i.e. it appears to go into and out of the screen. Representing 3D points correctly on the 2D screen involves some extensive arithmetic in order to convert each point (x,y,z) to a point (x,y). The procedures take away this drudgery by performing all the calculations necessary.

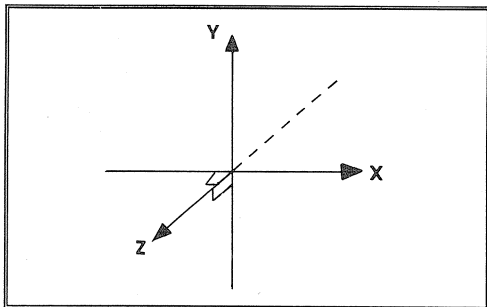


Figure 1

I suggest at this stage, that you type in and run the demonstration program. This draws a 3D wire-frame projection of a wine glass sitting on a grid, with an umbrella in it. The program can be made to view the wine glass from any angle, and accounts for perspective as well. Lines 1000-1280 contain all the routines that you need

to use PROCmove and PROCdraw. Removing line 1150 will remove perspective, giving a true isometric representation of the object.

EXPLAINING THE DEMONSTRATION PROGRAM

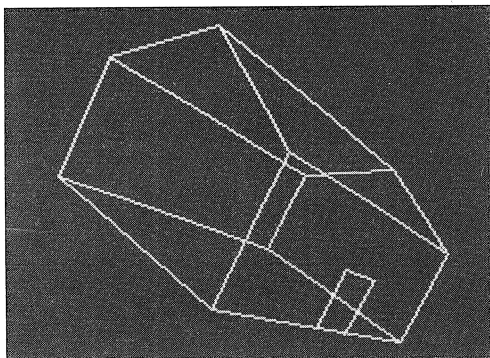
Before PROCmove and PROCdraw can be used to display the object, PROCinitrotation(x,y,z) must be called (see line 110), the parameters representing the orientation of the object. The easiest way to understand this is to run the demonstration program, and try changing the values. For example, start with PROCinitrotation(0,0,90) and you will see the glass standing upright on its base grid. The reason why the object needs to be rotated by 90 degrees about the z axis to bring it upright, is because this makes it easier to determine the coordinates in the position (0,0,0). Now try PROCinitrotation(0,45,90) which means "display the object rotated 45 degrees about the y-axis". You will now see the glass standing on a rotated grid. If you want to look down on the glass you should try rotating about the x-axis by 90 degrees ie 90,0,90. Try it; the base will be closest to you. Using -90,0,90 puts the umbrella closest to you. Negative values rotate the object in the opposite direction. These last two sets of values give a superb demonstration of perspective. You should also try them with PROCperspective in line 1150 removed.

OPERATION OF THE PROCEDURES

PROCmove(x,y,z) and PROCdraw(x,y,z) work in a similar fashion to the standard MOVE and DRAW commands, but in '3 dimensions'. The apparent position of a 3D point on the screen depends on the rotation set up in PROCinitrotation(x,y,z), and the perspective setting. PROCrotate, which is called by PROCmove and PROCdraw, calls PROCperspective(S) at line 1150 where S is the size of the image on the screen (try values from 300-800 for most objects). In the example, S is set to 700 at line 110.

I suggest that when drawing your own 3D designs you initially remove the call to PROCperspective, and only insert it when your design is completed. Your drawing will still look presentable, even if it is not truly in

perspective. You will probably find it helpful to sketch your object on paper first - isometric graph paper is a great help here.



When designing your own programs using these procedures, note that the following variables are used by the procedures:

k m n o p q r s t u v w x y z

and your programs should not alter them. It is particularly easy to forget and use the variables x, y or z in your program.

To summarise, just set S to establish the size of your object, call PROCinitrotation to set its starting position and from then on use PROCmove, and PROCdraw to plot it out. Delete PROCperspective at line 1150 if perspective is not required, and steer clear of variables k-z. All the other procedures look after themselves.

TECHNICAL NOTES - THE PROCEDURES

The program uses the matrix method to perform the rotation. Assuming we are rotating the vector

$\begin{pmatrix} x \\ y \end{pmatrix}$ by an angle of A degrees, then:

$$\begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} \cos A & -\sin A \\ \sin A & \cos A \end{pmatrix} = \begin{pmatrix} \text{new } x \\ \text{new } y \end{pmatrix}$$

$$\begin{aligned} \text{i.e. new } x &= x \cdot \cos A + y \cdot \sin A \\ \text{new } y &= x \cdot \sin A + y \cdot \cos A \end{aligned}$$

It is unnecessary to calculate the sines and cosines of A repeatedly: we get PROCinitrotate to do this at the beginning, and store the results

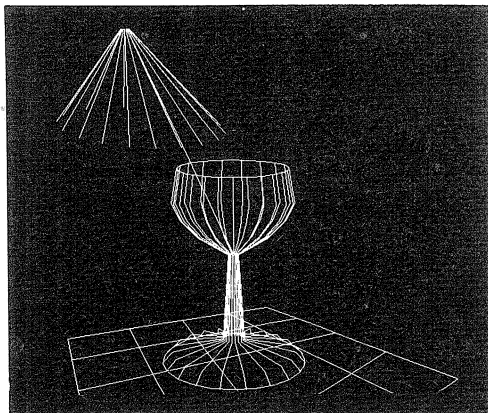
in variables k to s. To perform a rotation, it is then a simple matter of adding the results of two multiplications. This is done by PROCrotate.

Perspective is catered for by remembering that the z axis is perpendicular to the plane of the screen. Therefore dividing each x and y coordinate by z, and applying a scale factor S, the image will be foreshortened appropriately depending on whether z is large or small. To overcome the possibility of z being zero, and hence a division by zero error occurring, z has 1000 added to it before the scaling takes place.

Having called PROCperspective, 640 is added to the x, and 512 to the y co-ordinates to put the model at the centre of the screen.

TECHNICAL NOTES - THE MAIN PROGRAM

The main program to draw the cocktail glass, base and umbrella, lies between lines 120 and 320. It uses PROCmove and PROCdraw, after issuing a PROCinitrotation. The loop from line 180 to 280 calculates points along the vertical lines of the wine glass. Other loops draw a circle at the top and bottom of the wine glass, and an umbrella.



PROGRAMMING YOUR OWN 3D DISPLAY

If you wish to provide the code for your own object, lines 110 to 330 will need to be replaced. A suitable example to try yourself is that of the house shown in figure 2, listing 1 is the resulting code. You will need a reasonably good 3D eye for this. but I have kept it simple

by placing the base of the house on the x-z plane, allowing many of the y coordinates to be zero. Having decided on the coordinates of each of the points, all that remains is to draw them in the right order. There are many ways to do this, the one shown in listing 1 is almost certainly not the one taking the least moves. Start by loading the wine glass program and delete lines 110 to 340 inclusive. Now add the lines given in listing 1. The missing line numbers are for the insertion of the rotation routine discussed later.

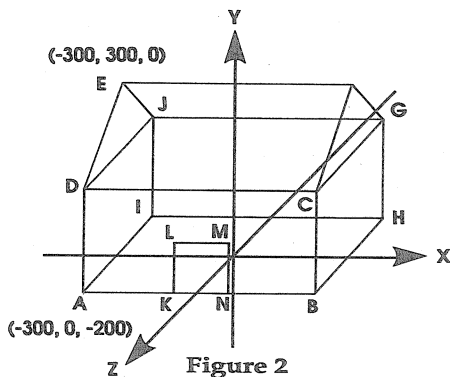


Figure 2

Listing 1

```

110 S=1000
120 xx=300:yy=200:zz=200
160 PROCinitrotation(0,0,0)
170 PROCmove(-xx,yy,-zz):REM D
180 PROCdraw(-xx,0,-zz):REM A
190 PROCdraw(xx,0,-zz):REM B
200 PROCdraw(xx,yy,-zz):REM C
210 PROCdraw(-xx,yy,-zz):REM D
220 PROCdraw(-xx,yy+yy/2,0):REM E
230 PROCdraw(xx,yy+yy/2,0):REM F
240 PROCdraw(xx,yy,-zz):REM C
250 PROCmove(xx,yy+yy/2,0):REM F
260 PROCdraw(xx,yy,zz):REM G
270 PROCdraw(xx,0,zz):REM H
280 PROCdraw(xx,0,-zz):REM B
290 PROCdraw(xx,0,+zz):REM H
300 PROCdraw(-xx,0,+zz):REM I
310 PROCdraw(-xx,0,-zz):REM A
320 PROCdraw(-xx,0,+zz):REM I
330 PROCdraw(-xx,yy,+zz):REM J
340 PROCdraw(-xx,yy+yy/2,0):REM E
350 PROCdraw(-xx,yy,+zz):REM J
360 PROCdraw(xx,yy,zz):REM G
370 REM Door
380 PROCmove(-xx/6,0,-zz):REM K
390 PROCdraw(-xx/6,yy/2,-zz):REM L
400 PROCdraw(+xx/6,yy/2,-zz):REM M
410 PROCdraw(+xx/6,0,-zz):REM N
440 END

```

Remember that you must not use the variables k and m - z. Instead, the program uses xx, yy and zz, setting their sizes in line 90. The program also starts with PROCrotation(0,0,0) to keep the image exactly as shown in figure 2. You may like to put the house-drawing routine in a loop with say sx, sy and sz changing by 5 degrees each time:

```

130 FOR sx=0 TO 360 STEP 5
140 sz=sx:sy=sx
150 CLS
160 PROCinitrotation(sx,sy,sz)
.
.
420 Q=INKEY(50)
430 NEXT

```

I hope that this article provides the basis for your own 3D displays. Having experimented you may like to provide procedures that translate (shift the position of the whole image), and perhaps shear (scale the coordinates relative to their position).

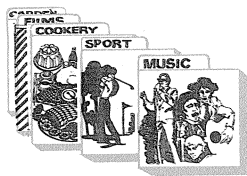
Demonstration Program

```

10 REM Program WineGlass
20 REM Version 2
30 REM Author David Lowndes Williams
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 MODEL:ON ERROR GOTO 2000
110 S=700:PROCinitrotation(0,0,90)
120 REM GRID BASE FOR GLASS
130 GCOLOR,1
140 FOR n%= -400 TO 400 STEP 100:PROCmove(-400,n%, -400):PROCdraw(-400,n%,400):NEXT
150 FOR n%= -400 TO 400 STEP 100:PROCmove(-400, -400,n%):PROCdraw(-400,400,n%):NEXT
160 REM WINE GLASS
170 GCOLOR,3
180 FOR I=0 TO 2*PI STEP 0.3
190 sin=SIN(I):cos=COS(I)
200 PROCmove(-400,200*cos,200*sin)
210 PROCdraw(-375,175*cos,175*sin)
220 PROCdraw(-350,30*cos,30*sin)
230 PROCdraw(-100,15*cos,15*sin)
240 PROCdraw(-50,100*cos,100*sin)
250 PROCdraw(0,150*cos,150*sin)
260 PROCdraw(50,175*cos,175*sin)
270 PROCdraw(150,160*cos,160*sin)
280 NEXT
290 PROCmove(-400,200,0):FOR I=0 TO 2*PI+0.4 STEP 0.4:PROCdraw(-400,200*COS(I),200*SIN(I)):NEXT

```

Continued on page 62



VIDEO CATALOGUER

Tired of keeping track of the contents of all your video cassettes? Let the computer take the strain with Keith Sumner's cataloguing program.

This is a novel database application program which allows the user to keep track of what is on each video tape, where on the tape each item is, and how much time remains. The program is menu driven, allowing the standard database manipulations of loading, data entry, editing, searching, browsing, sorting, deleting and saving, and will also print a label for any selected cassette.

The program is being published in two parts. In this issue we will provide the essential facilities such as the loading, saving, and editing of video cassette information. Next month we will continue with the routines which will allow records to be sorted and a hard copy and cassette labels to be printed.

The program should first be typed in and saved to disc or tape. When run, the program presents the user with a series of options (eleven in total) together with some program status information. Each of the possible options is discussed below. The first time you run the program you will have no data file to load, so the only options available to you will be 'data entry' and 'exit'. If you have the magazine tape or disc, you can load an example file called VIDDAT using the menu option 0 (load data).

OPTION 0 - DATA LOADING

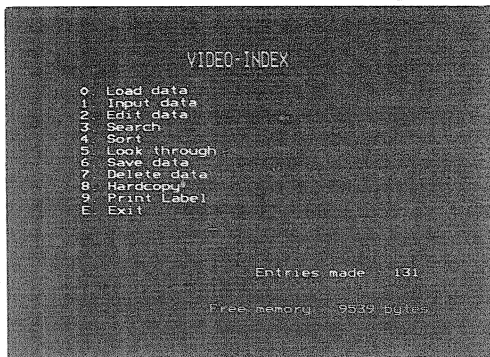
Enter the name of a previously created file to be loaded. If data is already present in memory a warning is given before the loading operation is allowed to continue. If the program is to be used on a single density DFS system it is recommended that each data file be kept on a

separate disc to prevent 'Can't extend' errors occurring.

OPTION 1 - DATA ENTRY

The program requires the following information for each title or record entry.

A. The tape identification number: a three digit number (between 000 and 999) which uniquely identifies the tape to the user and the database program. Pressing Return brings back the main menu.



B. The tape type: a three digit number relating to the tape length (120, 180 or 240 minutes).

C. The tape counter 'begin' and 'end' numbers: each a four digit number corresponding to the video programme start and end points. This number has a twofold purpose: it allows the user to locate a recording within a tape, and allows the program to calculate the duration of the recording and the time remaining on the tape (see notes on video recorder calibration).

D. The video programme category: a three character abbreviation such as FLM for film, SPT for sport, HUM for humour, etc. It is up to you to devise your own mnemonics.

E. The video programme title: up to eighty characters. It is advisable to keep titles short if many titles are to be stored.

OPTION 2 - EDITING

This prompts for the number of the tape to be edited. If the user presses Return at this stage, control is returned to the menu. Editing allows

the user to modify each of the information fields relating to a particular entry on a tape. Typing Return simply leaves the particular field unchanged.

OPTION 3 - SEARCHING

(to be published next month)

OPTION 4 - SORTING

(to be published next month)

OPTION 5 - LOOK THROUGH

Allows the user to browse through the contents of the database. Use the up/down cursor keys to move forward/backwards through the datafile. Function key f1 allows the user to jump to different parts of the data file after specifying a three digit record number as the target. Function key f0 returns the user to the main menu.

OPTION 6 - SAVING DATA

By specifying a filename the user can either create a new file or update an existing one.

OPTION 7- DELETING DATA

This prompts for the number of a tape on which the delete procedure will operate (pressing Return again returns the user to the main menu). The routine then allows deletion of selected titles on a tape, or the complete removal of the tape number from the data file. Suitable safety checks to prevent accidental erasure of entries are included.

OPTION 8 - HARD COPY

(to be published next month)

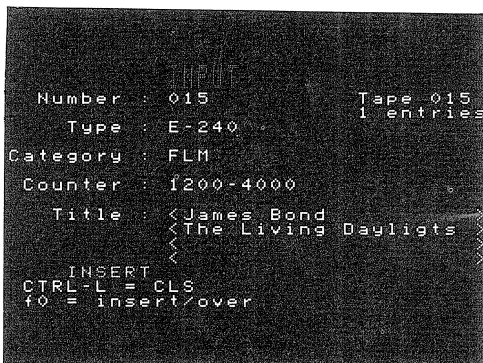
OPTION 9 - PRINT LABEL

(to be published next month)

CALIBRATION OF VIDEO RECORDER

The program already caters for the three tape lengths, E-120, E-180, E-240, but you may wish to calibrate it for improved accuracy. The calibration process involves recording the tape counter number at 5 minute intervals throughout the length of each of the tape types used (this does take ages!). This information is then stored in lines 2850 to 2920. The list of data for each tape type must be terminated by 9999 as an end-of-data marker.

The calibration information for each of the three tape types enables the time remaining on each tape and the length of each programme entry to be calculated.



```
Number : 015           Tape 015
                        1 entries
Type : E-240
Category : FLM
Counter : 1200-4000
Title : <James Bond
      >>The Living Daylights<>>
      >>
      INSERT
CTRL-L = CLS
f0 = insert/over
```

PROGRAM NOTES

There are some important program variables (set in PROCvars at line 2520) which need to be mentioned. M% defines the maximum number of records that are allowed. It is currently and arbitrarily set at 200. Z% holds the number of records associated with the most recently updated data file. MAXENT defines the maximum number of entries that can be made on a tape. This is arbitrarily set at 8. M120, M180 and M240 are the maximum values of the tape count for each of the three types of tape. This data is obtained from the calibration process.

It should be noted that the data entry routine is such that an exact number of characters must be entered (with the exception of the film title which uses a specially written input handler routine). This routine either accepts solely numerical data (numbers 0-9) as input or any alphanumeric character string. The type of input expected by the routine is flagged by one of the procedure call arguments. A short beep is emitted if an incorrect entry is attempted.

The options provided this month will be sufficient for you to start organising your video library. The program will be completed next month with the remainder of the routines allowing the library to be sorted, hard copy to be printed, and searches for individual titles and tapes to be made.


```

10 REM Program Video Index
20 REM Version B 1.04
30 REM Author Keith Sumner
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 160
110 MODE7:PROCvars
120 REPEAT:PROCmenu:UNTILexit
130 CLS
140 END
150 :
160 ON ERROR OFF: *FX 4,0
170 MODE 7:REPORT
180 PRINT" at line ":ERL
190 END
200 :
1000 DEFPROCmenu:*FX21,0
1010 FO=0:PROChead(10,cy$+"VIDEO-INDEX"
)
1020 PRINTTAB(0,3)"0. Load data"
1030 PRINT"1. Input data""2. Edit data
"
1040 PRINT"3. Search""4. Sort"
1050 PRINT"5. Look through""6. Save da
ta"
1060 PRINT"7. Delete data"
1070 PRINT"8. Hardcopy""9. Print Label
"
1080 PRINT"E. Exit"
1090 PRINTTAB(18,18)yl$+"Entries made :
":Z%
1100 PRINTTAB(18,19)gr$+"Entries left :
":M%-Z%
1110 DIM P%-1:PRINTTAB(13,21)cy$+"Free
memory : ":HIMEM-P%:" bytes"
1120 PRINTTAB(0,14)rd$+"Option No. :á";
1130 REPEAT:K=GET-48:IFK=21 K=10
1140 UNTIL K>=0 AND K<11
1150 IFZ%=0 AND K>1 AND K<10 PROCinfo:P
ROCmenu
1160 IFK=0 PROCread ELSEIFK=1 PROCinput
1170 IFK=2 PROCedit
1180 IFK=5 PROClook
1190 IFK=6 PROCsave ELSEIFK=7 PROCKill
1200 IFK=10 exit=TRUE:*FX4,0
1210 ENDPROC
1220 :
1230 DEFPROCinfo
1240 VDU7,12
1250 PRINT"rd$+fl$+"MEMORY EMPTY"
1260 PROCspace(22):ENDPROC
1270 :
1280 DEFPROCinput
1290 PROChead(10,gr$+"INPUT")
1300 IF Z%=M% PRINTrd$+fl$+"MEMORY FULL
":PROCspace(22):ENDPROC

```

```

1310 PRINTTAB(3,2)"Number : ";
1320 REPEAT: N$=FNinpt(1,3,0):IF N$=""
UNTIL TRUE:ENDPROC
1330 e=0:FOR A%=1 TO Z%:IF LEFT$(N$(A%
),3)=N$ e=e+1
1340 NEXT:PRINTTAB(25,2)"Tape "N$TAB(25
,3);e+1" entries":UNTIL e<MAXENT
1350 REPEAT: PRINT TAB(5,4)"Type : E-";
1360 l$=FNinpt(1,3,1)
1370 UNTIL INSTR("120 180 240",l$)<>0
1380 PRINTTAB(1,6)"Category : ";
1390 C$=FNinpt(0,3,1)
1400 REPEAT: PRINTTAB(2,8)"Counter : ";S
PC(3);TAB(12,8);
1410 c$=FNinpt(1,4,1):UNTIL NOT(FNC(c$
))
1420 cl$=FNinpt(1,4,1):IFFNC(cl$) THEN1
440
1430 IFcl$<=c$ THEN1440
1440 REPEAT: Ti$=FNeditor(9):UNTIL Ti$<
>"
1450 Z%=Z%+1
1460 T$(Z%)=(Ti$+C$):N$(Z%)=(N$+c$+"-"+
cl$+l$+STRING$(6,CHR$(32)))
1470 A%=Z%:PROCleft:PROCinput
1480 ENDPROC
1490 :
1500 DEFPROClook
1510 LOCALK%:CLS:PROCcard
1520 PRINTTAB(2,6)rd$+"f0"wh$"- Menu."TA
B(2,7)rd$+"f1"wh$"- Jump."TAB(20,6)"Recor
d No."yl$": "
1530 A%=1:REPEAT
1540 PRINTTAB(33,6)SPC(3);TAB(33,6);A%
1550 PROCprint:REPEAT:*FX21,0
1560 IFA%=1 SOUND1,-10,200,6:PRINTTAB(3
3,6)"BOF"
1570 IFA%=Z% SOUND1,-10,230,6:PRINTTAB(
33,6)"EOF"
1580 K%=GET:UNTIL K%=138 OR K%=139 OR K
%=128 OR K%=129
1590 IF K%=129 THEN PROCcalculate
1600 IF K%=138 AND A%<Z% A%=A%+1
1610 IF K%=139 AND A%>1 A%=A%-1
1620 UNTIL K%=128
1630 ENDPROC
1640 :
1650 DEF PROCcalculate
1660 REPEAT
1670 PRINTTAB(2,8)fl$cy$"TO"fo$wh$"?":S
PC(4);:PRINTTAB(10,8);:A%=VALFNinpt(1,3,
1)
1680 UNTIL A%>0 AND A%<=Z%
1690 PROCclear(0,8,8,16):PROCclear(0,0,
5,38)
1700 ENDPROC
1710 :

```

```

1720 DEFPROCsplit
1730 C$=RIGHT$(T$(A%),3):R$=MID$(N$(A%),4,9)
1740 t$=LEFT$(T$(A%),LENT$(A%)-3)
1750 L$=MID$(N$(A%),13,3):D$=RIGHT$(N$(A%),3)
1760 d$=MID$(N$(A%),16,3):N$=LEFT$(N$(A%),3)
1770 ENDPROC
1780 :
1790 DEFPROCprint:PROCsplit
1800 IF K<5 AND FO=0 CLS:PROCcard
1810 PRINTTAB(26,9)N$TAB(7,10)L$TAB(14,11)C$TAB(15,18)
1820 IFD$<>"****" PRINTTAB(15,18)d$;" mins";SPC(3);TAB(15,20)R$;SPC(2)TAB(15,22)D$+" mins"+STRING$(3,CHR$(32)):GOTO1840
1830 PRINTTAB(15,18)fl$+yl$+d$+wh$+fo$+"mins"TAB(15,20)fl$+yl$+R$+fo$TAB(15,22)fl$+yl$+D$+fo$+wh$+"mins"
1840 VDU28,12,17,31,13,12:PRINTLEFT$(t$,79):VDU26
1850 ENDPROC
1860 :
1870 DEFPROCa
1880 PROCprint:FO=1:PROCspace(2)
1890 ENDPROC
1900 :
1910 DEFFNC(a$):=(l$="120" AND VALa$>M120 OR l$="180" AND VALa$>M180 OR l$="240" AND VALa$>M240)
1920 DEFPROCspace(Q):*FX21,0
1930 PRINTTAB(4,Q)"Hit <spacebar> to continue";REPEATUNTILGET=32
1940 PROCclear(0,Q,Q,38):ENDPROC
1950 :
1960 DEFPROCedit
1970 PROChead(15,yl$+"EDIT")
1980 PRINT'"Tape No. ":a$=FNinpt(1,3,0):IFa$="" ENDPROC
1990 q$="":FOR A%=1 TO Z%
2000 IFa$=LEFT$(N$(A%),3) ANDFO=0 PROCprint:FO=A%:ELSE2030
2010 PRINTTAB(0,1)"Is this the required tape (y/n)":q$=GET$
2020 IFINSTR("Nn",q$)>0 FO=0 ELSEA%=Z%
2030 NEXT:IFFO<>0 A%=FO:PROCed1:ENDPROC
2040 IFq$<>" " FO=1
2050 PROCnomatch:GOTO 1970
2060 ENDPROC
2070 :
2080 DEFPROCed1:PROCsplit
2090 VDU28,0,5,39,0:PROChead(15,yl$+"EDIT"):VDU26
2100 PRINTTAB(6,2)"Type : E-";
2110 l$=FNinpt(1,3,0):IFINSTR("120 180 240",l$)=0 THEN2100

```

```

2120 IF l$="" l$=L$:PRINTl$
2130 PRINTTAB(2,4)"Tape no. : ";
2140 n$=FNinpt(1,3,0)
2150 IF n$="" n$=N$:PRINIn$
2160 PRINTTAB(3,6)"Counter : "SPC(5);TAB(13,6);
2170 c$=FNinpt(1,4,0):IFFNC(c$) THEN2160
2180 IF c$="" c$=LEFT$(R$,4)
2190 REPEAT: REPEAT: PRINTTAB(13,6)c$-" ";SPC(4);TAB(18,6):c1$=FNinpt(1,4,0)
2200 UNTIL NOT(FNC(c1$))
2210 IF c1$="" c1$=RIGHT$(R$,4):PRINTc1$
2220 UNTIL c1$>c$
2230 PRINTTAB(2,8)"Category : ";
2240 ca$=FNinpt(0,3,0)
2250 IF ca$="" ca$=C$:PRINTca$
2260 N$(A%)=(n$+c$+"-"+c1$+l$)
2270 PROCleft:T$(A%)=t$+ca$
2280 PROCclear(0,2,8,22):PROCprint
2290 Ti$=FNeditor(1):IFTi$="" Ti$=t$
2300 T$(A%)=Ti$+ca$
2310 PROCclear(0,2,5,38):PROCclear(0,6,8,20):PROCprint
2320 PRINTTAB(0,3)"Is the edited card now correct (y/n)":q$=GET$
2330 IF q$="N" ORq$="n" PROCed1
2340 ENDPROC
2350 :
2360 DEFPROCread:*FX21,0
2370 PROChad(4,fl$+yl$+"Loading data file")
2380 IF Z%<0 VDU7:PRINT"Continue (y/n) ?":Y$=GET$:IFINSTR("Nn",Y$)>0 ENDPROC
2390 INPUT"Filename : "Z$:IFZ$="" ENDPROC
2400 Z%=1:Z=OPENIN(Z$)
2410 IF Z=0 PRINT"File Not Found":GOTO 2390
2420 PRINT'"Loading"'Z$
2430 REPEAT:INPUT#Z,T$(Z%),N$(Z%):Z%=Z%+1:UNTIL EOF#Z:CLOSE#Z
2440 Z%=Z%-1:ENDPROC
2450 :
2460 DEFPROCsave:PROChead(4,fl$+gr$+"Saving data")
2470 INPUT"Filename : "Z$:IFZ$="" ENDPROC
2480 y=1:Z=OPENOUT(Z$)
2490 REPEAT:PRINT#Z,T$(y),N$(y)
2500 y=y+1:UNTILy>Z%:CLOSE#Z:ENDPROC
2510 :
2520 DEFPROCvars:*FX225,128
2530 Z%=0:M%=200:MAXENT=8:exit=FALSE:*TV255
2540 M120=3238:M180=4295:M240=5518

```

```

2550 DIMIT$(M%),N$(M%),A(MAXENT),B$(4):*
FX4,1
2560 rd$=CHR$129:gr$=CHR$130:yl$=CHR$13
1:bl$=CHR$132:mg$=CHR$133:cy$=CHR$134:wh
$=CHR$135:fl$=CHR$136:fo$=CHR$137
2570 RESTORE2930:FORZ=0TO4:READB$(Z):NE
XT:ENDPROC
2580 :
2590 DEFPROCswap
2600 Z$=T$(I%):T$(I%)=T$(I%+S%):T$(I%+S
%)=Z$:Z$=N$(I%):N$(I%)=N$(I%+S%):N$(I%+S
%)=Z$:F%=TRUE
2610 ENDPROC
2620 DEFPROChead(X,A$):CLS:PRINTTAB(X)C
HR$141;A$;CHR$135;CHR$140TAB(X)CHR$141;A
$;CHR$135;CHR$140:ENDPROC
2630 :
2640 DEFFNinpt(F2,ML,Q)
2650 LOCALB,B$:a$="":PRINTSPC(ML);STRIN
G$(ML,CHR$8);
2660 *FX21,0
2670 REPEAT: B$=GET$:B=ASCB$
2680 IF B=13 THEN IF(Q=1 AND LENA$=ML)
OR(Q=0 AND a$="") OR (Q=0 AND LENA$=ML)
THEN UNTIL TRUE:=a$
2690 IF B=127 AND a$="" THEN UNTIL FALS
E
2700 IF B=127 a$=LEFT$(a$,LENA$-1):PRIN
TB$;:UNTIL FALSE
2710 IF LENA$=ML OR B<32 OR B>126 THEN
VDU7: UNTIL FALSE
2720 IF NOT(F2=0 OR B=32 OR (B>=48 AND
B<=57)) VDU7: UNTIL FALSE
2730 PRINTB$;:a$=a$+B$:UNTIL FALSE
2740 :
2750 DEFFNcalc(a$)
2760 LOCALa%,B%,C%,L%
2770 IF1$="180" RESTORE2880
2780 IF1$="120" RESTORE2860
2790 IF1$="240" RESTORE2910
2800 L%=-1:READA$:REPEAT:READB$
2810 C%=A$:A%=B$:L%=L%+1
2820 UNTIL (VALa$>=C% ANDVALa$<A%) ORB%=
9999
2830 t=VAL1$-L%*5:ra=5*((VALa$-C%)/(A%-
C%))
2840 =INT(t-ra+.5)
2850 REM E-120
2860 DATA0,100,200,300,400,502,606,714,
824,937,1053,1172,1296,1422,1552,1687,18
26,1976,2123,2283,2450,2640,2818,3021,32
38,9999
2870 REM E-180
2880 DATA0,82,165,245,330,415,500,590,6
80,770,860,950,1050,1150,1250,1350,1450,
1550,1650,1765,1875,1985
2890 DATA2100,2230,2350,2475,2600,2750,

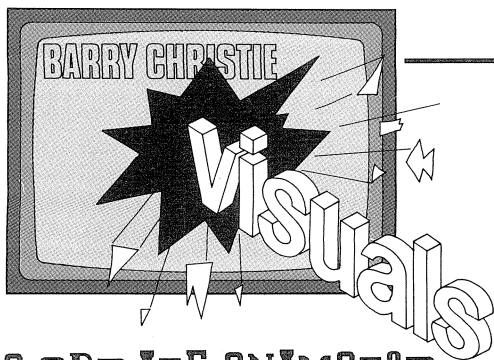
```

```

2880,3025,3180,3340,3505,3680,3870,4075,
4295,9999
2900 REM E-240
2910 DATA0,78,155,235,315,395,475,558,6
40,725,800,895,980,1070,1158,1248,1340,1
430,1525,1620,1715,1810,1910,2012,2115,2
218,2325
2920 DATA2431,2540,2652,2766,2882,3000,
3123,3250,3375,3505,3640,3780,3922,4070,
4222,4380,4546,4720,4900,5094,5298,5518,
9999
2930 DATA"Film title ","Section of titl
e ","Tape number ","Film category ","Min
. time left on a tape "
2940 DEFPROCleft
2950 LOCALa$,C%,V%,ti,to,re,l$,te$,tf$
2960 FOR C%=0 TO MAXENT:A(C%)=0:NEXT:C%
=0:to=0:l$=MID$(N$(A%),13,3)
2970 FOR V%=1 TO Z%
2980 IF (LEFT$(N$(V%),3)=LEFT$(N$(A%),3)
) AND (MID$(N$(V%),13,3)<>MID$(N$(A%),13,
3)) THENte$=LEFT$(N$(V%),12):tf$=RIGHT$(
N$(V%),3):N$(V%)=te$+l$+tf$
2990 IF (LEFT$(N$(V%),3)=LEFT$(N$(A%),3)
) ANDA%<>V% A(C%)=V%:C%=C%+1
3000 NEXT
3010 ti=ABS(FNcalc(MID$(N$(A%),4,4))-FN
calc(MID$(N$(A%),9,4))
3020 to=to+ti:q$=RIGHT$("000"+STR$ti,3)
3030 N$(A%)=LEFT$(N$(A%),15)+q$
3040 FOR V%=0 TO C%-1:to=to+VALMID$(N$(
A(V%)),16,3):NEXT
3050 re=VAL(l$)-to:IFre<0 q$="****" ELSE
q$=RIGHT$("000"+STR$re,3)
3060 N$(A%)=LEFT$(N$(A%),18)+q$
3070 FOR V%=0 TO C%-1
3080 N$(A(V%))=LEFT$(N$(A(V%)),18)+q$
3090 NEXT:ENDPROC
3100 :
3110 DEFPROCnomatch:LOCALQ$,Y,SP,G,D
3120 IF FO>0 Q$=rd$+"NO FURTHER MATCH":
Y=5:SP=3
3130 IF FO=0 Q$=rd$+"NO MATCH FOUND":Y=
20:SP=22
3140 FOR G=0 TO 20:PRINTTAB(0,Y)SPC(38)
TAB(G,Y)Q$:FORD=1TO250:NEXT:NEXT
3150 PROCspace(SP):ENDPROC
3160 :
3170 DEFPROCkill:PROChead(11,mg$+"Delet
e")
3180 PRINT""Tape No. ":a$=FNinpt(1,3,
0):IFa$="" ENDPROC
3190 F%=1:ENT=0:ent=0:FORA%=1TOZ%:IFLEF
T$(N$(A%),3)=a$ ent=ent+1:A(ent)=A%
3200 NEXT:IFent=0 PROCnomatch:PROCKill:
ENDPROC

```

Continued on page 68



A SPRITE ANIMATOR

In his last Visuals column, Barry implements a Sprite Animator to put life into sprites created with last month's program.

The program developed last month allows the creation of multicolour sprites of any size. The Animator featured here will allow you to move these sprites around the screen. In fact there are four programs listed this month, so it may sound a little complicated. The main program (listing 1) is the Animator itself. This creates a block of code which includes moving routines (in machine code) together with sprite definition codes. You then create a Basic program which installs the sprite code created with the Animator, and which moves your sprites around by making calls to the sprite code. An example program is given in listing 2. But before we see how this works, we will return to listing 1.

THE ANIMATOR

To get the Animator working, first type in listing 1, and save away the program. Before you can run it you will need a datafile (made from Basic DATA statements) which it will automatically append to itself. All this data does is to tell it what mode the sprites are to be used with, and what the sprite filenames are. So to run the Animator you need a block of data similar to that in listing 3. Its structure should be clear from the REM statements. It should always start at line 5000, and should give the required mode, followed by the names of each sprite to be called. The last item is the word "END". In our example, we have included ten mode 4 sprites.

For the purposes of example, listing 4 is a short program which will actually create the ten (identical) sprite files. So if you are going to use

the Animator with the data block given in listing 3, you should first run listing 4 to create the ten sprites - phew! Now we can run the Animator. It will ask you for the name of your sprite datafile (the name under which you saved listing 3). When linking is complete, it will request a filename under which to save the compiled sprite mover code. When the job is done the program will display various data, and you should note for future use the value of HIMEM given.

```
=====
BEEBUG Mode 4 Sprite Animator
< Various totals relating to sprites >

NUMBER of sprites linked = 10
DATA table size (bytes) = 320
INFO table size (bytes) = 80
CODE mover size (bytes) = 282
OVERALL length (bytes) = 682
CODE (LOAD/EXEC) address = &5536

< Set HIMEM to &5500
=====
By B. Christie - (C) BEEBUG 1988
=====
```

SPRITES FROM BASIC

We now turn to the Basic program which will make use of the mover code. Typically this might be a game of some sort, or any program which would benefit from an animated display. As mentioned above, a sample program is provided as listing 2. It begins by assigning mode, and setting HIMEM to &5500 (the value provided by the Animator when our example sprite code was generated). This is essential as the sprite code will be held above HIMEM. Next (at line 60) is the command which loads in and installs the sprite code (use the filename which you previously supplied for saving the mover code).

The mover code provides the user with three machine code calls. The call addresses are always as follows:

```
Step Sprite &900
Move Sprite &903
Put Sprite &906
```

Put Sprite (called in line 110) places the sprite at a particular position on the screen. Before the call is made, A% should contain the number of the sprite, and X% and Y% the required co-ordinates of the sprite (see below).

Once a sprite is on the screen, it can be moved by using the other two calls. To do this you must first call Step Sprite (line 120) with A% containing the Sprite number, and X% and Y% containing the co-ordinate increment required in each re-drawing of the sprite. Then each time that you call Move Sprite (line 160) with A% containing the sprite number (X% and Y% are not required), the sprite is moved by the increment defined with Sprite Step. To remove a sprite at any time, just make a call to Put Sprite giving the present co-ordinates of the sprite to be deleted.

To keep the routines fast, the X and Y co-ordinate ranges are as follows:

Y (all modes) range 0-255
X (modes 0, 1 & 2) range 0-79
X (modes 4 & 5) range 0-39

Position 0,0 corresponds to top left. Space is very tight this issue, and I must leave you to experiment with these routines. One last point, you may wish to use *FX19 before moving any sprite, to make things smoother - but it will slow everything down.

This is the last of our Barry Christie Visuals series, but we shall continue to cover visual aspects of the Beeb in future issues, and Barry will be contributing programs in future issues of BEEBUG and RISC User.

Listing 1

```

10 REM Program Sprite Animator
20 REM Version B 0.4
30 REM Author Barry Christie
40 REM Beebug March 1988
50 REM Program subject to copyright
60 :
100 MODE 7
110 ON ERROR MODE7:GOTO 2150
120 HIMEM=PAGE+62000
130 LOMEM=TOP+6400
140 SPTOP=TOP-2
150 PROCspriteinit1
160 PROCspriteinit2
170 PROCspritelink
180 PROCTableslink
190 PROCdetailinfo
200 HIMEM=&7C00:VDU26:PRINTTAB(30,20);
210 END
220 :
230 DEF PROCspritelink
240 READ spritename$
250 REPEAT
260 print$="Linking sprite '"+spritena
me$+"":CLS

```

```

270 PRINT TAB((39-LEN(print$))/2,8)pri
nt$
280 PROCloadsprite
290 READ spritename$
300 UNTIL spritename$="END"
310 !SPTOP=&FF0D
320 ENDPROC
330 :
340 DEF PROCloadsprite
350 channel=OPENUP(spritename$)
360 sprtmode%=BGET#channel AND &07
370 sprtXwdh%=BGET#channel
380 sprtYwdh%=BGET#channel
390 CLOSE#channel
400 IF sprtmode%=mode% THEN PROCsprite
isvalid ELSE PROCspriteinvalid
410 ENDPROC
420 :
430 DEF PROCspriteisvalid
440 sprtXwdh%=(sprtXwdh% DIV pixbytes%
)-((sprtXwdh% MOD pixbytes%)<0)
450 PROCspritedata:PROCspriteinfo:link
%=link%+1
460 SOUND &15,-15,100,1:SOUND &17,-12,
150,1
470 ENDPROC
480 :
490 DEF PROCspriteinvalid
500 CLS:VDU7
510 PRINT TAB(4,7)"Error, this is a mo
de ";sprtmode%;" sprite"
520 PRINT TAB(7,9)"Press <SPACE> to co
ntinue"
530 REPEAT:UNTIL GET=32
540 ENDPROC
550 :
560 DEF PROCspritedata
570 sprtsize%=sprtXwdh%*sprtYwdh%
580 dataarea%=dataarea%-sprtsize%
590 OSCLI("LOAD "+spritename$+" "+STR$
~(dataarea%-3))
600 ENDPROC
610 :
620 DEF PROCspriteinfo
630 infoarea%!=00=dataarea%-offset%
640 infoarea%&02=sprtXwdh%
650 infoarea%&03=sprtYwdh%
660 infoarea%!=04=0
670 infoarea%=infoarea%+8
680 ENDPROC
690 :
700 DEF PROCTableslink
710 infosize%=8*link%:CLS
720 PRINT TAB( 6,6)"Enter sprite mover
filename"
730 PRINT TAB(10,8)">>> ..... <<
<"
740 INPUT TAB(14,8) movername$
750 PROCLinkcode(&900,&900)
760 assemble%=dataarea%-infosize%-code
size%

```

```

770 execaddr%=scrnaddr%-(7C00-dataarea%)-infosize%-codesize%
780 PROClinkcode(assemble%,execaddr%)
790 PROClinkinfo
800 savecode$="SAVE "+movername$+" "+S
TR$~(assemble%)
810 OSCLI(savecode$+" 7C00"+STRING$(2,
" "+STR$~(execaddr%)))
820 PRINT TAB(10,10)"<Linkage Complete
d>":VDU7
830 ENDPROC
840 :
850 DEF PROClinkinfo
860 infoarea%=dataarea%-infosize%
870 FOR info%=0 TO infosize%-1
880 info%?infoarea%=info%?HIMEM
890 NEXT info%
900 ENDPROC
910 :
920 DEF PROClinkcode(codearea%,coderun
n%)
930 adrlo=&70:adrhi=&71
940 width=&72:depth=&73
950 xcord=&74:ycord=&75
960 tempw=&76:tempd=&77
970 temps=&78
980 FOR pass%=4 TO 6 STEP 2
990 P%=coderunn%
1000 O%=codearea%
1010 [ OPT pass%
1020 .setupvectors
1030 LDY #&08
1040 .transfervectors
1050 LDA vectors,Y:STA &900,Y:DEY
1060 BPL transfervectors
1070 RTS
1080 .vectors
1090 EQUW &4C:EQUW steplot
1100 EQUW &4C:EQUW movplot
1110 EQUW &4C:EQUW putplot
1120 .spritenummer
1130 ASL A:ASL A:ASL A
1140 STA temps:TYA:LDY temps
1150 RTS
1160 .steplot
1170 JSR spritenummer
1180 STA infotable+&07,Y:TXA
1190 STA infotable+&06,Y
1200 RTS
1210 .putplot
1220 JSR spritenummer
1230 STA infotable+&05,Y:TXA
1240 STA infotable+&04,Y:JMP plot
1250 .movplot
1260 JSR spritenummer
1270 JSR plot:LDY temps
1280 CLC:LDA infotable+&04,Y:ADC infota
ble+&06,Y:STA infotable+&04,Y
1290 CLC:LDA infotable+&05,Y:ADC infota
ble+&07,Y:STA infotable+&05,Y
1300 .plot
1310 LDA infotable+&00,Y:STA store+&01

```

```

1320 LDA infotable+&01,Y:STA store+&02
1330 LDA infotable+&02,Y:STA width
1340 LDA infotable+&03,Y:STA depth
1350 LDA infotable+&04,Y:STA xcord
1360 LDA infotable+&05,Y:STA ycord
1370 LDA depth:STA tempd
1380 .depthloop
1390 JSR calcaddr
1400 LDX #&00:LDY #&00
1410 LDA width:STA tempw
1420 .widthloop
1430 .store
1440 LDA &BEEB,X
1450 EOR (adrlo),Y:STA (adrlo),Y:INX
1460 CLC:TYA:ADC #&08:TAY
1470 DEC tempw:BNE widthloop
1480 TXA :CLC:ADC store+&01:STA store+&
01
1490 LDA store+&02:ADC #&00:STA store+&
02
1500 INC ycord:DEC tempd:BNE depthloop
1510 RTS
1520 .calcaddr
1530 LDA #&00:STA adrhi
1540 LDA &75 :LSR A:LSR A
1550 AND #&FE:TAY:LDA &74 :ASL A
1560 ASL A:ROL adrhi:ASL A:ROL adrhi
1570 CLC:ADC addrtable+&00,Y:STA adrlo
1580 LDA addrtable+&01,Y:ADC adrhi:STA
adrhi
1590 CLC:LDA &75:AND #&07:ADC adrlo:STA
adrlo
1600 LDA adrhi:ADC #&00:STA adrhi
1610 RTS
1620 .addrtable EQUW STRING$(64,"0")
1630 .infotable
1640 ]
1650 NEXT pass%
1660 codesize%=P%-coderunn%
1670 FOR address%=0 TO 31
1680 !(O%-64+2*address%)=address%*rowby
tes%+scrnaddr%
1690 NEXT address%
1700 ENDPROC
1710 :
1720 DEF PROCdetailinfo
1730 delay=INKEY(200):CLS
1740 PRINT "< Various totals relating t
o sprites >"
1750 PRINT"  NUMBER of sprites linked
=";link%
1760 PRINT"  DATA table size (bytes)
=";&7C00-dataarea%
1770 PRINT"  INFO table size (bytes)
=";link%*8
1780 PRINT"  CODE mover size (bytes)
=";codesize%
1790 PRINT"  OVERALL length (bytes)
=";&7C00-dataarea%+link%*8+codesize%
1800 PRINT"  CODE (LOAD/EXEC) address
=";&~execaddr%
1810 PRINT

```



```

1820 PRINT TAB(8)"< Set HIMEM to &";~(e
xecaddr% DIV &100)*&100;" ";
1830 ENDPROC
1840 :
1850 DEF PROCspriteinit1
1860 dashes$=" "+STRING$(38,"=")+" "
1870 colour$=CHR$129+CHR$157+CHR$135+CH
R$141
1880 title1$=colour$+"< Multi-Mode Spr
ite Animator > "+CHR$156
1890 title2$=colour$+"By B. Christie -
(C) BEEBUG 1988 "+CHR$156
1900 PRINT TAB(0,21)dashes$;title2$;tit
le2$;dashes$:VDU30,11,11
1910 PRINT TAB(0,0)dashes$;title1$;titl
e1$;dashes$
1920 VDU28,1,20,38,4
1930 PRINT TAB(3,7)"Enter name of spri
te data file ... "
1940 PRINT TAB(10,9)">>> ..... <<
<"
1950 INPUT TAB(14,9)datafile$
1960 OSLI("LOAD "+datafile$+" "+STR$(
TOP-2))
1970 CLS:VDU26:RESTORE:READ mode$
1980 title1$=colour$+"BEEBUG Mode "+mo
de$+" Sprite Animator "+CHR$156
1990 PRINT TAB(0,0)dashes$;title1$;titl
e1$;dashes$
2000 VDU28,1,20,38,4
2010 ENDPROC
2020 :
2030 DEF PROCspriteinit2
2040 mode%=VAL(mode$):link%=0
2050 pixbytes%=2^(3-(mode% MOD 4))
2060 rowbytes%=&140*(2-(mode% DIV 4))
2070 scrnaddr%=&3000+(mode% DIV 4)*&280
0
2080 infoarea%=HIMEM
2090 dataarea%=&7C00
2100 offset%=dataarea%-scrnaddr%
2110 ENDPROC
2120 :
2130 REM Error handler
2140 erf=FALSE
2150 IF ERR=17 THEN PRINT"<ESCAPE> pr
essed, linking aborted ...":erf=TRUE
2160 IF ERR=214 THEN PRINT"<DATA FILE>
";datafile$;" not found ...":erf=TRUE
2170 IF ERR=222 THEN PRINT"<SPRITE> "
;spriteName$;" not found ...":erf=TRUE
2180 IF NOT erf THEN REPORT:PRINT" at l
ine ";ERL
2190 !SPTOP=&FF0D:CLOSE#0:VDU7,7
2200 END
2210 :
2220 REM Sprite data sub-program
2230 REM is appended after this point
*****

```

Listing 2

```

10 REM Complete sprite demo program
20 REM ref .>SprDemo
30 :
40 MODE 4
50 HIMEM=&5500
60 *RUN SPRcode
70 INPUT"Enter number of sprites (1-
10) "sprites%
80 stepsprite%=&900:movesprite%=&903:
putsprite%=&906:CLS
90 CLS:VDU23,1,0;0;0;0;
100 FOR I%=0 TO sprites%-1
110 A%=I%X%=I%*2:Y%=I%*8:CALL putspri
te%
120 A%=I%X%=2-RND(4):Y%=32-RND(64):CA
LL stepsprite%
130 NEXT
140 REPEAT
150 FOR I%=0 TO sprites%-1
160 A%=I%:CALL movesprite%
170 NEXT
180 UNTIL FALSE
*****

```

Listing 3

```

5000 REM Data for Animator
5010 REM ref.>DemoData
5020 REM Screen mode
5030 DATA 4
5040 REM filenames of sprites
5050 DATA sprite0,sprite1,sprite2
5060 DATA sprite3,sprite4,sprite5
5070 DATA sprite6,sprite7,sprite8,sprit
e9
5080 REM End of data marker
5090 DATA END
*****

```

Listing 4

```

10 REM Program creates 10 sprites
20 REM For use with mode 4 demo
30 REM ref .>SprMake
40 MODE 7
50 FOR sprite%=0 TO 9
60 RESTORE
70 X=OPENOUT("sprite"+STR$(sprite%))
80 FOR I%=1 TO 5
90 READ data$
100 FOR J%=1 TO LEN(data$) STEP 2
110 BPUT#X,EVAL("&"+MID$(data$,J%,2))
120 NEXT: NEXT
140 CLOSE#X
150 NEXT sprite%
160 PRINT"Sprites created":END
170 :
180 REM demo sprite data
190 DATA "041010"
200 DATA "03C00FF01998333C"
205 DATA "711E799EBEFDBEFD"
210 DATA "BE7DB3CD799E7C3E"
215 DATA "3FFC1FF80FF003C0"

```



MASTER EMULATION ROM

In this and next month's issues, Bernard Hill evaluates two software packages which offer to turn your Model B into a Master.

Can you really turn your Model B into a Master? It's a nice idea, especially if you can retain your Model B to run all those Master incompatible games (sorry, programs). MER from Dabs Press claims to do just that.

Product	Master Emulation ROM
Supplier	Dabs Press 76 Gardner Road, Prestwich, Manchester M25 7HU. Tel. 061-773 2413
Price	£19.95 inc VAT - ROM £14.95 inc VAT - 5.25" disc £16.95 inc VAT - 3.5" disc

To attempt to turn a model B into a Master requires more than just software. Additional hardware, particularly RAM for various purposes is required as well. MER implements most of the Master's additional star commands, but its full potential will only be realised if you add some of the missing hardware facilities and this I have covered first in some detail.

The MER software is available in a variety of formats, and in all cases is accompanied by extensive documentation. Let's examine then the extent to which a Model B can be upgraded to a Master using the MER package and relevant hardware add-ons.

RAM UPGRADES TO A MASTER

If you could upgrade your Model B (or even B+) to a Master you would need quite a lot of extra memory, and this comes in various types:

1. Shadow RAM.

Firstly, you could add shadow RAM to your Model B (by Aries, Watford or Solidisk). MER works well with these; in fact it does so by behaving quite independently, leaving the star commands which drive your particular shadow RAM quite intact. There is no attempt to implement MODE 128 type statements, and even *SHADOW does not necessarily work with all shadow RAM boards.

2. Sideways RAM.

Secondly, your Model B could have banks of sideways RAM added, and that's fine too (in fact MER can handle all the known access protocols - e.g. from Solidisk, Watford, ATPL, etc.). What it can't do is handle a mixture of protocols, for instance a 16K module of Acorn User RAM sitting alongside an early Solidisk system, or with an ATPL board. But it works particularly well with sideways RAM all of one type, such as the Watford ROM/RAM board, or the Solidisk TwoMeg 128, and up to 64K can be accommodated.

3. CMOS RAM.

Thirdly, the Master has CMOS RAM, in which it keeps its *STATUS/*CONFIGURE options. MER's implementation here is excellent. If you have the Solidisk Real Time Clock installed in your Model B, then MER will use this for its 50 bytes of CMOS RAM just as the Master does. If not, then you can use the last 50 bytes of a nominated sideways RAM bank (including the one occupied by MER itself if you're running a ROM image) - preferably with battery backing. Failing this, MER will use bytes &140-&171 of the 6502 stack. If you have no battery-backed RAM then, to avoid losing all your configurations at power-down, MER has a *SCONFIG command which saves all this information to a disc file, and *LCONFIG will reload it when you switch on again. Just as on the Master, these 50 bytes hold options such as loud/quiet beep, UNPLUGGED ROMs, default modes, disc speeds, printer destinations, Caps Lock status etc. This all works splendidly and the full range of *CONFIGURE and *STATUS commands is implemented, with commands identical to those in the Master's Operating System.

4. Private RAM.

So far so good. But one final type of RAM which you can't add to your Model B is Private RAM: that's the bit of the Master which keeps PAGE down at &E00, so don't expect MER to shift PAGE down from &1900. In fact, MER needs some workspace too, and the Dabs Press made the decision NOT to use workspace (like the DFS) which would raise PAGE. Neither does MER run solely from sideways RAM and so use its own SWR image as workspace (like their other product SideWriter). To my mind that's a great pity, because MER now has to use other areas, including some of pages 6, 7 and 9 causing workspace clashes with other ROMs (such as BEEBUG's own Master ROM).

FACILITIES

So you've got your Master Emulation ROM installed. Just what have you got you didn't have before? Briefly put, you have the commands which are available on the Master and the Compact. For the first time you can use *APPEND to supplement *BUILD; *SPOOLON to supplement *SPOOL; *EX to add to *INFO (even on DFS, tape and ROM filing systems!); *CREATE to add to *SAVE; *FREE and *MAP to add to *CAT on the DFS and *REMOVE to go with *DELETE. You could fit a 1770 interface to run the ADFS and yet keep the ADFS *UNPLUGged to conserve PAGE. You can allocate sideways RAM as DATA with *SRDATA or *SRROM, and even call ID if you have View 3. If you have no Real Time Clock fitted then *TIME gives one second before the year 2000, like the Compact.

In fact the whole system FEELS just like a Master, right down to the "Acorn MOS" prompt on Break (the "BBC Computer 32K" has gone). Your Beeb even enters a 'No language' environment if you *CONFIGURE LANG to a socket which isn't a language, and the only difference from a real Master which I have spotted is, as mentioned above, that shadow RAM support is left to your shadow support ROM. And in order to 'disable' your Master there's a *MODELB command which completely kills MER and leaves you with your old system until powered-up again. The kids can still run Elite!

TEMPORARY FILING SYSTEMS

But the most impressive implementation is an almost-complete emulation of the Master's temporary filing system commands. This may be nothing new to Master users, but to be able to *-TAPE-CAT, *-ROM-EX or even *-ADFS-CDIR without interrupting your DFS working is quite an eye-opener to me! And *MOVE too? Many times I have wanted to make a second copy of a data file on the same disc but under a new name, and have to resort to *LOAD and re-*SAVE, but now I've got *MOVE <file1> <file2>. And if you have got an ADFS, from ADFS to DFS too - and vice versa - what a joy (even if it does take an age).

But I dare say you noticed the description "almost complete". What's missing? Nothing really, but we do come bang up against that workspace problem. Perhaps because MER uses

page 9, I've not managed to successfully *MOVE between tape (or RFS) and disc. Sometimes the file is corrupt when MOVED, and sometimes it just crashes. LOAD "-TAPE-name" and reSAVE is fine, however. And perhaps the most serious drawback I've spotted is that I can't SAVE and LOAD from View 2.1 (page 7 being used for workspace is naughty as it really is the property of the current language) although READ and WRITE are fine. And anyway View 3 doesn't show this problem. David Atherton at Dabs Press tells me that they are looking into these problems.

COMPATIBILITY

The launch of a product like MER (at such a reasonable price) takes considerable courage for Dabs Press. Not only do they have to try to match all of the third party hardware add-ons but they also have to aim for compatibility with Master software. Here the issues are perhaps clearer to see, or at least it is easy to predict what won't work. First, the Master has a 65C12 processor with extra instructions which the 6502 hasn't, and you can't run machine-code software which uses those. Secondly, there is software which needs the extra space, which the Master's page value of &E00 gives (or machine code which loads into this area) - that's out too. And then the Master software may need some of the workspace which MER uses (such as View 2.1, and the BEEBUG Master ROM). Finally, there are some aspects of the Master OS which can't be emulated, such as the new OS calls, e.g. OSWRSC at &FFB3 (Bang goes my Colossus 4 64K version), or ROM polling. For this reason you can't expect MER to run all Master software, but Dabs Press claim that their ROM is compatible with 90-95% of what's available.

One final point. If you buy MER, be sure you have a copy of version 2.01 (it's the one with an addenda sheet for the manual). If you have version 1.0, then Dabs will exchange it free of charge, and David Atherton also assures me that he operates a money-back guarantee if you have tried to work MER with some strange alien hardware and failed. That has to be a very fair offer.

Next month Bernard Hill takes a look at the Solidisk Real Time Clock mentioned in this review and provides some final thoughts on both products.

ⓑ

1st COURSE

Character Control

Mike Williams explores a variety of techniques for use in the important subject of character manipulation.

Character manipulation is a very important part of computer programming, and more experienced programmers will confirm that this forms a major part of many of their programs. Input, output, graphics and file handling are just

four of the topics which depend heavily on an ability to process characters. We shall be looking at the various character handling functions in BBC Basic, and at how these can be combined in a number of useful routines.

IN THE BEGINNING

Variables and arrays that are used to store characters always have names with a '\$' character, names such as `date$`, `word$` and so on. Similarly, all those Basic functions that return one or more characters also have names terminated with a '\$' sign. There are also a number of functions which we shall be looking at which take one or more characters as their argument but return a number.

There is, of course, no way that any digital computer can literally store characters in its memory. Instead, all characters are assigned a code value called an ASCII code (ASCII stands for American Standard Code for Information Interchange). All computers nowadays use this code, and you will find the codes detailed in the back of your User Guide.

Unfortunately, this code has two defects. It was devised in the USA, hence the inclusion of '#' and '\$', and the problems that can arise over the use of 'E'. This was not included in the original code, and hence there is no ASCII standard for this character. Further, the range of characters that computers are called upon to handle has also tended to increase, and this can again result in problems over the use of non-

standard (and hence varying) codes. All BBC micros use the same set of ASCII codes, and thus you should encounter no problems unless you roam outside the BBC world, such as

in the use of a printer which is intended for use with a wide variety of different makes of micro.

Finally, to complete the terminology, a series of characters following one after the other, is usually called a character string (or just string for short). But note that a string may contain only one character, while a 'null' string containing no characters at all is quite feasible as we shall find later. If that were not bad enough, there is also a so-called 'null' character; it's the one with ASCII code zero, and is represented in programs as "". You can think of the null character as the string equivalent of zero in arithmetic.

LEFT, RIGHT AND CENTRE

Most people begin their exploration of string manipulation by learning how to use the three functions `LEFT$`, `MID$` and `RIGHT$`. As expected these functions will extract from an existing string the left, middle or right portions respectively. For example:

```
alphabet$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
A$=LEFT$(alphabet$,8)
B$=MID$(alphabet$,9,10)
C$=RIGHT$(alphabet$,3)
```

will have the following effects. After assigning the 26 letters of the alphabet to the string variable `alphabet$`, `A$` is assigned the first 8 letters (i.e. "ABCDEFGH"), `B$` is assigned 10 characters starting at the ninth character (i.e. "IJKLMNOPQR"), while `C$` is assigned the right-most three characters (i.e. "XYZ"). As with numbers, the same variable can appear on both the left and right of an assignment statement. For example:

```
B$=LEFT$(B$,3)
```

would result in the original contents of `B$` being replaced by the first three characters.

The `RIGHT$` function is generally less useful than the other two. Very often the need is to extract the righthand part of a string starting from a particular character, often when the total length of the string is unknown. One solution is to use the `LEN` function to determine the length of the string thus:

```
B$=RIGHT$(text$,LEN(text$)-5)
```

This example results in all the characters of

text\$ to the right of character five being assigned to the variable B\$. A neater solution is to use a shortened version of MID\$:

```
B$=MID$(text$,6)
```

When only one numeric parameter is given with MID\$, then all the characters from that position onwards are extracted (in the example, 6 is the starting position, one more than 5). In practice, most string manipulation uses this shortened form of the MID\$ function, and the LEFT\$ function.

PLAYING WITH STRING

Now we know enough to begin to write some useful routines. One of the most frequent requirements is to extract each character of a string in turn. This can be achieved with a simple FOR-NEXT loop, as in this short example:

```
100 INPUT word$
110 FOR I=1 TO LEN(word$)
120 letter$=MID$(word$,I,1)
130 PRINT letter$
140 NEXT I
```

Line 100 waits for any character string to be typed in, and the loop will then list each letter in turn down the screen. Try it and see. Here's another example. See if you can work out what this will do before trying it in on your machine:

```
100 INPUT word$
110 FOR I=1 TO LEN(word$)
120 PRINT LEFT$(word$,I)
130 NEXT I
```

Replace line 120 with:

```
120 PRINT MID$(word$,I)
```

In fact the result will look even better if you change line 120 to:

```
120 PRINT SPC(I-1);MID$(word$,I)
```

These may seem trivial examples, but the facility to extract and manipulate strings as we have in these examples is the foundation for more extensive usage. As a test of your own ability, try to write a modification of line 120 above which will print any word with each letter in turn replaced by a space (see illustration) - answer at the end of this article.

SEARCHING EXAMINATION

One common requirement is to be able to search a string for the presence of one or more characters. This is what the INSTR function is designed to do. For example, we might write:

```
REPEAT
INPUT"Answer Y or N" ans$
OK=INSTR("YyNn",ans$)
```

UNTIL OK>0

Many programs have instances where a yes/no answer is required to some question, and there are various ways of handling this. In the above example, the response to the question is assigned to ans\$ (assumed to be a single character). The INSTR function then searches the first string specified (the string "YyNn") to see if it can match the string ans\$. If a match is found then the position of the matching character is assigned to the variable OK. If no match is found then zero is assigned. Putting our two statements inside a REPEAT-UNTIL loop as above will mean that the question is repeatedly asked until one of the desired letters (YyNn) is input.

Be careful, though, in using the result of the INSTR function in a logical way. If you are ever tempted to use the combination:

```
NOT INSTR(. . . .
```

then you are likely to get misleading results (see Beginners Start Here Vol.4 No.2).

Another common occurrence is when we need to check a string for a particular character, for example:

```
INPUT"Which option?" option$
ch=INSTR(option$,"/")
IF ch THEN mod$=MID$(option$,ch+1):
option$=LEFT$(option$,ch-1)
```

Here, we input an option which may or may not have '/' followed by further characters appended at the end. The INSTR function searches for this character, and if found the characters following the '/' are assigned to mod\$, and option\$ is assigned that part of the input string to the left of the '/'. Note how the '/' character itself is dropped altogether.

There are two further examples which are well worth looking at. Suppose we want to find all the occurrences of a particular character in a string. Here is an example:

```
100 INPUT"Enter any character" char$
110 INPUT"Enter any string" string$
120 pos=0
130 REPEAT
140 pos=INSTR(string$,char$,pos+1)
150 IF pos>0 THEN PRINT pos
160 UNTIL pos=0
```

This uses a variation on the INSTR function in which a third numeric parameter is included. This determines the position within the search string at which the search will commence. To

```
ASINGSTOKE
B SINGSTOKE
BA INGSTOKE
BAS NGSTOKE
BASI GSTOKE
BASIN STOKE
BASING TOKE
BASINGS OKE
BASINGST KE
BASINGSTO E
BASINGSTOK
```

start with **pos** is set to zero so that the search starts at the first character (position **pos+1**). If the character being searched for is found, then **pos** will be assigned the position in which it occurs, and the next search commences at the character following that occurrence, hence the **pos+1** (hence the reason why **pos** has to be set to zero initially, not one). The REPEAT-UNTIL loop terminates when no more occurrences of the specified character are found.

We can use this idea as the basis for a second type of search, one where we want to find all the occurrences of more than one (single) character. Amend the last example with the following lines:

```
100 flag$=".,;:!?"  
105 FOR I=1 TO LEN(flag$)  
106 char$=MID$(flag$,I,1)  
.....  
170 NEXT I
```

Now, any string input will be searched for all the occurrences of each of the characters in **flag\$**.

VERSATILE INPUT FUNCTION

To conclude this month's article, I want to give you a more substantial example of character manipulation in action. In this case it is an input function. It is often desirable to implement one's own input function rather than use the standard INPUT of Basic. It gives us much greater control over what is happening, allowing the characters input to be checked in any way we wish. In this case, the function, called **FNinput**, allows a prompt to be specified as well as a position on the screen (**x,y**), the maximum number of characters to be input (**width%**), and a so-called 'paint' character which is used to paint or mark the data entry field on the screen. I suggest you type in the listing and try it out before proceeding any further.

The variable **char\$** holds each character as it is typed in, and the variable **pos** records the current entry position within the data entry field. The variable **string\$**, initially set to a null entry, holds the current form of the complete input string. Three possibilities are catered for: if Delete is pressed (ASCII code 127) the last character is deleted both from the screen and from the variable **string\$**. If the entry field is full, a 'bleep' is sounded. Otherwise the character is displayed and appended to the variable **string\$**. When Return is pressed the

function returns the contents of **string\$**. Line 110 shows a simple call of this function.

For clarity, the coding for the append and delete functions has been put in a separate procedure in each case. Note too, the use of two further string functions. **ASC** returns the ASCII code of a character and is useful for dealing with non-printing characters like Return. **STRING\$** is a simple way of replicating a sequence of characters.

Try calling the function with different parameters. See also if you can modify the function to accept upper case letters only, or digits and decimal point only. More on characters next time.

```
10 REM Program InputFN  
20 REM Author Mike Williams  
100 MODE 3  
110 Z=FNinput(10,10,"Test Input",10,".")  
120 END  
130 :  
1000 DEF FNinput(x,y,prompt$,width%,paint$)  
1010 LOCAL string$,pos:string$="":pos=1  
1020 PRINTTAB(x,y)prompt$;" ":STRING$(width%,paint$)  
1030 PRINTTAB(x+LEN(prompt$)+2,y);  
1040 REPEAT  
1050 char$=GET$  
1060 IF ASC(char$)=127 AND pos>1 THEN PROCdelete ELSE IF pos>width% THEN VDU7 ELSE IF ASC(char$)>=32 AND ASC(char$)<127 THEN PROCappend  
1070 UNTIL ASC(char$)=13  
1080 =string$  
1090 :  
1100 DEF PROCappend  
1110 string$=string$+char$  
1120 pos=pos+1  
1130 PRINT char$;  
1140 ENDPROC  
1150 :  
1160 DEF PROCdelete  
1170 VDU8,ASC(paint$),8  
1180 string$=LEFT$(string$,LEN(string$)-1)  
1190 pos=pos-1  
1200 ENDPROC
```

Answer to problem:

```
120 PRINT LEFT$(string$,I-1);SPC1;  
MID$(string$,I+1)
```

B

ACORN IN ACTION

A totally new experience for show visitors

at the
**Royal Horticultural Hall,
Westminster, London SW1**



**10am-6pm Friday May 13
10am-6pm Saturday May 14
10am-4pm Sunday May 15**

You'll find the very latest software and peripherals for the complete Acorn range at the Electron & BBC Micro User Show.

But this time there'll be so much more to enjoy.

Acorn In Action will demonstrate some of the truly amazing projects currently involving the machines...

SEE ★ A spectacular laser light show controlled by a BBC Micro. (Saturday only)

★ The research work on the BBC Micro that has helped to bring new hope to sufferers of the eye disease glaucoma. (Friday and Sunday)

★ A program developed by an amateur astronomer to locate distant galaxies. (Saturday and Sunday)

★ The Beeb system being used by doctors at Guy's Hospital to provide a breakthrough in the treatment of arterial disease. (Saturday and Sunday)

PLUS watch your own heartbeats displayed, measure your manual dexterity and hear your own voice backwards – all courtesy of a BBC Micro.

Take your seat in the Archimedes Demonstration Theatre run by Acorn's own experts. Thirty minute special introductory courses to the new machine will be held on the hour, every hour throughout the three days. Price just £1.

It all adds up to a fantastic day out for the whole family!

Avoid the queues! Get your ticket in advance – and SAVE £1 A HEAD!

Post to: Electron & BBC Micro User Show Tickets,
Europa House, Adlington Park, Adlington,
Macclesfield SK10 4NP.



**Royal Horticultural Hall
Westminster
London SW1
May 13-15, 1988**

*Advance ticket orders must
be received by Wednesday,
May 4, 1988.*

Admission at door: £3 (adults), £2 (under 16s)

Please supply:

— Adult tickets at £2 (save £1) £ _____
(Order four adult tickets,
get the fifth FREE!)

— Under 16s tickets at £1 (save £1) £ _____
(Order four under-16s tickets,
get the fifth FREE!)

Total £ _____

Name _____ Signed _____

Address _____

Postcode _____
A355

☐ I enclose a cheque made payable to Database Exhibitions
☐ Please debit my Access/Visa
card no: _____

Expiry date: / /

PHONE ORDERS: Ring Show Hotline: 0625 879920
PRESTEL ORDERS: Key *89 then 614568383
MICROLINK ORDERS: Mailbox 72:MAG001

*Please quote credit card number
and full address*

Personal Ads

Genie Junior £20. BEEBUG Masterfile II £12. Studio 8 £10. Teletext pack 40T £5. Design 40T £8. AMX SuperArt (Master) £20. AMX MAX £10. CC's Printmaster (Star) ROM £10. Wigmore House Artist ROM £10. the Hobbit 40/80T £5. CJE's MFNLQ ROM, standard disc and font disc 'B' £22. All originals with manuals. Also HCR External ROM box for Master £50. Tel. (0527) 32613.

ViewStore manual, new, £4 post paid. D.Langton, 13 Whitmore Close, New Southgate, London N11 1PB.

BBC Master 512, Zenith ZVM-1240 high res. amber monitor, dual DS 40/80T disc drives, a full set of manuals, several utility ROMs and cartridges. Includes DOS Plus O.S. with GEM Write, GEM paint and GEM Desktop. Package price £760. Tel. (0334) 77617 eves. or 76161 ext 7164 day.

BBC B issue 7, Cumana 100k drive, Wordwise, Toolkit, Communicator ROMs plus software, games, books, £300. Tel. 01-669-5087.

WANTED Epson LQ 2500 printer in good condition with sheet feeder and colour option. Tel. (0745) 825036 eves.

ADI £18, Printmaster £15 new, Spellmaster £35 new, Master 128 SuperArt + MK3 mouse £45 new, Ref. manuals 1 and 2 £20 new, ISO Pascal £35 new. Tel. (0295) 65262.

BEEBUG Vols. 3 & 4, 10 cassettes (£5 per volume), £7.50 the lot. Tel. (0920) 5406.

Master 512 owner contact required for INFORMATION EXCHANGE on suitable DOS + software. Tel. (0846) 682187 eves.

Time Warp, Murom, Office Master and Mate. Offers accepted. Tel. (0923) 224548.

BEEBUG back issues from Vol.1 up to date. All perfect and complete, £15. Tel. (0602) 231395.

Master 128, £275. 512 co-processor with GEM software and mouse, £120. Twin DS DD 40/80T disc drives £100. Taxan 620 hi-res colour monitor £200. All in good condition and with manuals. Plus software, magazines, joysticks, data recorder - call for prices. Tel. (06282) 5062 eves.

Original ROM software and manuals: Spellmaster £36, ADI £18, Acornsoft Pascal £35, AMX MK3 mouse & SuperArt (M128) £45, ViewPlot £15. Tel. (0295) 65262.

Acorn Electron computer with Plus One interface, inc books (various User guides, 30h Basic, Start programming with the Electron etc.), power supply and dust cover, £85 the lot. Acorn 6502 2nd processor + Hi-Basic ROM £100. BBC B User and Advanced User Guides, £15 the two. Forth ROM + User Guide £25, BBC B dust cover £3. Tel. (0909) 565257.

Watford NLQ ROM unused, £20. Tel. (0865) 66426, eves.

BBC B series 7, external ROM/RAM (32k) expansion board in case with separate PSU. Modem with comms software. Over £1000 worth of cassette, disc and ROM software, inc View, Databases, Elite, Trivial Pursuit. Joystick, leads etc. £600 ono. Tel. (0483) 38285 eves. Master 128, Acorn dual disc drives (400k) in plinth, Zenith amber monitor. InterWord, InterSheet, Le Modem, all as new. £550. Tel. (07072) 63617.

WANTED Microwriter complete. Tel. (0752) 701261 daytime, (0822) 852867 eves.

Twin 360k 5.25" full height DS disc drives, boxed, new/unused. Power and data cables for BBC computer. £69.00 VAT included. Tel. (0734) 479550.

200 cps wide carriage Dot Matrix Printers Serial/Parallel interface. Friction and Tractor feed. Comprehensive User Manual. New, unused £99.95 VAT included. Tel. (0734) 479550.

Master Compact with second drive, Microvitec 1451 colour monitor, RS232 joystick Mertec companion upgrade, Viglen ROM cartridge adaptor, printer cable. As new, worth £900. Offers £550. Tel. Oxford 739766.

Spellcheck III ROM package, mint condition, boxed £15. Tel. 01-654-7609.

BBC B, needs attention but works, £150. Opus twin 40/80T drive £100. Cub 1431 colour monitor £100. Printmaster £10, Master ROM £20, Digimouse £20. Ref. manuals 1&2 £10. 2 Cartridges £5, Acorn User barcode reader £20. Drawer CAD £20, Program Builder £5, Discmaster £5. Tel. (0205) 68276 after 6

Centronics microprinter P1 with a spare roll of paper, £20. Acorn NFS 3.6, £5. Tel. (0705) 594845.

Pace Nightingale system for BBC with Autodial, Comstar, OBBS Bulletin Board, complete with software, cables and manuals. Practically unused, £90 ono. Tel. Exeter (0392) 34149.

Epson printer FX80, working order. Serial and parallel I/Fs £160. Tel. 01-908 4974.

Compact RS 232 kit £15, Fleetstreet Editor BBC B 40T, £20. JUKI 2200 daisywheel typewriter with Centronics interface £100. Tel. (0635) 62758.

Vine Micros Replay for B+ £15. DiscDoctor £12 inc. p+p. Tel. (0274) 875217.

BBC Master, Turbo coprocessor, dual 40/80 DS disc drive fitted in Viglen console, Microvitec 1451 RGB Monitor, Kaga KP810 printer, AMX mouse. Plus manuals, software, £1000. For details/offers tel. (0256) 464889.

Selfpowered HCR external ROM/RAM system (for 16 ROMs/RAM) £40. Software at about half price: Help II, ROMIT, Toolkit+, Sleuth (ROMs); DiscMaster Printwise, Program Builder, Hershey Characters (80T discs), all from BEEBUG. Wigmore House Artist (with mouse) ROM and disc. Bar Code Reader Project, Disc Duplicator, Signwriter, Fontaid, Tabler Database, Mini Office II, Watford Printer Dumpout III, Spellmaster (ROMs), light pen. Tel. (027583) 2979.

BBC B, Watford DFS, Basic 2, Wordwise ROM, Seikosha GP100 printer, tape recorder, joystick, maths tutor etc, £250. Tel. 01-907-2958.

View, ViewStore, Printer driver generator, DiscDoctor, 80 column printer stand. All boxed with manuals £75 the lot. Tel. 01-560-7531.

ViewStore complete with ROM disc + manual £25. Tel. (0903) 755412.

Continued on page 36

BEEBUG MAGAZINE OFFERS

BEEBUG FILER

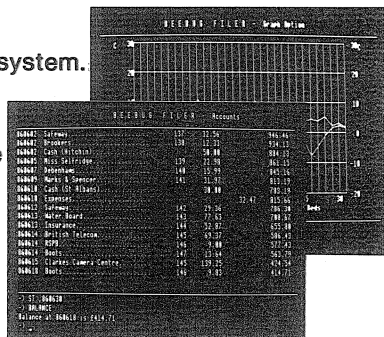
BEEBUG magazine's own database management system.

Suite of five programs including:

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

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

Price £5.50 plus £1.00 post and packing.



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

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

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

Price £8.95 (kit) £12.95 (ready made)

Post & packing £1.00 in each case

BEEBUG Master Alarm ROM — see Vol.6 No.3

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

Supplied on EPROM

Price £6.45 plus £1.00 post & packing

ORDER FORM

Name

Address

Membership Number

Signature

I enclose cheque for £

Please debit my Access/Visa No.

Please send: Price

Filer Disc £

BEEBUG Sideways RAM Kit £

BEEBUG Sideways RAM Module £

BEEBUG Master Alarm ROM £

Post & Packing £

Total £

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

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

Personal Ads (continued from page 34)

Over 40 tape and disc games for the BBC B from £1, including Scrabble, Monopoly + Trivial Pursuit. All originals with full documentation. List from Mr J.Pinner, 10 Moor Park, Millom, Cumbria LA18 5DX.

For BBC B: Aries B32 + B12 £60. PMS 6502 2nd proc. £50. Acorn 6502 2nd proc. £40. Watford ROM/RAM board £15 (slight damage). Micro User clock £5. Solidisc DFDC (issue 1) + DFS + ADFS ROMs £20. Viglen console £7. Various ROMs at £10 each inc. Acorn DNFS/ADFS/View, BROM, Caretaker. All with original manuals. Many C10/C15 tapes, 10 for £1. Tel. (0332) 360023 after 6 pm.

APT L Sidewise ROM/RAM board £18. DiscDoctor ROM £8. Tel. (0482) 811227.

Cumana twin DS 40/80 switchable drives with own PSU £150. Acorn Z80 second processor with all original software, Cobol, Nucleus, WP etc, £150. HCR ROM/RAM box for 24 ROMs and 32k RAM, own PSU, no soldering, £60. Prism 1000 Modem + Micronet software £30. Open Logo on ROMs £30. Tel. (04024) 74633.

Commstar II Communications ROM, Hayes Compatible (Ideal for Pace Linnet Modem), £10. Tel. (09905) 7689.

Taxan Kaga Supervision II 620 monitor, leads, instructions, IBM/BBC compatible, VGC, boxed, £190. Akter 40/80T DS disc drive, VGC, leads, instructions, £75 ono. Masterfile II DFS, boxed, instructions, £6. ViewStore Version 01 plus manual, keystrip and reference card, £25. Tel. 01-341-2187.

Philips 12" monitor, choice of BM7502 green or BM 7522 amber, as new £40. Tel. 01-836-5454 ext. 2439 or 01-348-5957.

Acorn second processor, software, little usage, £150. View/ViewSheet guides £14 the pair. Cumana DSDD 40/80T disc drive £200. Master guide Pt.1 £10. 50 DS 5.25" discs £35. SAE with offers please. 7 Low Rd. Port Logan. Tel. 077686260.

Master 128k + Interword ROM as new. Cumana 40T SS disc drive with power supply. Brother HR5 Printer, joystick, leads, speech, games & blank discs. Reasonable offers please. Tel. (021353) 6506 after 6.30 pm.

Aries B12 and B12c adaptor, boxed with instruction manual, £25. Tel. Hamilton (0698) 458137.

ISO Pascal £35, Micro Prolog £45 and Comal £30 Acornsoft language ROMs and documentation in original boxes. Acorn Graphics ROM £15. Minerva System Delta ROM, manual and Inter/View link £65. Tel. 031 667 4180 eves.

Master 128k with 512k DOS, mouse and GEM software. 80T twin DSDD drive. 12" Zenith monitor and all necessary cables. Comprehensive set of manuals inc. Ref.Manual 1&2, View, ViewSheet and Advanced User Guide. Command, Exmon and Dumpmaster ROMs. More software (both BBC and MSDOS). £700. Tel. (0509) 612960 day, (0509) 880183 eves.

BBC Master 128 as new, only £335. Master Reference books, Part 1 and 2, £10 each. Tel. Bedford (0234) 67067 eves.

Acorn 6502 second processor £90, Watford solderless ROM/RAM board with 16k RAM and battery backup £25. BEEBUG ROMit ROM £12, Chalice ROMmaster ROM £7. Tel. Malvern (06845) 64106.

Viglen 40T disc drive £30. Spellcheck II £8 and Aviator (disc) £6. Tel. Windsor (0753) 853193.

Acorn Prestel adaptor £50, Cumana dual 80/40T switchable disc drive, own PSU £80. ACP advanced 1770 DFS £20. Wordstar Professional for BBC Z80 £85. Tel. (0533) 312661.

Tractor feed for Brother HR-15 printer. Tel. (04022) 50500 daytime.

BBC model B (issue 7), Watford ROM/RAM board 64k sideways RAM, Watford 1.44 DFS, Acorn 6502 second processor, Cumana twin 40/80T drive, Microvitec high-res colour monitor (1441), cassette drive, mouse, joystick, Wordwise+, Pagemaker, manuals, software, books, all BEEBUG mags to date, £700. Tel. (04022) 20305.

Oxford Pascal (v2.1), original ROM + manual £25. Tel. 061-439 9665.

Software, all original. Disc (model B): Graphic Adventure Creator £12, Word Perfect W.P. £5, Strike Force Harrier £6.50. ROM: Wordwise Plus with manuals and typing tutor £30. More programs on tape for B and Master. Postage included. Tel. (0388) 767040.

Daisy Step 200 high quality daisy wheel printer, good condition, 13" platen, c/w power and BEEB cables, instruction book. Offers. Tel. Ludlow (0584) 2596 eves.

Wordwise ROM with cassette and manual. Offers. Tel. (0395) 263638.

BBC B+ 64k with Acorn 1770 DFS, AMX, SuperArt ROM + mouse and lots of software. Cost £800. Will sell for £275 ono. Tel. Stretham (035389) 273.

BEEBUGSOFT: PrintWise on disc £16, Hershey Characters disc £7, Sleuth ROM £16. AMX Paint Pot disc £7. FloppyWise ROM £15. All with manuals. Tel. (0925) 811420 eves.

Miracle Technology modem WS2000 with DataBeeb communications ROM and RS423 lead, £40. Tel. (0473) 213907 after 6pm.

BBC B 1.2 OS, Acorn DFS inc. tape recorder + games (tapes and discs), all manuals £200. Tel. Lincoln (0673) 61163 eves.

BBC BD, View 3.2 and Watford 32k shadow RAM. Dual DS 40/80T drive in plinth. Kaga KX12 hi-res green screen monitor. All excellent condition. £400 ono. Tel. 01-349-9381.

Master Compact, TV adaptor and printer cable, boxed as new, some software and books, £260 ono. Tel. Lancaster (0524) 859228.

ROMs boxed with handbooks etc: Oxford Pascal £121, DiscDoctor £8, System ADE £20. Tandon 40T 100k disc drive with handbook, utils. disc and leads £35. Advanced User manual £5, Toolkit 2 tape and manual £5. Immaculate condition. Tel. (Nottingham) (0602) 392554.

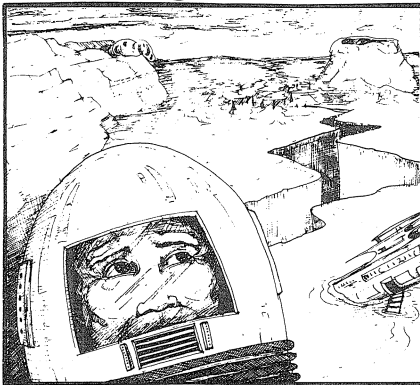
LVL Music keyboard with EchoSoft OrganMaster Software. Very good condition, plugs into user port on the BBC micro, only £30. Tel. Colwall (0684) 40220.

Epson RX 80F/T printer with BBC lead, Printmaster ROM and NLQ ROM £160. Tel. (0934) 513135 eves. or MBX Prestel 934513135.

Epson MX80 FT3 dot matrix printer with manual, stand, interface lead for BBC B/Master, and PMS Multi-font NTQ ROM set and manual, £120. Tel Gravesend (0474) 363503.

AMX Pagemaker 2x16k ROMs, 2 discs, not used, £25. Tel. 03406468.

Nostalgia



Remember "Countdown to Doom"? Peter Kilworth's classic sci-fi adventure may well have been the first game you ever bought. Well, RETURN TO DOOM - Part 2 of his Doom trilogy- is now available from Topologika:

You are flying through the universe, minding your own business, when a distress call comes in. "Mayday! The Galapoxi, taking the Ambassador of Regina on an important mission to Flaxo, has just crashed on Doom. Rescue needed! Heading for Cleft..." As the only person ever to have survived Doom, you steer once more for that dangerous planet. This could be your finest hour.

Or maybe even longer.....

As tough as Countdown - but a different sort of challenge - RETURN TO DOOM is sure to bring back memories. £12.95 inc disc, manual,VAT and P&P.



Tel: (24 hours ACCESS) 0733 244682

FREEPOST PO Box 39, Stilton
PETERBOROUGH

ADVERTISING IN BEEBUG

For advertising details, please
contact
Yolanda Turuelo
on
(0727) 40303
or write to
Dolphin Place,
Holywell Hill,
St Albans,
Herts.AL1 1EX

SPRITE EDITOR

This keystrip is designed to be used with the Sprite Editor from BEEBUG Vol.6 No.8. Either cut out the keystrip or photo-copy this page and place under the plastic strip on your micro when using this program. Note that Ctrl-f9 should be labelled QUIT as shown, and not STAR as given last month.

Key f0	Key f1	Key f2	Key f3	Key f4	Key f5	Key f6	Key f7	Key f8	Key f9
LOAD		SAVE		FLIP X		FLIP Y		CLEAR SCREEN	QUIT
Black/White	Red/Cyan	Green/Magenta	Yellow/Blue	Blue/Yellow	Magenta/Green	Cyan/Red	White/Black		
Black	White	Yellow	White						
Black	Red	Green	Yellow	Blue	Magenta	Cyan	White		
Black	Red								

BBC USER GROUP INDEX

Over the next few issues we intend publishing a List of BBC User Groups operating in the UK and Overseas. This list has been updated recently, however we would appreciate any information about changes or cessation of operation. If you are aware of any User Groups not listed here, please let us know for the benefit of other members of BEEBUG.

BEDFORDSHIRE

LUTON

Meets on Mondays at Strathmore Av. Methodist CHURCH HALL from 6.30pm and on Wednesdays at LUTON COLLEGE room 212 from 6.30pm.

BIRMINGHAM

Meets on the second Tuesday of each month 7.30pm at the Headquarters of the Midland Radio Society, Unit 5 Henstead House, Henstead Street, Birmingham 5. Tel. Michael Nyman 021-382-3606

BRISTOL, Yate & Sodbury Computer Club

Meets on alternate Mondays in canteen at NEWMANS, Station Rd, Yate. Contact Kay Crowe Chipping Sodbury 317461

BRISTOL

C.A.Hynam, 23 Baugh Gardens Downend, Bristol. Tel. (0272) 561237 [preferably after 4 p.m.]

BRISTOL (SOUTH) and District

B.Boyde-Shaw, BYTE, 7 Riverway, Nailsea, S.Bristol, Avon BS19 1HZ. Tel: (0272) 851337

BUCKINGHAMSHIRE

Aylesbury Computer Club,

Chris Pyves, 2 Bishop's Walk, Aylesbury, Bucks HP21 7LF. Tel. (0296) 26347

BUCKINGHAMSHIRE

South Bucks Acorn Computer Club

Contact Colin Mills, 70 Chestnut Lane, Amersham HP6 6EH. Tel. Amersham 6103. Meeting 8pm-10pm on 1st Tuesday of the month at St.Leonards Church Hall, Chesham Bois.

BUCKINGHAMSHIRE

John Haig, 141 Leas Drive, Iver, Bucks. Phone Iver 654431. Meets bi-monthly at the 'Huntsmoor Room', Village Hall from 7.30 until 10pm on the 2nd and 4th Thursdays of each month.

CAMBRIDGE

Bottisham Acorn User Group

Contact Peter Rank (Cambridge 812 080) or Gerald Wilcockson (Saffron Walden 23793).

CHESHIRE

Mid-Cheshire Computer Club

Hold twice monthly meetings (2nd and 4th Fridays) between September and March, and 1 meeting per month (2nd Friday) in summer. Meetings held in Winsford's main library and start at 7.30pm. For further details, call Winsford 53339.

MID-CORNWALL

Mid-Cornwall College of F.E., Palace Road, St Austell. Tel. Par 2399 any time.

CUMBRIA

The West Cumbria User Group

Details: P.Majid (0946) 62732 or K.Purkiss (0946) 66586

DERBYSHIRE

GLOSSOP Computer Group

T.S.Fox (secretary), 4 Park Lane, Little Hayfield, Stockport, Cheshire SK12 5NW. Tel. (0663) 44260.

ESSEX

Chelmsford BBC Micro User Group

Ian Jefferies, 2 Gernon Close, Broomfield, Chelmsford, Essex. Tel. 0245 440054

ESSEX

Anyone interested in starting a Beeb User Group in the **ILFORD** area? Please contact P.Jones at: 1 de Vere Gardens, Granbrook, Ilford, Essex. IG1 3EB. or telephone 01-554-9825. The club will use the facilities of the Computer department at Forest School, Snaresbrook.

NORTH and MID-ESSEX BBC Users Group

Witham Library, Newland Street, Witham, Essex. Meets 2nd Thursday and 4th Wednesday of each month. For details contact either A.Purkiss (0376) 515609 or D.Watts (0245) 358127.

GLOUCESTER

Laurie Dann, 10 Highbank Park, Gloucester GL2 9DY. Tel. Gloucester 21245.

HERTFORDSHIRE

HARPENDEN - General User group

contact R.S. Welch on 05827 3398.

HANTS

FAREHAM & PORTSMOUTH

Peter Smith, 5 Barnfield Road, Sheet Petersfield, Hants. Tel. Petersfield 64059 evenings only
Group meets at 7pm on Tuesdays at the
Portchester Community Centre.

INVERURIE (NORTH EAST SCOTLAND)

Grampian Amateur Computer Society
Meets on Monday evenings. Produces bi-monthly newsletter. Contact Paul Cuthbertson on (0467) 24030, 18 Morningside Crescent, Blackhall, Inverurie AB5 9FA, or Bruce Edelsten on (0224) 639911.

IPSWICH

Karl Brandenburg, 19, Oxford Road,
Ipswich IP4 1NL.

KENT

D.D.Singer, 86 Southborough Road, Bromley,
Kent BR1 2EN.

KENT

C. Rutter, Medway Acorn Users group,
St John Fisher School, Ordnance Street,
Chatham ME4 6SG. Tel. Medway 42811

LANCASHIRE

Mr.A.Edwards, 50, Station Rd., Banks, Southport
PR9 8BB.

LANCASHIRE

St. Helens BBC User Group. St Helens ITEC,
Windle Pilkington Centre, Waterloo Street, St.
Helens. Meets 2nd & 4th Thursday of each month.
For Further Details ring John 0744-817787

LIONCOLNSHIRE

BOSTON Acorn Computer Users Club
Contact J.C.Goodwin, BACUC, 245 Church Green
Road, Fishtoft, Boston, Lincolnshire PE21 0RP.
Telephone Tootsoft Enterprises on Boston 57409.

LIVERPOOL

Mersey BBC User Group (MBUG)

Meets twice monthly on first Wednesday of month
at OLD SWAN TECH. COLLEGE, Room C33 from
7.30pm and on third Thursday at BIRKENHEAD
TECH. COLLEGE, first floor Science & Maths dept
from 7.30pm. For more info call Nik Kelly on
051 525 2934]

LONDON - MIDDLESEX

Anyone wishing to start a user group please
contact V.K.Batta, 187 Waye Av., Cranford,
Hounslow, Midx. TW5 9SH

NORTH LONDON BBC Micro Users Group

Ernest Bebbington 82 Hornsey Lane,
Highgate, London N6 5LU.
Tel: 01-263 6760 (Evenings).

LONDON - WC1

Meetings held on the second Tuesday of each
month at Dept. of Psychology, University College,
26 Bedford Way, London WC1. Tel. 01-387-7050
ext 413 for details and directions.

LONDON (West)

West London Personal Computer Club. Contact
Graham on 01-997 8986, or Neil on 01-997 9437.

MERSEYSIDE

Fred Shaw, Merseyside Micro Group,
14 Albany Avenue, Ecclestone Park, Prescot,
Merseyside L34 2QW.

MIDLANDS (WEST)

Anyone interested in joining or running a
computer enthusiast club? Contact: Simon Doyle,
The Brandon Hall Hotel, Brandon, Nr.Coventry,
Warwickshire CV8 3FW.

MIDLANDS (WEST)

Dudley Area

Roger Luff, Tel. Kingswinford 288721

W.NORFOLK

BBC Micro User Group,
Norfolk college of Arts and Technology,
Tennyson Av, Kings Lynn.

NOTTINGHAM

BBC Micro club. Meets on second Monday of each
month. For details call R.Hampton on 254056, or
J.Day on 225660.

OXON

BBC Micro Wantage User Goup

Meets on the 2nd and 4th Mondays of each month,
8pm at the Wantage Civic Hall. Subscriptions:
adults 50p, under 18s 25p, first-timers free.
Contact Steve Cooper on Wantage (02357) 2501.

RUGBY

RASUG, Mr.Nicholas Shaw, 13 Pipewell Close,
Bilton, Rugby. Tel. 0788 811678

SCOTLAND (Central)

Anyone interested in joining a user group in this
area, please contact Mr.D.Davidson, 1 Roxburgh
Place, Larbert, Stirlingshire FK5 4UE.
Tel. (0324) 558692.

SHEFFIELD

ABUG . c/o J.Fryer, 17 Edgedale Rd, Sheffield S7
2BQ.

STAFFORDSHIRE

Contact Mrs. Linda Yeomans,
13 Regent St, Church Gresley,
Burton-Upon-Trent, Stafis DE11 9PL. Tel. (0283)
216445 after 2pm.

To be continued

Business Ads

CHESS EXPERT SYSTEM (Master/B/B+). A complete disc-based game storage and analysis system with extensive features and data links to Colossus 4 and White Knight 12. Supplied with 2000-move openings database, or optional 37000-move set. Send £1 for demonstration disc to: *Bernard Hill, Hawthorn Bank, Scott's Place, SELKIRK TD7 4DP.*

SHAREMATIC. The best software package for the keen investor, (runs on BBC Master 128). Free - automatic updating from Teletext. Free - 500 trading days of data for 170 shares. Plots, Graphs, Point & Figure, & Oscillator and more. Only £35. For brochure write to: *Citymagic Associates, Dept BB3, 40 Manor Road, Goldington, Bedford MK41 9LQ. Tel. (0234) 856050/67067.*

EVENTS

The Electron & BBC Micro User Show
New Horticultural Hall
Westminster, London
13-15th May 1988

The Electron & BBC Micro User Show
New Horticultural Hall
Westminster, London
11-13th November 1988

PCW 88 Show
11th Personal Computer World Show
Earls Court, London
14th-18th September 1988

Both BEEBUG and RISC User will have their own stands at the Micro User show in May. You are very welcome to visit our stands and talk with our magazine, sales and technical staff.

SOME MORE BULLETIN BOARDS

IBBS Nottingham

Sat 10am-4pm
Sun 10am-4pm

0602 830231
(Nottingham)
1200/75
300/300 all ASCII

DCT Database
24 Hours

0384 239944
(Dudley)
Prestel compatible

Phantom Viewdata
24 Hours

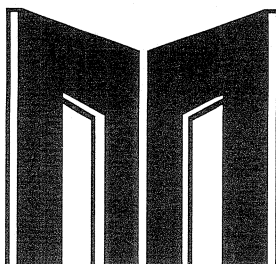
0226 732140
(South Yorkshire)
1200/75

Surrey BB
24 Hours

08832 5932
(Surrey)
200/75 300/300
Viewdata/Scrolling

Enterprise BB
10pm-8am

0482 868388
(North Humberside)
1200/75, Terminal/
Viewdata

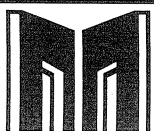


THE MASTER PAGES

Devoted to the Master
Series Computers

In our Master pages this month, Lee Calcraft shows how the ADFS routines presented in the previous issue may be combined into two complete utility programs. One is a customised ADFS menu system, the other a useful 'CATALL' facility.

Dave Somers has been looking at the Master Extensions ROM from DABS Press, and we complete this section with further hints and tips specific to the Master and Compact.



MASTER
SERIES
TALKING TO
THE ADFS
(part 2)

Lee Calcraft concludes his experiments on the ADFS with the implementation of two utilities: a customised ADFS menu and a whole-disc catalogue.

Last month we explored the use of certain filing system calls to create three packaged procedures which could be used to read catalogue and other allied information from an ADFS disc. Here we will put those procedures to work in the implementation of two utilities. Both require lines 1000 onwards of last month's code, though none

of the lines before 1000 are needed. I suggest therefore that you load in last month's program, if you have it, and delete the lines lower than 1000, and save the remainder away for future use.

CUSTOMISED ADFS MENU

With lines 1000 onwards in your machine, type in this month's listing 1. This will implement a complete ADFS menu. When the program is run you will see something similar to that in the accompanying figure. Near the top of the screen are displayed the current drive number, and directory name and title (all obtained by calling PROCtitle). Then below that in alphabetical order is a display of objects in the current directory. Each has a letter beside it, and all directories are marked in reversed out text. Pressing a letter corresponding to a directory name causes that directory to be selected, and the menu to be re-displayed. If you press a key corresponding to a Basic program, it will be chained in. At present there are no options for loading text files into word processors, but this feature is easily added (see later).

Customised Menu Keys

^	Go to root directory
<	Go to the previous directory
v	Go up one directory
0	Select drive 0
1	Select drive 1
*	Perform star command
Ctrl @	Quit

Other options currently implemented include star commands, and the use of cursor keys to take you to the root directory or to move up one directory, or to the last selected directory. To change drives,

or change a disc, select the required drive with the "1" or "0" keys. Finally, Ctrl-@ may be used to quit the menu.

PROGRAM NOTES

The menu is controlled from the short REPEAT loop in lines 130 to 170, which calls 3 procedures. PROCtoppart handles the top few lines of the menu, including the title, drive and current directory information. This is obtained by calling PROCtitle. PROCtoppart also calls PROCfile, the second of last month's procedures, reading in the names of the objects in the current

```
VIDEO
Wineglass
SP200K4
JFM
WALLER
Palmco
WALLER
Authors
Backcover
B33000-014
Cocotail
Contente
Cover
WALLER
Editorial
JULY68
WALLER
LETTER1
LETTER2
LETTER3
LETTER4
WALLER
WALLER
WALLER
WALLER
```

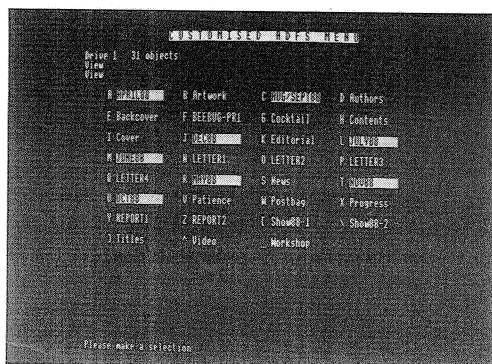
IF (file%(key-64,1) AND &FFFF)=&8000
and load it into sideways RAM, or that its
execution address is &FFFF using

A *CATALL FACILITY

reversed out text, and each new level of nesting causes the display to be indented by four character positions. If you have a disc which nests below about 15 levels, the display will wrap around, but if this is a problem, you can change the indenting factor (the second 4 in line 410 could be reduced to 2, 1 or even 0).

As it continues its display of objects, it will eventually reach the end of the current directory (i.e. the directory which it is currently displaying). When this occurs, the procedure executes `*DIR ^` to move up one level, and then terminates (i.e. with `ENDPROC`), thus dropping the program back into the procedure which called it (i.e. `PROCdispal` at one level above). The display then continues from where it left off until it reaches a new directory, in which case it makes another recursive call, or it reaches the end of the current directory, where it goes back one level. Eventually, the program "surfaces", and the display is complete.

42



Listing 1

```

10 REM Program ADFS Menu
20 REM Version B 1.0F
30 REM Author Lee Calcraft
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 REM Program requires procs from
70 REM last month (lines 1000-3300)
80 :
100 ON ERROR GOTO 760
110 MODE0:DIM file$(50),file%(50,4)
120 *FX 4,1
130 REPEAT
140 PROCToppart
150 IF fileno>0 THEN PROCreadcat
160 PROCchoice
170 UNTIL key=0
180 *FX4
190 END
200 :
210 DEFPROCToppart
220 CLS:COLOUR0:COLOUR129
230 PRINTTAB(18,0)"C U S T O M I S E D
  A D F S M E N U"
240 COLOUR1:COLOUR128
250 PROCTitle
260 PROCfile
270 PRINT"Drive ";drive;SPC3;fileno"
objects"
280 PRINTtitle$
290 PRINTname$'
300 ENDPROC
310 :
320 DEFPROCreadcat
330 PROCcatinfo(TRUE)
340 FOR A=1 TO fileno STEP 4
350 FOR B=0 TO 3
360 AB=A+B-1
370 IF AB<fileno PRINTTAB(5+16*B);CHR$(
  (AB+65)SPC1;
380 IF file%(A+B,0)=2 THEN COLOUR0:COL
  OUR129
390 IF AB<fileno PRINTfile$(A+B);
400 COLOUR7:COLOUR128
410 NEXT:PRINT':NEXT

```

```

420 ENDPROC
430 :
440 DEFPROCchoice
450 IF fileno>0 PRINT TAB(0,31)"Please
  make a selection "; ELSE PRINT"No files
  ";
460 REPEAT
470 T=FALSE:*FX15
480 key=GET
490 IF key-64>0 AND key-64<=fileno THE
  N PROCfiledir:T=TRUE
500 IF key=42 THEN PROCstar:T=TRUE
510 IF key=48 THEN OSCLI("DIR :0"):T=T
  RUE
520 IF key=49 THEN OSCLI("DIR :1"):T=T
  RUE
530 IF key=139 THEN OSCLI("DIR $"):T=T
  RUE
540 IF key=136 THEN OSCLI("BACK"):T=TR
  UE
550 IF key=138 THEN OSCLI("DIR ^"):T=T
  RUE
560 IF key=0 AND INKEY-2 THEN T=TRUE
570 UNTIL T
580 ENDPROC
590 :
600 DEFPROCfiledir
610 *FX4
620 IF file%(key-64,0)=2 THEN OSCLI("D
  IR "+file$(key-64))
630 IF (file%(key-64,2) AND &FF00)=&80
  00 THEN CHAIN file$(key-64)
640 REM Add further file actions here
650 *FX4,1
660 ENDPROC
670 :
680 DEFPROCstar
690 INPUT""command$
700 CLS
710 OSCLI(command$)
720 PRINT"Press space ";
730 REPEAT UNTIL GET=32
740 ENDPROC
750 :
760 REPORT:PRINT" at line ";ERL
770 *FX4
780 END
790 :
  * * * * *

```

Listing 2

```

10 REM Program Display all files
20 REM Version B 0.9
30 REM Author Lee Calcraft
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 REM Program requires procs from
70 REM last month (lines 1000-3300)
80 :
100 ON ERROR GOTO 490
110 MODE3
120 DIM file$(50),file%(50,1),ends%(12
  8)

```

Continued on page 46



Dave Somers takes a look at MOS Plus from DABS Press - a utility ROM to enhance (and correct) the Master's Operating System.

Although the Operating System (OS) on the Master microcomputer is excellent, it does have a number of irritating bugs and omissions.

This is where MOS+ from DABS Press comes in, for it fixes these bugs in OS version 3.10, and adds some very useful additional commands.

Product	MOS Plus
Supplier	DABS Press 76 Gardner Road, Prestwich, Manchester M25 7HU. Tel. 061-773-2413.
Price	£12.95 inc VAT (ROM) £7.95 inc VAT (5.25" disc) £9.95 inc VAT (3.5" disc)

It comes in the form of a sideways ROM, and can be supplied either as a ROM, or on disc for loading into sideways RAM. An eleven page A5 instruction manual details its various facilities. After installing MOS+, the sign-on message produced by the computer changes from "Acorn MOS" to "Acorn MOS+" to let you know that everything is OK.

ENHANCED STAR COMMANDS

A number of enhancements have been made to some of the star commands, with quite a few of them as found on the Master Compact. Both *APPEND and *BUILD will now accept all 256 ASCII characters, rather than the first 128 as before. When loading data into sideways RAM, using either SRLOAD or SRWRITE, an optional "I" parameter can be added to initialise the data. For SRLOAD, this removes the need to press Ctrl-Break before an image is recognised by the Operating System. *UNPLUG can also have this parameter added, in which case the ROM is unplugged immediately, by removing its entry from the ROM information table.

*SHOW has been tidied up so that omitting a key number will cause all the key definitions to be displayed. For *STATUS, the entries are now displayed in alphabetical order, and *ROMS lists Sideways RAM as "RAM", and doesn't duplicate entries.

To backup discs, you don't have to search for the Welcome Disc, as MOS Plus comes complete with the necessary code. *BACKUP automatically handles both DFS or ADFS formats, and if required can use sideways RAM to speed up the process. You can also use it to backup (on DFS discs) from side :0 to :2 (or :1 to :3) on separate discs - it automatically prompts you to change discs.

The *BASIC @ command has been extended so that the tokeniser will accept text from any location, and not just from the EDIT's buffer.

NEW STAR COMMANDS

Nine new star commands have been added. For ADFS users, *DRIVE has been added to allow DFS software using this command to still work under the ADFS; *CATALL will catalogue all the files on the disc; *EXALL is similar to *CATALL, but performs *EX on all files; *FIND will search for a specified file (with wild cards allowed) and display their path if found. A formatter and verifier are also included.

A command called *SETTIME can be used to set the date or time or both of them. The syntax is identical to Basic's TIME\$ command. This will allow the RTC to be modified within other languages that do not support it, eg Comal. To complement the sideways RAM handling commands, *SRKILL will clear out the first 16 bytes of one or more RAM banks.

ALARM CLOCK SYSTEM

Part of the 146818 Real Time Clock contains an Alarm Clock facility, which is unfortunately not implemented by the Operating System. MOS Plus allows you to use this, provided you make a link on the main circuit board.

*ALARM is used to control the system. If followed by no parameters, the current setting will be displayed. The alarm can be configured

HELP*MOS+ 1.13**

MOSPlus - For list of new commands
 BASIC - For list of BASIC keywords
 View - For help on View commands
 Alarm - For help on using the Alarm clock
 Terminal - For help on Terminal commands

***HELP BASIC**

BASIC keywords listed by token

***HELP VIEW <topic>**

for help on view family
 Topics available are :

Commands
 Stored
 Functions
 Highlights
 Sheet
 Spell

HELP MOSPlus*MOS+ 1.13**

BACKUP <src> <dest> (S) (P) (Y)
 BLIST <ob spec> (<format>)
 CATALL (<drive>)
 EXALL (<drive>)
 FIND <list spec> (D) (F)
 FORMAT <drive> <S,M,L> (Y) (N)
 SETTIME <date/time>
 SRKILL <bank id>...
 VERIFY (<drive>)

***HELP TERMINAL**

Terminal commands: (Use CTRL-F1 to enter command mode)
 MODE <number> - Change mode
 VDU <no.> [,<no.>] - VDU sequence
 TERMINAL - Select ANSI mode
 BBC - Select raw mode
 GS - Select GS format mode
 TTY - Select dumb terminal mode
 AWM ON/OFF - Control line wrap
 CKL ON/OFF - Control cursor keys
 MCL ON/OFF - Enable/Disable Modem control
 PROT ON/OFF - Control protection
 RFC ON/OFF - Enable/Disable receive flow control
 TFC ON/OFF - Enable/Disable transmit flow control
 WWM ON/OFF - Control word wrap

***HELP ALARM**

MOS+ Alarm Clock 1.10
 ALARM
 ALARM <hh:mm:ss>
 ALARM <H,M,S>
 ALARM ON
 ALARM OFF
 ALARM QUIET
 ALARM LOUD
 CONFIGURE ALARM QUIET
 >CONFIGURE ALARM LOUD

*Examples of *HELP Displays*

to go off every second, minute, hour, or at a specific time. When the alarm is triggered, a beeping noise is made. Its volume can be adjusted to either low or high, and can be set permanently in the CMOS RAM, if required. For machine code users, the alarm can be trapped by intercepting user event number 9.

MOUSE DRIVER

To avoid having to install a ROM such as AMX, etc., MOS Plus comes with the mouse driving code necessary, along with the OSWORD &64 call to access it.

BUG FIXES

With OS 3.10 there were a couple of bugs which could occasionally cause problems. The infamous CLOSE#0 bug, where the DFS does not close files with the correct length in the event of *CLOSE or CLOSE#0, has now been fixed; after *EDIT there can be any number of

spaces before the filename; if more than one filename is specified after *REMOVE, the call will correctly fail.

When re-setting the CMOS RAM with R/power-on-reset, the settings now default to sensible values.

HELP SYSTEM

A series of help texts can be obtained about MOS Plus itself, the Alarm facility, Basic, View, and Terminal. This is an excellent aide-memoire.

FINAL THOUGHTS

MOS Plus is an excellent product. It offers all those facilities which you are likely to use without having to resort to the Welcome Disc, and a few others that are most useful. It has been well thought out and is a most welcome addition inside any Master. B

Hints Hints Hints

SIMPLE AUTO-SAVE

Lee Calcraft

The following EXEC file will automatically save a Basic program under the name stored in a REM line within the program itself. The format is similar to that used on the Archimedes and in the Master ROM. The EXEC file should be created using a word processor or the Master's text editor, and could be saved under the name "S". Once it is installed in your Library directory, every time you issue the command *S your currently resident Basic program will be saved with the name given. The format for the filename is:

```
10 REM >filename
```

The name must immediately follow the ">" character, and must itself be at the end of a

*| Automatic Program Save

```
VDU21
P%=PAGE
REPEAT P%=P%+1:C%=?P%:UNTIL C%=ASC(">") OR
P%>PAGE+100
IF C%<>ASC(">")VDU6:P.TAB(10)*** File name
not found ***:VDU7,21:OS.("FX125"):VDU6
*FX15,1
S$=$(P%+1)
VDU6:P.TAB(3)"SAVING ";S$
SAVE S$
```

program line. The only other proviso is that the ">" character must fall within the first 100 bytes of the program.

The routine displays the name of the file which it is using for the save, or if it finds no name, it will beep and report the fact.

AUTOMATIC BACKUP (ADFS)

Lee Calcraft

Whether you are using the Master ROM to implement an auto-save facility, or the method described above, you can create an additional command to back up any program to a separate drive by altering just one line of the above. Simply replace the third line from the bottom (S\$=\$(P%+1)) with the following:

```
S$=":1.$ .ProgBackup."+$(P%+1)
```

Now re-save the entire EXEC file under the name PB (for Program Backup), and every time that you type *PB the Basic program currently in memory will be saved to the directory ProgBackup on drive 1, using the so-called "incore" filename with the same format as above. Again the routine will report on its progress, and will inform the user if no suitable filename is found. **B**

TALKING TO THE ADFS (continued from page 43)

```
130 PROCinit
140 PROCdispal
150 END
160 :
170 DEFPROCinit
180 PROCtitle
190 PRINT"WHOLE DISC CATALOGUE"
200 PRINT"Current directory: "name$
210 level=0
220 FOR A=0 TO 128
230 ends%(A)=1
240 NEXT
250 :
260 DEFPROCdispal
270 back=FALSE
280 PROCfile
290 dir=0
300 IF fileno>ends%(level) THEN PROCdispcat
310 IF dir THEN ends%(level)=A:level=level+1:OSCLI("DIR "+file$(A-1)):PROCdispal
```

```
320 IF back THEN GOTO 270
330 IF level>0 THEN ends%(level)=1:level=level-1:back=TRUE:OSCLI("DIR ^")
340 ENDPROC
350 :
360 DEFPROCdispcat
370 PROCcatinfo(FALSE)
380 A=ends%(level)
390 REPEAT
400 dir=(file$(A,0)=2)
410 PRINTTAB(4+4*level);
420 IF dir THEN COLOUR0:COLOUR129
430 PRINTfile$(A);
440 COLOUR128:COLOUR7:PRINT
450 A=A+1
460 UNTIL A>fileno OR dir=TRUE
470 ENDPROC
480 :
490 REPORT:PRINT" at line ";ERL
500 *FX4
510 END
520 : B
```


PRINTER ROMS

If you feel that you are not getting the most out of your printer, or you can't cope with the multitude of printer codes, you might find one of the two ROMs reviewed by Geoff Bains quite useful.

The skill required to master the Escape codes used to get the most out of your dot-matrix printer is quite rightly held by many people to be on a par with quantum theory. These two ROMs go some way towards making life with your printer easier.

Product	PrintROM
Supplier	Windmill Software Smock House, Hull Lane, Terling, Chelmsford, Essex CM3 2QD. Tel. (024533) 371
Price	£15.95 (ROM) £10.95 (SWR)

PrintROM is available in ROM or on disc ready for sideways RAM, and in ten different versions for different printers. Most of the printers beloved by Beeb owners are catered for - Taxan KP810, Epson LX/RX/FX/MX, Canon PW1080, Centronics GLP, Brother M1009, Panasonic KXP1080/1, Shinwa CP-80, Juki 5500, MP165 and Star SG-10. However, the newer 24-pin machines are particularly noticeable by their absence from this list.

PrintROM is essentially a memory aid. With a series of star commands of obvious and memorable names, the most commonly used (and forgotten) printer code sequences can be easily sent to your printer. Like all star commands, these can be issued either in immediate mode, from within Basic programs or from Wordwise Plus, Interword and some other word processing software - either from the menu or embedded in the text.

The common printing effects are covered. Some commands are followed with ON or OFF to select the effect, and others require numbers. The *CHARSET command used to select the international character set presents the user with a mode 7 menu of the nine possibilities selectable with the cursor keys.

The *CMENU command presents a menu of all the other commands - handy for leaving any decision about a document's presentation until the last minute. However, this menu is somewhat ruined by not accepting lower case letters to select the effects. To have to switch to upper case (mentally or on the keyboard) in a menu which supposedly saves keypresses is just silly. In addition, a very useful *LPRINT command is provided, which prints the characters following it, and a graphics screen dump is provided for good measure (the Taxan dump is a good one but printer dumps are two a penny these days).

If you are totally baffled by your printer then PrintROM will help. It has the great advantage of being model-specific and so requires no setting up, and for the most part is pretty self-explanatory. However, a hefty £16 for what amounts to simple code substitution is not good value. For my part, I would prefer to invest a few pence in a list of codes to stick on the wall above the monitor.

Product	Hyperdriver
Supplier	DABS Press 76 Gardner Road, Prestwich, Manchester M25 7HU. Tel. 061-773 2413
Price	£29.95 inc VAT (ROM) £24.95 inc VAT (SWR)

Hyperdriver is an altogether different story. This is a cleverly thought out package that doesn't just substitute for your own mastery of your printer but augments it. It is of course much harder in itself to master but is ultimately worth the time taken. Hyperdriver is for Epson compatible printers. However, as the vast majority of printers around today are (at least) Epson compatible this shouldn't pose too many problems.

Like PrintROM, Hyperdriver is based on star commands which substitute for the hard-to-remember printer Escape codes. Many more printing effects are also controllable. There is little your printer can do which isn't covered by this ROM. The Hyperdriver commands are all two letter mnemonics. Although this makes the commands convenient to type, it doesn't help their memorability. True, they all follow a logical pattern - *UL for underline, *XU to cancel underline, *IT for italics, *XI to cancel italics - but the pattern is the author's and not

your own. I would have preferred to see full commands which could be abbreviated. However, this is a minor gripe. This command set, like all others, is soon learnt.

Hyperdriver does not just stop at code substitution. For a start, the software is much more 'intelligent' than PrintROM. If something is dangerous or impossible it says so. The error messages are always clear and appropriate. This ROM has the unusual habit of requiring acknowledgement of all error checks. After each error message you must press Return to continue. This does have the advantage that you cannot miss the error. However, it just gets tedious after a while. It would be nice to be able to switch this feature off!

A more useful feature is that Hyperdriver commands can be strung together into 'macros'. This both saves on memory space and is very convenient. The macros can be saved to disc and loaded back in with simple commands. By using different sets of macro definitions, a single document, liberally embedded with macro calls, can easily be printed in any number of different ways just by loading up the required macro definitions first.

All this is fine for users of Wordwise Plus or Interword, but View and View Professional users cannot embed commands in their documents. For them, Hyperdriver includes a complete View printer driver. The commands are embedded between the View highlights, and the driver interprets them to give the correct control codes to the printer when the document is printed. Of course that removes the WYSIWYG nature of View. Wordwise Plus is not a WYSIWYG word processor to start with.

But Hyperdriver can even fix this for you. First the Beeb is switched to 'test print' mode. Now all attempts to print are redirected to the screen without the usual reams of wasted paper trying

to get the layout right. Now switch on the 'screen effects' and the common printer effects such as enlarged text, italics, underlined, bold, superscript and subscript can all be produced

As well as making easier many standard printer effects such as *italics*, **enlarged**, **condensed**, **emphasised**, and ^{sub} or ^{super}script, Hyperdriver can also produce text in a pseudo near letter quality style of print on standard non-NLQ printers and screen-style print in the same range of sizes as you will find on the Beeb

on the screen - true WYSIWYG. Of course none of this is as convenient as using a word processor which provides true WYSIWYG in the first place but for View or Wordwise Plus users it's a boon.

Not only can Hyperdriver simulate the

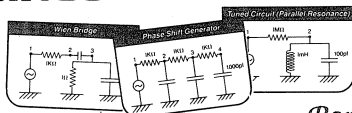
printer on the screen, but it can also print screen characters (including user defined ones) on the printer. 'Graphics print' produces a sort of continuous text printer dump of characters on paper. All the same printing effects are there but in a new font and at different sizes depending on the screen mode you are in. It's a nice touch and can be quite useful.

What is certainly useful for owners of older printers is the NLQ mode. This produces NLQ print on printers without an NLQ facility with three passes of the printhead over the paper. The result is not as fine nor as fast as some NLQ printers' output and is only available proportionally spaced. Nevertheless, it is a useful addition and quite adequate for most needs.

In many ways Hyperdriver and PrintROM are just two of a kind with the higher price of Hyperdriver just serving as a deterrent. However, looking beneath the surface and considering the way the ROMs treat errors, the thought that's gone into the way Hyperdriver is used with word processors and a million other good design features of Hyperdriver, the far greater value of this ROM stands out.

Many Beeb users can live in peace with their printer without assistance from anyone. However, if you are one of those who feel there is more to a printer than wading through lists of Escape codes, Hyperdriver will prove an ingenious blessing. B

CIRCUIT ANALYSIS



Part Two

Colin Attenborough completes his article on circuit analysis with a routine to display the results created using last month's program.

Last month I presented an analysis program and a simple user-hostile display program. Here is the complete DISPLAY program. It reads data written to disc by the analysis program (the long or the short version), and presents it in meaningful tabular or graphical form. If you alter line 260 of last month's magazine program to read:

```
260 IF choice%=1 @%=&90A:Z%=OPENIN OF$
:$$&900=OF$:CHAIN"Display"
then this month's program Display will be
chained in by it, and the result file will be
passed across. If the display program is run
independently, it will simply ask for the name
of an analysis data file. Note that this trick will
not work on the tube, and the "$&900=OF$"
should be left out of the line, or be adjusted so
as to use a different address. If you are using
the extended version of last month's program,
then you should alter line 450 instead, as
follows:
```

```
450 @%=&90A:Z%=OPENIN OF$:$&900=OF$:CH
AIN"Display5"
remembering that the Tube proviso still holds.
```

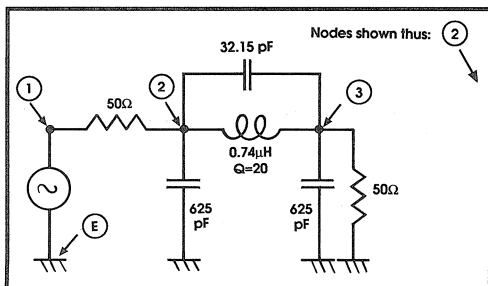


Figure 1. Filter Circuit

Once a suitable datafile has been opened DISPLAY then prompts the user for a node, accepting only node numbers existing in the analysed circuit. If the input node is selected,

the input impedance is calculated at each frequency after the user has been asked to select rectangular or polar representation (i.e. real and imaginary parts of magnitude and phase angle). For any other selected node, the ratio of node voltage to input voltage is calculated at each frequency, both in magnitude (given in decibels) and in degrees of phase shift.

The information is presented as a table (in paged mode) and then as a scaled graph with frequency as the X axis (unless a single frequency has been used). Data can be read from the graph by moving a cursor across the X axis using the left and right cursor keys. To make the cursor move faster, hold down Shift as well. The values of the results at the frequency indicated by the cursor line are displayed beneath the graph.

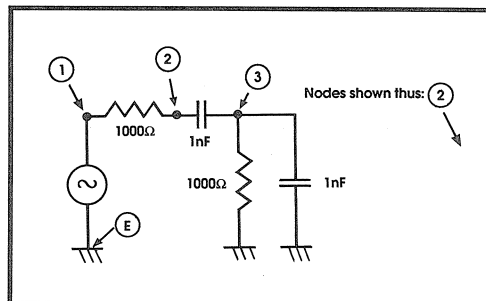


Figure 2. Wien Bridge Circuit

The user then has the option of seeing the results again, printing them, changing the selected node, editing the circuit file, writing a new circuit file, or re-running the display program. If a graph dump is selected, the cursor and the associated display are replaced by a summary of the limits of the result quantities. I have assumed that users will call their own graphics dump at line 3150.

CIRCUIT FILES

Last month I gave a simple circuit diagram and the corresponding circuit file. To see how other circuits are handled, we will take a look at a filter circuit (figure 1) and a Wien bridge network (figure 2). Their circuit files are reproduced as figures 3 and 4 respectively, and generally speaking they follow the pattern introduced last month, though there are one or two new features.

***FILTER**

```
*Filter Circuit
R1,1,2,50
R2,3,E,50
C1,2,E,625p
C2,3,E,625p
C3,2,3,32.15p
*a lossy inductor
*0.74 microhenries
*with a Q factor of 20
L1,2,3,0.74u,20
FMIN=10M,FMAX=100M,LOG,STEPS=10
IN=1
```

Figure 3. Filter Circuit File

Firstly, comments have been introduced. They must start on a new line, and be preceded by an asterisk. The analysis program completely ignores all comments. Secondly, figure 3 contains a lossy inductor; that is to say an inductor which has a non-inductive element to its impedance (like all real inductors). This is represented by the string:

L1,2,3,0.74u,20

In other words, inductor L1 lies between nodes 2 and 3. It has an inductance of 0.74 micro Henries, and a Q factor of 20 (the "Q" factor of a coil gives a measure of its quality - how near to an ideal inductor it actually is).

***WIEN**

```
*Wien bridge
R1,1,2,1000
C1,2,3,1000p
R2,3,E,1000
C2,3,E,1000p
FMIN=100K,FMAX=200K,DF=10K
IN=1
```

Figure 4. Wien Bridge Circuit File

The only other new element also appears in the filter file. Here the frequency range for analysis is given as:

FMIN=10M,FMAX=100M,LOG,STEPS=10

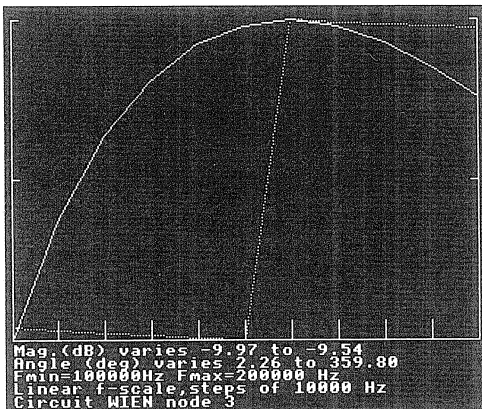
This requests an analysis in the range 100-200 MHz. But instead of the linear sweep of last month's example and of the Wien bridge, a logarithmic sweep is requested. The "STEPS=10" means that 10 steps are required in the given range. While on the subject of frequency sweeps, if you require analysis at a single frequency only, then you should replace the frequency definition line with one such as the following:

F=10M

which will cause analysis to take place at 10 MHz only.

HOW THE PROGRAMS WORK

The analysis program simulates forcing a current of one ampere into the input terminal and out of the common terminal of the circuit to be analysed. It calculates the voltages at all the nodes by solving N simultaneous equations in N variables, where N is the number of nodes. The variables of the equations are in general complex (unless the circuit contains neither inductors nor capacitors), so the voltages are also in general complex. At the end of the analysis program the real and imaginary parts of all node voltages are stored to disc, together with information about the number of frequencies, type of frequency sweep and circuit file name.



The display program reads data from the disc, and calculates the ratio of the voltage at the selected node to the voltage at the input node. This gives the voltage gain between the input and the selected node. If the selected node is the input node, the input node voltage is displayed as the input impedance.

The program may give errors if there is a very large ratio between the values of components used - the capacitors in a quartz crystal's equivalent circuit, for example. The program can handle relatively complex circuits with the following constraints. The number of each type of element used is limited to 30 (25 in the magazine disc version) by the size of the arrays (and by your patience during analysis). The number of frequency steps is limited to 80.

Some error trapping is provided: if you try to read a non-existent file, you'll be invited to try again. If you miss a connection between two nodes, you get a division by zero error, and can re-enter Wordwise to correct the circuit file. This also happens with 'No such variable' errors, which mean that information is missing from the circuit file. The program also checks the lines it reads from the circuit file to make sure it understands them and that they have the correct number of parameters.

PROGRAM NOTES

The analysis program will happily read files written using VIEW as well as those written with Wordwise, but the VIEW user must provide different methods of re-entering the editor.

The program normally reads disc files but if you have a RAMdisc (e.g. Opus Challenger), it will speed operation of the program.

ADFS users should alter line 3210 to read:

```
*CAT O
```

If you write a !BOOT file:

```
*BASIC
```

```
CH. "ANALYSE" (adjust name as reqd)
```

and type *OPT 4,3 then you can just press Shift-Break to start the analysis process, even from within Wordwise.

This month's magazine disc contains a version of the extended analysis program (capable of analysing circuits containing transistors, OP amplifiers, FETs, transformers etc.), modified as described here, together with this month's display program. A number of circuit descriptor files are also included. Additionally there is a text file describing how to use the extended features.

If readers would like a listing of the extended version of last month's program (including notes on its use), they should send an A5 SAE to the editorial address, with the envelope marked "Circuit Analysis".

```
10 REM Program Circuit Display
20 REM Version B 0.5
30 REM Author C.Attenborough
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 ON ERROR CLOSE#0:@%=&96A:MODE7:REP
ORT:PRINT" at line ";ERL:END
110 PROCinit
120 MODE 7
```

```
130 IF (!&900 AND &FFFF)<>&2E4F THEN P
ROCofferfiles ELSE filename$=&900:Z%=OP
ENIN filename$:&900=""
140 table%=TRUE:nodeok%=FALSE:polar%=F
ALSE:print%=FALSE:dump%=FALSE
150 INPUT #Z%,CF$,fi,fa,df,in$,st%,nod
emax%,steps
160 PROCchoosenode
170 IF node%=in% PROCrcpol
180 VDU23,1,0;0;0;0;
190 CLS
200 PRINTTAB(10,12);"Please wait...."
210 PROCreadfile
220 MODE3:PROCTable
230 IF print%=FALSE MODE 4:PROCgraphic
:print%=FALSE:dump%=FALSE
240 MODE7
250 PROCOptions
260 IF choice%=1 GOTO220
270 IF choice%=2 print%=TRUE:GOTO220
280 IF choice%=3 AND st%>>1 dump%=TRUE
:GOTO230
290 IF choice%=4 THEN &900=filename$:
GOTO120
300 IF choice%=5 load$="KEY9 *WORDWISE
|M 2"+CF$+"|M |":PROCreload
310 IF choice%=6 load$="KEY9 *WORDWISE
|M |":PROCreload
320 IF choice%=7 CLOSE#0:RUN
330 CLOSE#0:@%=&90A
340 END
350 :
1000 DEF PROCinit
1010 DIM F(80),result(80,2),RVIN(80),IV
IN(80),RVN(80),IVN(80),result$(2),result
min(2),resultmax(2)
1020 F$="Freq,HZ"
1030 ENDPROC
1040 :
1050 DEF PROCreadfile
1060 FOR J%=1 TO steps
1070 INPUT #Z%,F(J%)
1080 FOR K%=1 TO nodemax%
1090 INPUT #Z%,RV:INPUT #Z%,IV
1100 IF K%=in% RVIN(J%)=RV:IVIN(J%)=IV
1110 IF K%=node% RVN(J%)=RV:IVN(J%)=IV
1120 NEXT
1130 NEXT
1140 FOR J%=1 TO 2
1150 resultmax(J%)=-1E30:resultmin(J%)=
1E30
1160 NEXT
1170 FOR J%=1 TO steps
1180 IF node%=in% PROCrin ELSE PROCtran
s
1190 FOR K%=1 TO 2
1200 IF result(J%,K%)>resultmax(K%) res
ultmax(K%)=result(J%,K%)
1210 IF result(J%,K%)<resultmin(K%) res
ultmin(K%)=result(J%,K%)
1220 NEXT
1230 NEXT
1240 ENDPROC
```

```

1250 :
1260 DEF PROCchoosenode
1270 REPEAT
1280 CLS
1290 INPUTTAB(10,12);"What node? "node%
1300 IF node%>0 AND node%<=nodemax% nod
eok%=TRUE
1310 IF nodeok%=FALSE CLS:PRINTTAB(10,1
0);"No such node":PRINTTAB(10,12);"Press
any key":G%=GET
1320 UNTIL nodeok%=TRUE
1330 ENDPROC
1340 :
1350 DEF PROCcrepol
1360 REPEAT
1370 CLS
1380 PRINTTAB(8,12);"Rectangular or pol
ar? (R/P)"
1390 G$=GET$
1400 UNTIL G$="R" OR G$="r" OR G$="P" O
R G$="p"
1410 IF G$="P" OR G$="p" polar%=TRUE
1420 ENDPROC
1430 :
1440 DEF PROCTable
1450 *FX21,0
1460 IF print% VDU2:PRINT:PRINT" Cir
cuit ",CF$;," node ";node%:PRINT ELSE VD
U14
1470 @%=&20205
1480 IF NOT print% VDU28,0,1,79,0
1490 PRINTTAB(5);F$;TAB(30);result$(1);
TAB(60);result$(2)
1500 IF NOT print% VDU28,0,23,79,2
1510 FOR J%=1 TO steps
1520 PRINTTAB(5);F(J$);TAB(30);result(J
$,1);TAB(60);result(J$,2)
1530 NEXT
1540 PRINT
1550 @%=&90A
1560 IF NOT print% PRINTTAB(34)"Press a
ny key"
1570 VDU15
1580 *FX21,0
1590 IF NOT print% G%=GET
1600 VDU3
1610 VDU26
1620 ENDPROC
1630 :
1640 DEF PROCgraphic
1650 nograph%=TRUE
1660 FOR K%=1 TO 2
1670 IF FNconst(K%) PROCnograph ELSE PR
OCannotate:nograph%=FALSE
1680 NEXT
1690 IF nograph%=FALSE PROCaxes
1700 FOR K%=1 TO 2
1710 IF NOT FNconst(K%) PROCgraph
1720 NEXT
1730 IF dump% PROClabel
1740 IF dump%=FALSE AND nograph%=FALSE
PROCcursor

```

```

1750 IF nograph%=TRUE PROClabel
1760 IF dump% PROCgdump
1770 ENDPROC
1780 :
1790 DEF FNconst(Q%)
1800 IF resultmax(Q%)-resultmin(K%)<=AB
S(0.001*resultmax(Q%)) THEN=TRUE ELSE=FA
LSE
1810 :
1820 DEF PROCnograph
1830 @%=&20205
1840 PRINTTAB(0,25+K%);result$(K%);" co
nstant at ";resultmax(K%)
1850 @%=&90A
1860 ENDPROC
1870 :
1880 DEF PROCaxes
1890 VDU23,1,0;0;0;0;
1900 xcl%=0:xch%=1279
1910 ycl%=200:ych%=1000
1920 MOVE xcl%,ycl%:DRAW xcl%,ych%
1930 MOVE xcl%,ycl%:DRAW xch%,ycl%
1940 DRAW xch%,ych%
1950 MOVE xcl%,ych%
1960 DRAW xcl%+20,ych%
1970 MOVE xcl%,0.5*(ycl%+ych%)
1980 DRAW xcl%+20,0.5*(ycl%+ych%)
1990 MOVE xch%,ych%
2000 DRAW xch%-20,ych%
2010 MOVE xch%,0.5*(ycl%+ych%)
2020 DRAW xch%-20,0.5*(ycl%+ych%)
2030 dx=(xch%-xcl%)/(steps-1)
2040 IF steps<21 PROCvertlines
2050 ENDPROC
2060 :
2070 DEF PROCvertlines
2080 FOR J%=2 TO steps
2090 MOVE (J%-1)*dx,ycl%
2100 DRAW (J%-1)*dx,ycl%+50
2110 NEXT
2120 ENDPROC
2130 :
2140 DEF PROCgraph
2150 yscale=(ych%-ycl%)/(resultmax(K%)-
resultmin(K%))
2160 MOVE xcl%,(result(1,K%)-resultmin(
K%))*yscale+ycl%
2170 FOR J%=2 TO steps
2180 PLOT(16*K%-11),xcl%+(J%-1)*dx,(res
ult(J$,K%)-resultmin(K%))*yscale+ycl%
2190 NEXT
2200 ENDPROC
2210 :
2220 DEF PROCannotate
2230 @%=&20205
2240 PRINTTAB(0,25+K%);result$(K%);" va
ries ";resultmin(K%);" to ";resultmax(K%
)
2250 @%=&90A
2260 ENDPROC
2270 :
2280 DEF PROClabel

```



```

2290 IF st%>1 PRINTTAB(0,28);"Fmin=";fi
;"Hz Fmax=";fa;" Hz"
2300 IF st%=2 PRINTTAB(0,29);"Linear f-
scale, steps of ";df;" Hz"
2310 IF st%=3 PRINTTAB(0,29);"Log f-sca
le, ";steps;" steps"
2320 PRINTTAB(0,30);"Circuit ";CF$;" no
de ";node%
2330 ENDPROC
2340 :
2350 DEF PROCcursor
2360 halfwidth=640/(steps-1)
2370 PRINTTAB(0,26);SPC(40)
2380 PRINTTAB(0,27);SPC(40)
2390 PRINTTAB(0,26);"Circuit ";CF$;" no
de ";node%
2400 IF st%=2 PRINTTAB(0,27);"Linear f-
scale, steps of ";df;" Hz"
2410 IF st%=3 PRINTTAB(0,27);"Log f-sca
le, ";steps;" steps"
2420 PRINTTAB(0,28);F$
2430 PRINTTAB(0,29);result$(1)
2440 PRINTTAB(0,30);result$(2)
2450 GCOL 4,0
2460 oldx%=-10:x%=512:@%=&20205
2470 REPEAT
2480 IF INKEY(-1)dx%=16 ELSE dx%=4
2490 IF INKEY(-26) AND x%>xcl%+dx% x%=x
%-dx%
2500 IF INKEY(-122) AND x%<xch% x%=x%+d
x%
2510 IF x%<oldx% MOVE oldx%,ycl%:DRAW
oldx%,ych%:MOVE x%,ycl%:DRAW x%,ych%:old
x%=x%
2520 band%=INT(x%/halfwidth)
2530 index%=INT(band%+1) DIV 2+1
2540 PRINTTAB(15,28);F(index%)
2550 PRINTTAB(15,29);result(index%,1)
2560 PRINTTAB(15,30);result(index%,2)
2570 PRINTTAB(11,31);"Press SPACE to ex
it";
2580 UNTIL INKEY(-99)
2590 *FX15 0
2600 @%=&90A
2610 ENDPROC
2620 :
2630 DEF PROCtrans
2640 PROCdiv(RVN(J%),IVN(J%),RVIN(J%),I
VIN(J%))
2650 PROCpolar(RQ,IQ)
2660 result(J%,1)=20*LOG(result(J%,1))
2670 result$(1)="Mag. (dB)":result$(2)="
Angle (deg)"
2680 ENDPROC
2690 :
2700 DEF PROCrin
2710 IF NOT polar% result(J%,1)=RVIN(J%
):result(J%,2)=IVIN(J%):result$(1)="Re Z
in ohms":result$(2)="Im Zin ohms":ENDPRO
C
2720 PROCpolar(RVIN(J%),IVIN(J%))
2730 result$(1)="Mod Zin ohms":result$(

```

```

2)="Ang Zin deg"
2740 ENDPROC
2750 :
2760 DEF PROCpolar(RE,IM)
2770 result(J%,1)=SQR(RE*RE+IM*IM)
2780 result(J%,2)=FNang
2790 ENDPROC
2800 :
2810 DEF FNang
2820 IF RE=0 =90*SGN(IM)
2830 ang=DEG(ATN(IM/RE))
2840 IF RE<0 ang=(180+ang)
2850 IF ang <0 ang=ang+360
2860 =ang
2870 :
2880 DEF PROCdiv(RA,IA,RB,IB)
2890 RQ=RA*RB+IA*IB:IQ=IA*RB-IB*RA
2900 DQ=RB*RB+IB*IB
2910 RQ=RQ/DQ:IQ=IQ/DQ
2920 ENDPROC
2930 :
2940 DEF PROCoptions
2950 dump%=FALSE:print%=FALSE:REPEAT
2960 PRINTTAB(5,1)"1 See results again"
2970 PRINTTAB(5,4)"2 Print table"
2980 PRINTTAB(5,7)"3 Dump graph"
2990 PRINTTAB(5,10)"4 Choose another no
de"
3000 PRINTTAB(5,13)"5 Edit this circuit
file"
3010 PRINTTAB(5,16)"6 Create a new circ
uit file"
3020 PRINTTAB(5,19)"7 Display another O
. file"
3030 PRINTTAB(5,22)"8 Quit"
3040 G%=GET
3050 choice%=G%-48
3060 UNTIL choice%>0 AND choice%<9
3070 ENDPROC
3080 :
3090 DEF PROCreload
3100 OSCLI load$
3110 A%=138:X%=0:Y%=137:CALL &FFF4:END
3120 ENDPROC
3130 :
3140 DEF PROCgdump
3150 REM Your dump called here
3160 ENDPROC
3170 :
3180 DEF PROCofferfiles
3190 REPEAT
3200 CLS
3210 *CAT
3220 PRINT"Select circuit analysis fil
e:"
3230 INPUT TAB(20)"O."filename$
3240 filename$="O."+filename$
3250 Z%=OPENIN filename$
3260 IF Z%=0 VDU7:PRINT:PRINTTAB(2);"Fi
le not found. Press any key":G%=GET
3270 UNTIL Z%<0
3280 ENDPROC

```

EXPLORING ASSEMBLER

Part 8

A series for complete beginners
to machine code by Lee Calcraft

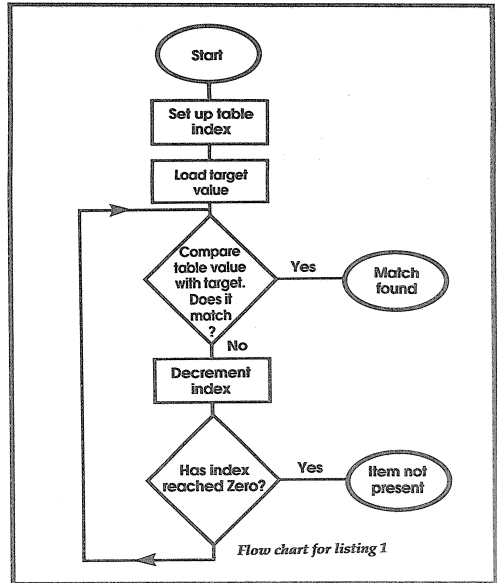
This month: Searching and look-up tables

A SIMPLE BYTE-SEARCH ROUTINE

This month we continue our discussion of the use of the CMP instruction with a look at the problem of searching. We begin with a very simple byte-search routine. Its function is to check a list of up to 255 elements to see if a particular byte is present. We will assume that the list is held at &A00 onwards, and that its elements are in no particular order. Essentially the search routine will need to compare the target value with each of the elements in turn until either a match is found, or until the list is exhausted.

The program in listing 1 achieves this. If you type it in and run it, you will be asked for a target value to be searched, and you should supply any integer between 0 and 255. The machine code search routine will then be called, and the result displayed. For the purposes of the example, the table of values examined by the program is created with a short Basic routine at line 300. This places a set of data of value between 0 and 127 in the table. Thus if you search for any integer in this range you should get a match, and not otherwise.

As we look through the listing, you may find it useful to check through the accompanying flow chart. The main routine begins at line 130 where the Y register is loaded with the number of items in the table, and the accumulator is loaded with the target value input by the user. In line 160 the target value is compared with the first value in the table (CMP table-1,Y). We have used indexed addressing for the CMP instruction so as to be able to point to each element in the table in turn. You will also notice that we have used *table-1* as the base address rather than *table*. This is because we are decrementing the Y register after each failed comparison, and checking that we have not



reached the end of the list by testing Y for zero (line 220). Thus the last value of Y for which a comparison test is made is 1. By using a base address of *table-1* we ensure that the final element of the table at address *table* is checked.

There are two exit conditions: if the number of items in the table is exhausted, or if a match is found. In the former case we return with zero in the accumulator (line 210), in the latter the accumulator is loaded with &FF (line 240). The Basic program has then to check the contents of the accumulator on exit in order to display the result of the search. This is accomplished in line 380:

Z=USR(&900) AND &FF

The USR function is an alternative way of calling a machine code routine (briefly introduced in part 5 of the series), and unlike the CALL statement, it returns a value. This value is a four-byte word made up of the contents of the accumulator, X and Y registers and status register at the time of exit. To obtain the contents of the accumulator (the lowest byte of the value returned by USR), we AND this four-byte word with &FF. The result is then displayed in line 390.

STRING SEARCH

The second program implements a string search. Here the user supplies a target string

Listing 1

```
10 REM Byte Search
20 REM Author Lee Calcraft
30 REM Version B 0.5
40 :
50 target=&70:total=&71:table=&A00
60 MODE0
70 FOR pass=0 TO 1
80 P%=&900
90 [
100 OPT pass*3
110 \ **SIMPLE SEARCH**
120 .main
130 LDY total
140 LDA target
150 .loop \ Byte compare loop
160 CMP table-1,Y \ Compare byte
170 BEQ match \ They match
180 DEY
190 BNE loop \ Check for table end
200 .nomatch
210 LDA #0 \ Indicate failure
220 BEQ exit
230 .match
240 LDA #&FF \ Indicate success
250 .exit
260 RTS
270 ]
280 NEXT
290 :
300 FOR A=0 TO 255
310 table?A=A/2
320 NEXT
330 :
340 ?total=255
350 REPEAT
360 INPUT"Search for value "value
370 ?target=value
380 Z=USR(&900) AND &FF
390 IF Z>0 PRINT"Item present" ELSE PR
INT"Item not present":VDU7
400 UNTIL FALSE
```

which is matched against a list of words. Again the program indicates whether a match was or was not found by loading the accumulator with &FF or 0 respectively. Because we are matching complete words, rather than single bytes, the routine is a little more complex than the search discussed above. But the approach is essentially similar in that it uses the LDA and CMP instructions to repeatedly compare single bytes.

In the present search program we will assume that the table of words uses the carriage return character (&0D) as an end of word marker, and that the end of the table is indicated by a byte of

value zero, immediately following the carriage return associated with the last word in the table. Lines 440 to 500 of the program are used to set up the table. For our example, I have used an alphabetical sequence, although this particular search makes no assumption about the order of the words supplied.

If you run the program, you may enter any word at the prompt, and you will be informed whether it is present in the list or not. As you will observe, the search is not case sensitive. This is achieved by operating on the user input characters with AND #&DF. This is achieved in line 200 of the program, and forces each character of the user-supplied string into upper case. The technique assumes that only alphabetic characters will be input. You can see how it works if you type the following three lines:

```
PRINT ASC("A")
PRINT ASC("a")
PRINT ASC("a") AND &DF
```

You will see that the first and last results are the same.

Listing 2

```
10 REM String Search
20 REM Author Lee Calcraft
30 REM Version B 1.0A
40 :
50 word=&B00:table=&A00
60 MODE0
70 FOR pass=0 TO 1
80 P%=&900
90 [
100 OPT pass*3
110 \ **STRING SEARCH**
120 .main
130 LDY #&FF
140 .nextword \ New word start
150 LDX #&FF
160 .letter \ New letter start
170 INX
180 INY
190 LDA word,X
200 AND #&DF \ Force upper case
210 CMP table,Y \ Make comparison
220 BNE newword \ No match
230 CMP #13 \ Whole word match?
240 BNE letter
250 LDA #&FF \ Indicate success
260 JMP end
270 :
280 .newword \ Find start of
```

```

290 DEY          \ next word
300 .newloop
310 INY
320 LDA table,Y
330 CMP #13
340 BNE newloop
350 LDA table+1,Y \ Check ahead for
360 BNE nextword  \ Table end marker
370 LDA #0        \ Table end so
380 .end          \ search fails
390 RTS
400 ]
410 NEXT
420 :
430 Z=0
440 REPEAT
450 READ Z$
460 PRINTZ$
470 $(table+Z)=Z$
480 Q=LEN(Z$)+1:Z=Z+Q
490 UNTIL Q=1
500 Z?(table-1)=0
510 :
520 REPEAT
530 INPUT"Word to be searched "word$
540 $word=word$
550 Z=USR(&900) AND &FF
560 IF Z>0 PRINT"Item present" ELSE PR
INT"Item not present":VDU7
570 UNTIL FALSE
580 :
590 DATA POSIT,POSITION,POSITIVE,POSIT
RON,POSNET,POSOLOGY,POSS,POSSE,POSSESS
600 DATA POSSET,POSSIBLE,POSSIE,POSSUM
,POST,POSTAGE,POSTAL,POSTEEN,POSTER,

```

HOW THE PROGRAM WORKS

As with the last program, the table of data is held from &A00 onwards, and we have again restricted the size of the table to a maximum of 255 characters. This keeps the coding a little simpler than it would otherwise be. Again we have used Y as an index to the table, and this time we also need an index to the current test character in the target word, since we must test for the presence of the target word a single character at a time.

The program begins by loading the two index registers X and Y with &FF. We have not loaded them with 0, as one might have expected, because it is convenient to have the INX and INY instructions which move to the next character, at the start of the main loop.

Thus by loading them with &FF and incrementing them by one, we ensure that both start with the value zero. Line 190 then loads in the first character of the target string, which is forced into upper case. The result is immediately compared to the first character in the table. If the two match, a test is carried out on one of the two matching characters to see if it is a word terminator (CMP #13 in line 230). If so, it means that two complete words have matched (after successive character matches), and the program ends. Otherwise it proceeds to test the next pair of characters.

If on the other hand the two characters do not match, the program branches to *newword* at line 280. The purpose of this routine is to move along the table to the start of the next word, so that the comparison routine can be called again. When this check is being made, a special look ahead test must be performed to check for the end of table marker. This is performed in line 350. If the end of the table has been reached, the program terminates with zero in the accumulator, indicating a failed search.

In order to make this program work with a search table larger than 255 characters, we would need to use indirect indexed addressing (introduced in part 4 of the series). In this case, the compare instruction would take the form:

CMP (base),Y

The location *base* would initially contain the address of the start of the table, but each time Y reached the value zero, after the INY instruction, the top byte of the 16-bit address stored at *base* +1 would be incremented by 1. This would mean that the routine could be used with tables as large as the 6502's 64K memory map will allow.

LOOK-UP TABLES

So far in our discussion of searching techniques, we have been concerned only with discovering whether or not a target byte or word is present in a data table. In many applications however, the result of a tabular search will be a value or address. In this case the table of data is referred to as a look-up table, for obvious reasons. Many examples spring to mind. A telephone book program might return with a telephone number on finding a surname match in its table, or more technically, a SIN table would return with the SIN of the target value. Or to take a final example, every time that you issue a star command at the computer, a very similar

matching process takes place. Here the look-up value which the search routine returns will be the address of the routine corresponding to the star command issued.

0931 60	0	RTS
	2	POSITION
	4	POSITION
	6	POSITION
	8	POSITION
	10	POSITION
	12	POSITION
	14	POSITION
	16	POSITION
	18	POSITION
	20	POSITION
	22	POSITION
	24	POSITION
	26	POSITION
	28	POSITION
	30	POSITION
	32	POSITION
	34	POSITION
	36	POSITION
	38	POSITION
	40	POSITION
	42	POSITION
	44	POSITION
	46	POSITION
	48	POSITION
	50	POSITION
	52	POSITION
	54	POSITION
	56	POSITION
	58	POSITION
	60	POSITION
	62	POSITION
	64	POSITION
	66	POSITION
	68	POSITION
	70	POSITION
	72	POSITION
	74	POSITION
	76	POSITION
	78	POSITION
	80	POSITION
	82	POSITION
	84	POSITION
	86	POSITION
	88	POSITION
	90	POSITION
	92	POSITION
	94	POSITION
	96	POSITION
	98	POSITION
	100	POSITION
	102	POSITION
	104	POSITION
	106	POSITION
	108	POSITION
	110	POSITION
	112	POSITION
	114	POSITION
	116	POSITION
	118	POSITION
	120	POSITION
	122	POSITION
	124	POSITION
	126	POSITION
	128	POSITION
	130	POSITION
	132	POSITION
	134	POSITION
	136	POSITION
	138	POSITION
	140	POSITION
	142	POSITION
	144	POSITION
	146	POSITION
	148	POSITION
	150	POSITION
	152	POSITION
	154	POSITION
	156	POSITION
	158	POSITION
	160	POSITION
	162	POSITION
	164	POSITION
	166	POSITION
	168	POSITION
	170	POSITION
	172	POSITION
	174	POSITION
	176	POSITION
	178	POSITION
	180	POSITION

But before discussing this kind of look-up, I want to mention a very smart look-up technique which can be used in certain circumstances, and which avoids the necessity for searching altogether. Essentially, it uses the address offset of each item in the table as an index to the table's contents. Thus for example, if we were to create a look-up table for SIN values from 0 to 180 degrees, we could use the following in Basic:

```
FOR A=0 TO 180
  A?base=250*SIN(A*PI/180)
NEXT
```

This will place an integer corresponding to $250 \cdot \sin(A)$ at location $base+A$ (where A is in degrees). To find the scaled SIN of N degrees, a piece of machine code would only need to perform:

```
LDY N
LDA base,Y
```

This works without the need for any searching because we have arranged the table so that the scaled value of $\sin(10)$ is in location $base+10$ and so on.

Of course, where data cannot be indexed in this very convenient way, we are forced to perform a search before the data corresponding to the target item can be looked-up. In such cases there are usually two ways of storing the data to be returned. It can either be embedded in the search table, adjacent to the item to which it refers, or it can be kept in a separate table linked to the first. The advantage of the first method is that each search item is held next to the look-up item, but the disadvantage is that the search routine must continually skip around the look-up data at the end of each search item. In the following we will add a

number of lines to the string search program above to allow it to perform a look-up function similar to that found in ROM software for interpreting star commands. We will use separate command and address tables.

To achieve this, just add the lines given in listing 3 to those of listing 2. When you run the program you will now see that each of the words in the table, which can be thought of as a command table, is displayed with an index number. Index numbers are given as even numbers only, since they will be used to index a two-byte address, as we shall see in a moment. The result of each positive search now includes a display of the index number for the target word. If we were using this code to interpret star commands, the table would be replaced by a list of command names, and the index returned whenever a match was achieved would be used to jump (using the JMP instruction) to the corresponding piece of code.

Listing 3

```
35 REM With look-up index
50 word=&B00:table=&A00:count=&70
125 LDA #0:STA count
345 INC count:INC count
430 Z=0:N=0
460 PRINT N SPC8 Z$
465 N=N+2
565 IF Z>0 PRINT"Index="";count
```

To make this work you would need to set up a jump table somewhere in memory, say at address $base$, and fill it with 16-bit addresses for the code for each star command. Then the following would do the trick:

```
382BEQ quit
383LDY count
384LDA base,Y:STA &80
385LDA base+1,Y:STA &81
386JMP (&80)
388.quit
```

This would execute an RTS (line 390) if no match were found, otherwise $\&80$ and $\&81$ are loaded with the look-up address, and an indirect jump is performed. This has the following effect: "Jump to the address made up from the two bytes stored at the following adjacent locations: $\&80$ and $\&81$ ".

Next month we will be returning to some pure arithmetic, and will take a look at routines for performing multiplication and division.

B

In this series of workshops Sheridan Williams explains how to make the most of the effects that your printer can produce. Necessarily the first article in the series is rather elementary, however future articles will cover FX3 and FX5, DIP switch settings, in-built printer buffers, configuration at power-on and more.

There are, of course, many dozens of popular printers, all quite easy to connect to the BBC micro, Master or Compact. Fortunately for once, everything shown in this series will work equally well on all Acorn's micros, including the Archimedes.

Epson printers, and the so-called Epson-compatible printers, are by far the most common category of printer. It therefore makes sense to concentrate on this range. However, for those with printers that are not Epson-compatible, this Workshop will show how to control these too. For those with daisywheel printers, the only effects that you can get are: underlining, backspace (hence overprinting), Tabs, number of characters per inch, and lines per inch, and on some printers - reverse paper feed. You cannot change the style of the characters (without changing the daisy wheel), nor are sophisticated graphics available. This series will, however, show how to obtain those effects that are relevant to daisywheel printers.

CONTROLLING THE PRINTER WITH BASIC

BBC Basic has three VDU commands that are of direct interest to us for printer control:

VDU1 - sends the next character to the printer only.
VDU2 - enables the printer.
VDU3 - disables the printer.

After issuing a VDU2 instruction all output normally sent to the screen, will be directed to the printer as well (in a later article we will cover how to send output solely to the printer).

Try this example:

```

10 FOR i=1 TO 10
20 IF i MOD 2=0 THEN VDU2
30 PRINT i;" ";
40 VDU3
50 NEXT

```

This program will print the numbers 1 to 10 on the screen, but only the even numbers will be sent to the printer. Notice how (in line 40) a VDU3 is executed even when in some cases a VDU2 has not previously been issued. This does not matter at all.

GLOSSARY

ASCII is the American Standard Code for Information Interchange. The most popular code for representing characters on microcomputers. See this month's First Course for more information on the subject.

In order to implement all the special effects offered by the printer we need to understand the use of VDU1. You should note however, that VDU1 will not work until a VDU2 has been issued previously. Once VDU2 has been given, the printer stays active and all subsequent PRINT and VDU statements send their output to both screen and printer. If you wish characters to be sent only to the printer, VDU1 is what you want. Note however, that VDU1 is different, and needs to be given before *each* character that is sent to the printer.

VDU1 simply sends the next character directly to the printer without displaying it on the screen, and we need this for issuing the special characters 0 to 31 inclusive, whose effect on the screen might be disastrous. For example, character 12 would clear the screen if sent with `PRINT CHR$12` or `VDU 12`, both of which have the same effect. To restrict its effect to the printer (and not the screen), we issue `VDU1,12` which sends character 12 to the printer only where it does the next best thing to clearing the screen and activates a **form feed**. Make sure your printer is switched on, is on-line, and has

paper in it. Now try:

```
VDU2
VDU1,12
VDU3
```

Alternatively, this can all be combined into one statement:

```
VDU2,1,12,3
```

Note how VDU commands may be strung together. After issuing this you will see a sheet of paper turned up on the printer.

Certain codes like 12 - FF (form feed), 13 - CR (carriage return), 10 - LF (line feed), are universal, and will hence have the same effect on all printers. Others like 15 (which in the table of ASCII codes is known as SI), are printer dependent. On Epson compatibles, character 15 switches the printer into condensed printing mode (17 characters per inch instead of the normal 10 ch/in); on the NEC PC-8023B-C printer you will need to send two characters to achieve the same result, namely 27 and 69.

From now on all printer codes will be given using VDU commands, and it will be assumed that a VDU2 has been given at some earlier point in the program. For example, to print in condensed mode on an Epson compatible printer you must issue VDU1,15, on the NEC PC-8023B-C you would issue VDU1,27,1,69. Notice how *each* printer code *must* be prefixed with 1. Can you see the difference between VDU1,27,1,69 and VDU1,27,69. If not try the latter and you will see that the letter E appears on the screen. This is because 69 is the ASCII code for the letter E, and as VDU1 sends one character to the printer *only*, when you omit the 1 the 'E' gets sent to the screen as well.

UNDERSTANDING THE MANUALS

Before continuing, it is essential that you can read and interpret the instructions given in the manual. So turn to the page on Condensed or Compressed printing, or Changing the font/character size. You must remember that the manuals are written for users of machines other than the Acorn range, and if they said "For condensed print use VDU1,15" this would be of no use to non-Acorn users. To quote from the TAXAN KP-810 manual:

SI

Condensed Mode Setting

Code: <0F>H <15>10

BASIC Syntax: CHR\$(&HF); CHR\$(15);

This is followed by a list of conditions which should be read, as it is here that the differences between printers often lie. For example, on some printers the chosen effect is cancelled when a new line is turned up, on others it

remains in force. Most manuals then go on and give various examples, which in most cases are far from clear, simply because they assume non-BBC Basic.

The most important information to extract from the headings in the manual is the characters to send to obtain a particular effect. You might as well use the base 10 values, as there is nothing to be gained from using the hexadecimal values prefixed of suffixed with an H. These decimal numbers may appear on their own, as in <15> or as Basic syntax as in CHR\$(15); in either case it is the number 15 that is required.

Armed with the knowledge that we can send the characters 0 to 127 to the printer, and that characters 32-127 produce the ordinary printable characters, we are apparently only left with characters 0-31 to achieve all the possible special effects. As there are many more than 32 special effects, how are these obtained? The answer is by means of Escape sequences.

Character 27 is the standard ASCII code for the Escape character. This is the character adopted by printer manufacturers to signal that there is a special sequence about to follow. For example, to produce underlined text the printer requires the sequence 27,45,1. To cancel this, the printer requires the sequence 27,45,0. Using the VDU1 statement this equates as follows:

```
27,45,1 becomes VDU1,27,1,45,1,1
```

```
27,45,0 becomes VDU1,27,1,45,1,0
```

Be careful: it is very easy to forget to prefix the final 1 with a 1. Try the following on your printer:

```
10 VDU2 20 PRINT"Ordinary, ";
```

```
30 VDU1,27,1,45,1,1:PRINT"Underlined";
```

```
40 VDU1,27,1,45,1,0:PRINT" Normal"
```

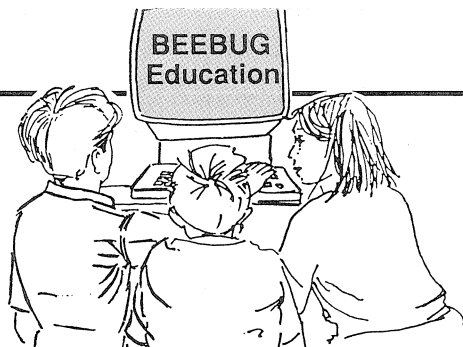
```
50 VDU3
```

Figure 1 is a table of the ASCII codes from 0 to 31 together with the standard abbreviations used to indicate them in some printer manuals.

	0	1	2	3	4	5	6	7	8	9
0	NUL	SH	SX	EX	ET	EQ	AK	BEL	BS	HT
10	LF	VT	FF	CR	SO	SI	DE	DC1	DC2	DC3
20	DC4	NK	SN	EB	CAN	EM	SB	ESC	-->	<--
30	^									

Fig. 1

In part two we will complete the single character codes, and cover the most common requirements of printers, namely underlining, enlarged, bold, italics, £ and characters. B



By Mark K. Sealey

This month in Beebug Education I want to look at some of the ways that the Acorn family of computers has been used in language development with pupils of all ages. The aim is to give teachers already familiar with other uses of the micro, such as maths and science, some pointers as to the way they might expand these, and to assess the likelihood of success in the light of accepted learning theory.

Suites or individual programs in this area fall into one or more of four categories. Each of these will be discussed, some suggested uses outlined, and a new example of the most worthwhile will be briefly examined.

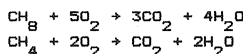
Firstly, word processing is by now such a common use of the BBC micro that attention is readily drawn to the more specialist applications. These include both the simplification of editing techniques, for instance, to cater for Special Needs, as well as those packages (usually ROMs) that reproduce a non-standard character set such as Hebrew, Russian or one with a high concentration of scientific symbols (see also the review of Word Power in BEEBUG Vol.6 No.8).

Product Supplier	Vue-Scientific Intelligent Machines Ltd. 66 Browning Road, Bushwood Estate, London E11 3AR.
Prices	£37.50 (Draft Quality) £49.95 (Letter Quality)

Vue-Scientific is a well produced, user-friendly text processing ROM that will reproduce Greek, scientific and mathematical characters on the screen in either mode 0 or 4. It also allows normal script (like this) to appear at the same time. Once you have edited, formatted and previewed your text, you can print it out. This will either be in draft mode - which is perfectly good, as in the illustration below - or near letter quality, but you must have an Epson (or compatible) printer.

I found the variety of normal word-processing commands available quite adequate. Wordwise users will feel at home too. Markers, Find and Replace, word-counts etc. are all possible, using the function keys. Superscript and subscript are easy to obtain as well. This good package (with equally good documentation) should be a boon to anyone in a secondary school, or a college or university maths or science department, who needs to produce documents, essays or lecture notes using these symbols.

The chemical equations are composed as follows:



The Navier-Stokes differential equations are:

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = -\frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial y^2} \right) + X$$

Product Supplier	'Prompt' and 'Writer' Micro Electronics Support Unit Unit 6, Advanced Technology Building, Sir William Lyons Road, Science Park, University of Warwick, Coventry CV4 7EZ.
Price	£3.00 (inc VAT) for one pack containing both.

At that price, how could you fail? MESU's best efforts have gone into upgrading these two excellent starter word-processors, Prompt being just that little bit easier to use than Writer. The use of menus and colour is admirable, the display a model of clarity and the documentation outstanding. True, these are aimed at Special Needs, and those involved in the development of the package are experts in that field, yet there is a versatility and flexibility that should make this suite a standard for children in infant, special schools and adult literacy classes. It includes utilities, support materials (such as A4 and A3 Concept Keyboard overlays and support of Votrax and Namal Speech input devices) as well as useful suggestions in the manual.

Secondly, there are those programs designed to sponsor some activity where language is important. One of the most influential, and still most popular classics, is Podd (from A.S.K. - take a look if you've not already seen it). Its success lies in the educational theory behind it.

Instead of expecting children to respond in one way only to a given stimulus or cue, it encourages child-centred exploration, recognizing that there are as many ways of reacting to a situation as there are children! When all is said and done, the major pedagogical texts agree that language is concerned with meaning. Chopping it up into artificial components cannot work: it needs to be used. Such open-endedness is also the strength of the latest offering from Topologika, "Giant Killer".

Product Giant Killer
Supplier Topologika
 P.O. Box 39,
 Stilton,
 Peterborough PE7 3RL.
Price £16 (B, B+ and Master)
 £18 (Compact) excl. VAT

This is the very best of its kind. Although most of the activities are mathematical ones, it is the co-operation and discussion by a group of children that actively enhances the quality of work being done. Here, a superior and very well thought out adventure (loosely based on the infamous beanstalk) encourages progress through all sorts of puzzling and worthwhile locations and situations. Maps are particularly important, and there are exemplary worksheets, children's and teachers' notes to go with it. If you haven't used this type of program with pupils before, this is as good a place to start as any.

Many teachers and educational users have long believed that almost any activity that promotes focused discussion will lead to enhanced learning, since the accommodation and assimilation of new ideas can be accelerated by articulating them. For this reason, you may judge how much the software that you use actually promotes discussion on and off the computer.

Thirdly there are those programs that aim to teach a particular aspect of language such as spelling or grammar. These are the least satisfactory in my opinion. This is because they often rely on memory for their success, or employ methods of reinforcement based on the

now surpassed psychology of stimulus-response. They frequently fail to distinguish at all between spoken and written language, which is a distinction that all specialists assert to be vital.

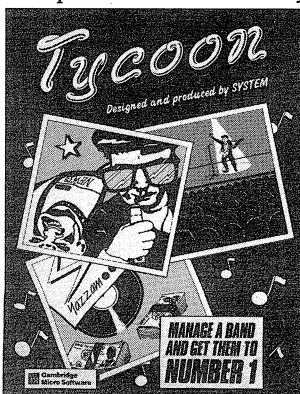
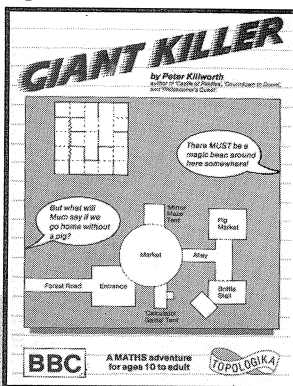
Lastly there are programs written to allow practice in (mostly) European modern languages. Unless they can claim to surround the user with a consistent "micro-world" or predictable environment of Spanish, German or whatever, there is the danger that there will be continual reference back to English.

I stand by the contention that only when the user is fully in control and able to determine where a program is going, can using the computer be superior to not using it. I'll end on a positive note with a superb example of the second activity identified earlier. Let your imagination go; fulfill your wildest dreams; spend time off the machine planning. Manage and run a pop-group!

Product Tycoon
Supplier Cambridge Micro Software
 The Edinburgh Building,
 Shaftsbury Road,
 Cambridge CB2 3RU.
Price £27.50 (excl. VAT) for B, B+ or
 Master with double disc drive

This too, is exemplary. The game is of the adventure type - with a healthy mix of simulation. You choose a personality for yourself and then a job; guide the group you have chosen through the hurly-burly of the entertainment world. If you are lucky, you'll meet and use lots of ideas and different registers of vocabulary on the way. The aims of the program are quite clearly set out in the front of the attractive notes that go with these discs. The subject-matter will be appealing to many pupils of 14 and upwards, at whom the package is aimed, and because that real environment in which things behave as you would expect is achieved from the outset, taking part with all the attendant decisions is pleasurable. There is no drilling, no wrong guesses and no expected answers. As a result, it works.

B



INSTANT LISTING OF FUNCTIONS AND PROCEDURES (continued from page 10)

```

2050 LDA line:SBC tablelo,X
2060 TAY:LDA line+1:SBC tablehi,X
2070 BCC buffer:STA line+1:STY line
2080 INC result:BNE con2
2090 .buffer
2100 STX temp:LDA result
2110 ORA #&30
2120 TAY:LDX #&00:LDA #&8A
2130 JSR osbyte:LDX temp
2140 DEX:BPL con1:RTS
2150 :
2160 .adjustpointer
2170 LDA pointer:CLC
2180 ADC length:STA pointer
2190 BCC checkend:INC pointer+1
2200 .checkend
2210 LDY #&01:LDA (pointer),Y
2220 CMP #&FF:RTS
2230 :
2240 .checkfortoken
2250 LDY #&03:LDA (pointer),Y
2260 STA length
2270 .loop
2280 INY:LDA (pointer),Y
2290 CMP #ASC" ":BEQ loop
2300 CMP #&DD:RTS
2310 :
2320 .tablelo
2330 EQU &E8640A01

```

```

2340 EQU &10
2350 .tablehi
2360 EQU &03000000
2370 EQU &27
2380 ] .
2390 NEXT
2400 ENDPROC
2410 :
2420 DEF PROCsavecode
2430 PRINT"Save code ? (Y/N)"
2440 IF FNyn THEN OSCLI"SAVE NEWLIST 9
00 "+STR$~P%
2450 CALL &900
2460 ENDPROC
2470 :
2480 DEF PROCcheckcode
2490 T%=0
2500 FOR N%=&900 TO P%-1
2510 T%=T%+?N%:NEXT
2520 IF T%<>&AB53 PRINT"Checksum Error.
...Re-check listing":END
2530 ENDPROC
2540 :
2550 DEF FNyn
2560 *FX21,0
2570 REPEAT:a$=GET$
2580 UNTIL INSTR("YyNn",a$)
2590 IFa$="N"Or a$="n" THEN=FALSE
2600 =TRUE

```

B

A COCKTAIL OF 3D PROCEDURES (continued from page 18)

```

300 PROCmove(150,160,0):FOR I=0 TO 2*P
I+0.4 STEP 0.4:PROCdraw(150,160*COS(I),1
60*SIN(I)):NEXT
310 REM UMBRELLA
320 GCOL0,2:VDU19,2,5,0,0,0:PROCmove(-
100,0,0):PROCdraw(500,250,0):FORI=0TO2*P
I STEP 0.4:PROCmove(500,250,0):PROCdraw(
300,200+200*COS(I),200*SIN(I)):NEXT
330 END
340 :
1000 DEF PROCinitrotation(X,Y,Z)
1010 LOCALa,b,c
1020 a=RADZ:p=COS(a):q=SIN(a):r=-q:s=p
1030 b=RADY:t=COS(b):u=SIN(b):v=-u:w=t
1040 c=RADX:k=COS(c):m=SIN(c):n=-m:o=k
1050 ENDPROC
1060 :
1070 DEF PROCrotate
1080 LOCALxs,ys
1090 xs=x
1100 x=(xs*p)+(y*r):y=(xs*q)+(y*s)
1110 xs=x
1120 x=(xs*t)+(z*v):z=(xs*u)+(z*w)
1130 ys=y

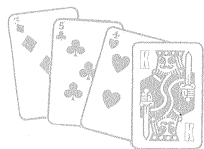
```

```

1140 y=(ys*k)+(z*n):z=(ys*m)+(z*o)
1150 PROCperspective(S)
1160 x=x+640:y=y+512
1170 ENDPROC
1180 :
1190 DEF PROCperspective(S)
1200 z=z+1000
1210 x=(S*x)/z
1220 y=(S*y)/z
1230 ENDPROC
1240 :
1250 DEF PROCmove(x,y,z)
1260 PROCrotate:MOVEx,y:ENDPROC
1270 :
1280 DEF PROCdraw(x,y,z)
1290 PROCrotate:DRAWx,y:ENDPROC
1300 :
2000 IF ERR=17 END ELSE REPORT:PRINT" a
t line ";ERL
2010 IF ERR=18 PRINT'"Probably in PROCp
erspective, if all else fails remove the
call to this procedure at line 1150."
2020 END

```

B



PATIENCE

Peter Coaker has produced a cracking good implementation of card Patience in one of its oldest and most popular forms. But be careful, it's surprisingly addictive.

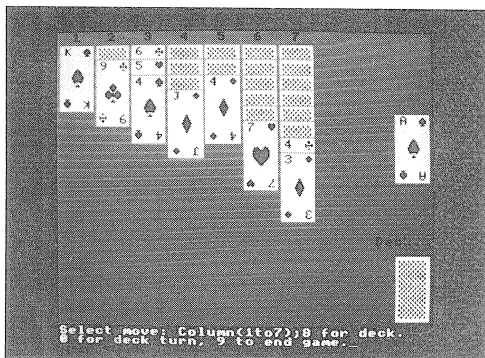
The game has been split into two separate programs so that it will work on the model B as well as on the Master. Unfortunately the listing still needs to be rather compact, so try not to insert too many spaces when typing it in. Type in the two listings and save them separately to tape or disc. Be sure to call the second program 'PAT2' so that it will be chained correctly by the first. When both listings have been entered and saved, run the first program. If PAGE is above &E00 you will have to wait while the entire program is moved down automatically to &E00. The cards will then be shuffled ready to start.

Initially the cards are laid out in seven columns, the number of cards in each column increasing from one to seven. Only the first card in each column is exposed, the others are laid face down. The exposed cards are laid on the first 'face down' card in the row above.

The ultimate object of the game is to assemble four stacks, one for each suit, with all the cards in order from the Ace upwards. Each stack must therefore be started by putting out any exposed Ace. Subsequent cards of the same suit, in ascending order, are put out as they come to the bottom of the columns. To expose the hidden cards, cards may be moved from one column to another, provided that they are in decending order, with red cards on black, and black on red. Groups of cards, in black-red-black sequence and in decending order, can be moved in blocks from one column to another.

When no further moves can be made, the deck of 24 remaining cards may be used. The deck is placed face down on the table. Three cards are

turned over at a time, the third being exposed. If this card can be put out, either to the stacks or to the columns, this should be done. Rules of sequence and colour still apply. Once the exposed card on the pack has been removed, the card that was underneath it and now exposed may be used. If this card cannot be used, another three cards may be turned over and the third card exposed in the normal manner. When the deck has been turned completely, the bottom card is put on the top of the deck and the process repeated until no further moves can be made in three cycles of the deck.



The rules may seem a little daunting when written down but, as ever, the easiest way of becoming familiar with the game is to play it. All moves are checked for validity, aces are automatically sent to stacks, stacks are removed from the table when the King is reached, and the computer will detect when it is not possible to finish the game.

The cards are easily moved around the table using single key presses in most cases. The computer will ask you the column from which cards are to be moved, the number of cards to be moved, and the destination column. You will also be given the option of shuffling the pack and using the exposed card. Everything else is completely automated.

This is a good old game of Patience, very well implemented on the micro, and you are sure to be impressed at just how often you go and play it. Why, next time you might well win.

Listing 1

```

10 REM Program Patience Loader
20 REM Version B1.09
30 REM Author P.B. Coaker
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 PROCchar
110 PROCassemble
120 CHAIN"PAT2"
130 END
140 :
1000 DEFPROCassemble:FORI%=0TO2 STEP2:P
%=&A00
1010 [OPTI%
1020 PHA:LDA#10:LDX#&70:LDY#0:JSR&FFF1
1030 .start:PLA:BNE rotate:JMP print
1040 .rotate:PHA:LDY#0
1050 .loop1:LDX#8:LDA#0
1060 .loop2:ASL A:CLC:ASL&70,X:BCC loop
3:ADC#0
1070 .loop3:DEX:BNE loop2:STA&79,Y:INY:
CPY #8:BNE loop1:LDX#0
1080 .loop4:LDA&79,X:STA&71,X:INX:CPX#8
:BNE loop4:PLA:SEC:SBC #1:PHA:JMP start
1090 .print:LDA #23:JSR &FFEE:LDA #254:
JSR &FFEE:LDX#0
1100 .repeat:LDA &71,X:JSR &FFEE:INX:CP
X #8:BNE repeat:LDA #254:JSR &FFEE:RTS
1110 ]
1120 NEXT:ENDPROC
1130 DEFPROCchar:VDU23,230,204,204,51,5
1,204,204,51,51:VDU23,231,8,28,62,127,62
,28,8,0:VDU23,232,54,127,127,127,62,28,8
,0:VDU23,233,8,28,28,107,127,107,8,28:VD
U23,234,8,28,62,127,127,28,62
1140 VDU23,235,1,3,3,7,7,15,15,31:VDU23
,236,0,128,128,192,192,224,224,240:VDU23
,237,31,15,15,7,7,3,3,1:VDU23,238,240,22
4,224,192,192,128,128,0
1150 VDU23,239,24,124,126,255,255,255,2
55,255:VDU23,240,48,124,252,254,254,254,
254,254:VDU23,241,255,127,127,63,31,15,7
,3:VDU23,242,254,252,252,248,240,224,192
,128
1160 VDU23,243,3,7,15,15,15,7,59,125:VD
U23,244,128,192,224,224,224,192,184,124:
VDU23,245,255,255,255,125,57,1,3,7:VDU23
,246,254,254,254,124,56,0,128,192
1170 VDU23,247,1,3,7,15,15,31,31,63:VDU
23,248,0,128,192,224,224,240,240,248:VDU
23,249,63,63,31,31,13,1,3,7:VDU23,250,24
8,248,240,240,96,0,128,192
1180 ENDPROC

```

* * * * *

Listing 2

```

10 REM Program Patience
20 REM Version B1.05
30 REM Author P.B. Coaker
40 REM BEEBUG March 1988
50 REM Program subject to copyright
60 :
100 MODE 7:PROCOFF
110 IF PAGE>&E00 GOTO 1980
120 ON ERROR MODE 7:REPORT:PRINT " at
line ";ERL:END
130 DIM P 52:DIM CL 164:DIM S 4:DIM CN
8:DIM L 8:DIM D 24:SH=FALSE
140 REPEAT
150 A$="A23456789TJQK":B$="0123456789"
:E$=STRING$(40,"*"):E$="":G%=0:NN%=21:U=
FALSE
160 FORI%=1 TO 52:P?I%=I%:NEXT:PROCshu
ffle
170 MODE1:PROCOFF:PROCTable:PROCLayout
180 FORC%=1TO7:PROCace(C%):NEXT
190 REPEAT:PROCselect:IF F%=9 OR (M%=0
ANDF%>0) GOTO230
200 IFF%=0 PROCover:GOTO230
210 PROCmove
220 IFNN%=0 U=TRUE:PROCAuto:GOTO230
230 UNTIL F%=9ORNO%=52
240 IFNO%=52 E$=" Well done! You moved
out all the cards."
250 MODE7:PRINTTAB((40-LENE$)/2,1):E$:
FORI%=1TO2:PRINTTAB(14,2+I%):CHR$141;"RE
SULTS.":NEXT
260 PRINTTAB(6,6);"Cards still in deck
: ";FNp(ND%&CN?8);TAB(6,7);"Cards put to
stacks : ";FNp(NO%);TAB(9,8);"Stacks com
pleted : ";FNp(NB%)
270 PRINTTAB(1,12);"Do you want anothe
r game? ";UNTIL FNno
280 CLS
290 END
300 :
1000 DEFPROCTable:VDU28,0,31,39,30:VDU2
4,0,70;1279;1023;VDU19,2,2,0,0,0:GCOL0,
130:CLG:CLS:GCOL0,0:VDU5:PROCOFF:FOR I%=
1 TO 7:MOVE125*I%-71,1017:PRINTSTR$(I%):
NEXT:MOVE1083,361:PRINT"Deck.":GCOL0,2:V
DU4:ENDPROC
1010 DEFPROCcard(C%,R%,V%):IF C%>0 AND
C%<8 AND R%<1 ENDPROC
1020 IFC%=0 X%=1155+125*(R%>2)ELSE IF C
%=8 X%=1030:Y%=85 ELSE X%=125*C%-110
1030 IFC%=0 Y%=539+(R%-1)MOD2*227 ELSE
IF C%=8 Y%=85ELSE Y%=812-50*R%
1040 IFV%>128 V%=V%-256
1050 MOVEX%-5,Y%-5:DRAWX%-5,Y%+217:PLOT
85,X%+115,Y%+217:DRAWX%+115,Y%-5:PLOT85,
X%-5,Y%-5:IFV%=0 THENENDPROC
1060 GCOL0,3:MOVE X%,Y%:DRAWX%,Y%+212:P
LOT85,X%+110,Y%+212:DRAWX%+110,Y%:PLOT85

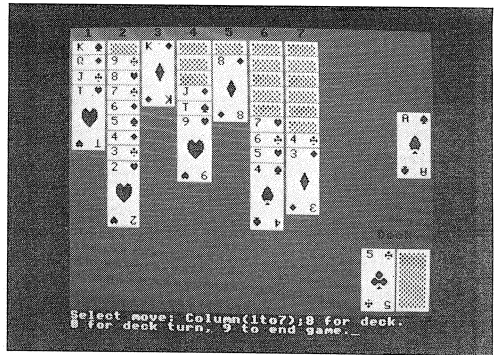
```



```

,X%,Y%:IFV%<0 THENPROCpattern:ENDPROC
1070 GCOL 0,-((V%-1)DIV13<=1)
1080 W%=V%MOD13:IFW%=0 W%=13
1090 VDU5:PROCOFF
1100 t=(V%-1)DIV13:MOVEX%+10,Y%+208:PRI
NTMID$(A$,W%,1):MOVEX%+76,Y%+208:PRINTCH
R$(231+t):MOVEX%+31,Y%+138:PRINTCHR$(235
+4*(t));CHR$(236+4*(t)):MOVEX%+31,Y%+106
:PRINTCHR$(237+4*(t));CHR$(238+4*(t))
1110 A%=2:MOVEX%+10,Y%+36:??&70=231+(V%-
1)DIV13:CALL &A00:MOVEX%+76,Y%+36:??&70=A
SCMID$(A$,W%,1):CALL &A00:GCOL0,2:VDU4:E
NDPROC
1120 DEFPROClayout:K%=0:FORR%=1TO7:FORC
%=R%TO7:K%=K%+1:V%=P?K%:IFC%>R%THENV%=-V
%
1130 CL?(20*(C%-1)+R%)=V%:PROCcard(C%,R
%,V%):NEXT:CN?R%=R%:L?R%=0:NEXT:NS%=0:NO
%=0:NB%=0:ND%=24:FORI%=1TO24:D?(25-I%)=P
?(I%+28):NEXT:CN?8=0:PROCdeck:L?8=0:ENDP
ROC
1140 DEFPROCdeck:X%=1155:Y%=85:IFND%>0T
HENGCOL0,3
1150 MOVEX%,Y%:DRAWX%+110,Y%:PLOT85,X%+
110,Y%+212:DRAWX%,Y%+212:PLOT85,X%,Y%:IF
ND%>0THENPROCpattern
1160 IFCN?8>0 PROCcard(8,0,CL?(140+CN?8
)):ENDPROC ELSEPROCcard(8,0,0):ENDPROC
1170 DEFPROCpattern:VDU5:PROCOFF:GCOL0,
1:FORL%=1TO6:MOVEX%+8,Y%+10+32*L%:FORJ%=
1TO3:PRINTCHR$(230);:NEXT:PRINT:NEXT:GCO
L0,2:VDU4:ENDPROC
1180 DEFPROCace(C%):CH=FALSE:IFCN?C%=0
THENENDPROC ELSEV%=CL?(20*(C%-1)+CN?C%):
IFV%MOD13=1 THENNS%=NS%+1:PROCstack(NS%,
C%):CH=TRUE:V%=CL?(20*(C%-1)+CN?C%)
1190 IF CH AND CN?C%>0 PROCconset(C%):PR
OCace(C%):ENDPROC ELSE ENDPROC
1200 DEFPROCstack(N%,C%):VM%=CL?(20*(C%
-1)+CN?C%):CL?(20*(C%-1)+CN?C%)=0:PROCca
rd(C%,CN?C%-L?C%,0):CN?C%=CN?C%-1:n%=CN?
C%-L?C%:IFn%>0ANDn%<128THENV%=CL?(20*(C%
-1)+CN?C%):PROCcard(C%,n%,V%+256*(V%>127
))
1210 PROCcard(0,N%,VM%):S?N%=VM%
1220 IF (S?N%)MOD13=0 PROCbox(N%)
1230 IF CN?C%>0ANDC%<8 PROCTurn(C%)
1240 NO%=NO%+1:PROCace(C%):PROConset(C%
):ENDPROC
1250 DEFPROCbox(N%):PROCcard(0,N%,-1):I
FN%<NS% THENFORJ%=N%TONS%-1:S?J%=S?(J%+1
):PROCcard(0,J%+1,0):PROCcard(0,J%,S?J%)
:NEXT
1260 S?NS%=0:PROCcard(0,NS%,0):NS%=NS%-
1:NB%=NB%+1:ENDPROC
1270 DEFPROCTurn(C%):IF CN?C%=0 OR C%=8
ENDPROC
1280 V%=CL?(20*(C%-1)+CN?C%):IFV%>127 N
N%=NN%-1:V%=256-V%:CL?(20*(C%-1)+CN?C%)=

```



```

V%:PROCcard(C%,CN?C%-L?C%,V%)
1290 ENDPROC
1300 DEFPROCany:VDU7:COLOUR1:PRINT" Hit
any key.";Z$=GET$:COLOUR3:CLS:ENDPROC
1310 DEFPROCselect
1320 PRINT"Select move: Column(1to7);8
for deck.""0 for deck turn, 9 to end ga
me.";REPEAT:F%=INSTR(B$,GET$):UNTILF%>0
:F%=F%-1:PRINTSTR$(F%):IF F%=9 PRINT"Are
you sure you wish to stop! ":IF FNno G
OTO 1320
1330 IFF%=0 ENDPROC
1340 IFF%=9 E$=" Game abandoned":ENDPR
OC
1350 IFF%=8 AND CN?8=0 PRINT"No cards e
xposed in deck.":PROCany:M%=0:ENDPROC
1360 IFCN?F%=0 PRINT"No cards in column
";STR$(F%):PROCany:M%=0:ENDPROC
1370 PROCbottom(1):IF F%=8 M%=1:GOTO140
0
1380 PRINT"Number of cards to move ";:I
NPUT,M%:IFM%>CN?F%THENPRINT"Only ";STR$(
CN?F%);" cards in column ";STR$(F%):PRO
Cany:PROCbottom(3):M%=0
1390 IFM%=0 ENDPROC
1400 PROCTop(1):PROCcheck1:ENDPROC
1410 DEFPROCbottom(c%):VDU5:PROCOFF:GCO
L0,c%:IFF%=8THENX%=1069:Y%=117 ELSEX%=12
5:F%-63:Y%=844-50*(CN?F%-L?F%)
1420 MOVEX%,Y%:PRINTCHR$(124):VDU4:GCOL
0,2:ENDPROC
1430 DEFPROCTop(c%):VDU5:PROCOFF:GCOL0,
c%:IFF%=8THENX%=1069:Y%=297 ELSEX%=125*F
%-63:Y%=974-50*(CN?F%-L?F%-M%)
1440 MOVEX%,Y%:PRINTCHR$(124):VDU4:GCOL
0,2:ENDPROC
1450 DEFPROCcheck1:IFM%=1THENENDPROC
1460 I%=0:REPEAT:I%=I%+1:J%=CL?(20*(F%-
1)+CN?F%-I%):UNTILI%=M%-10RJ%>128
1470 IF J%>128 PRINT"You cannot move tu
rned cards.":PROCany ELSE ENDPROC
1480 PROCbottom(3):PROCTop(3):M%=0:ENDP
ROC

```

```

1490 DEFPROCmove:PRINT"Move cards where
? Enter column (1-7)""0 to stack; 8 or
9 to abort move.";:REPEAT:T%=INSTR(B$,GE
T$):UNTIL T%>0:T%=T%-1:PRINTSTR$(T%)
1500 IFT%>7 PROCbottom(3):PROCTop(3):M%
=0:ENDPROC
1510 IFT%=0 PROCcheck2 ELSE PROCcheck3
1520 IFM%=0 ENDPROC
1530 IFF%=8 G%=0
1540 IFT%=0 PROCstack(N%,F%):ENDPROC
1550 FOR I%=1 TO M:N%=CN?F%+1-I%:PROCC
ard(F%,N%-L?F%,0):IFN%-L?F%>1 AND N%-L?F
%<128 PROCcard(F%,N%-L?F%-1,CL?(20*(F%-1
)+N%-1))
1560 NF%=20*(F%-1)+N%:NT%=20*(T%-1)+CN?
T%+M%+1-I%:CL?NT%=CL?NF%:CL?NF%=0:NEXT:F
ORI%=1TOM%:PROCCard(T%,CN?T%-L?T%+I%,CL?
(20*(T%-1)+CN?T%+I%)):NEXT:CN?T%=CN?T%+M
%:CN?F%=CN?F%-M%:PROCTurn(F%):PROCace(F%
):PROCOffset(T%):PROConset(F%):ENDPROC
1570 DEFPROCcheck2:IFM%>1THENPRINT"Only
one card may be stacked at a time!":PRO
Cany:M%=1
1580 N%=0:REPEAT:N%=N%+1:V%=CL?(20*(F%-
1)+CN?F%):UNTILV%-1=S?N%ORN%-NS%:IFV%-1=
S?N%THENENDPROC
1590 IF NOT U PRINT"No stack can receiv
e this card.":PROCany
1600 PROCbottom(3):PROCTop(3):M%=0:ENDP
ROC
1610 DEFPROCcheck3:V%=CL?(20*(F%-1)+CN?
F%-M%+1):IFCN?T%=0ANDV%MOD13=0 THEN ENDP
ROC ELSE IFCN?T%=0PRINT"Only Kings may b
e put in blank columns.":PROCany:PROCBot
tom(3):PROCTop(3):M%=0:ENDPROC
1620 J%=CL?(20*(T%-1)+CN?T%):IF (V%-1)MO
D13=(J%-1)MOD13-1 ANDABS((V%-1)DIV26-(J%
-1)DIV26)=1 ENDPROC
1630 IF (V%-1)MOD13<>(J%-1)MOD13-1 PRINT
"Column ";STR$(T%);" not in numerical or
der.":PROCany:PROCbottom(3):PROCTop(3):M
%=0:ENDPROC
1640 PRINT"Column ";STR$(T%);" not in "
";:COLOUR131:FORI%=1TO2:COLOUR1:PRINT"R";
:COLOUR0:PRINT"B";:NEXT:COLOUR128:COLOUR
3:PRINT" sequence.":PROCany:PROCbottom(3
):PROCTop(3):M%=0:ENDPROC
1650 DEFPROCoffset(C%):IFC%=8 THENENDPR
OC
1660 IF CN?C%-L?C%<18 ENDPROC ELSE L?C%
=CN?C%-12:PROCCol(C%):ENDPROC
1670 DEFPROCConset(C%):IF L?C%=0 OR CN?C
%=0 THEN ENDPROC
1680 IFCN?C%-L?C%>10 ENDPROC
1690 N%=CN?C%-12:IFN%<0 THENN%=0
1700 L?C%=N%:PROCCol(C%):ENDPROC
1710 DEFPROCcol(C%):MOVE125*C%-117,70:D
RAW125*C%+8,70:PLOT85,125*C%+8,988:DRAW1
25*C%-117,988:PLOT85,125*C%-117,70:IFCN?
C%=0 ENDPROC ELSEFORI%=1TOCN?C%-L?C%:V%=

```

```

CL?(20*(C%-1)+I%+L?C%):IFV%>128THENV%=V%
-256
1720 PROCcard(C%,I%,V%):NEXT:IFL?C%=0TH
ENPROCmark(C%,2):ENDPROC ELSE PROCmark(C
%,1):ENDPROC
1730 DEFPROCmark(C%,c%):VDU5:GCOLOR,c%:M
OVE125*C%-107,1017:PRINT"*":MOVE125*C%-3
5,1017:PRINT"*":GCOLOR,2:VDU4:ENDPROC
1740 DEFFNno:REPEAT:VDU7:Z=INSTR("YyNn"
,GET$):UNTILZ>0:IF Z>2 THEN PRINT"No.":=
TRUE ELSEPRINT"Yes.":=FALSE
1750 DEFFNp(N%):=RIGHT$(STRING$(3,CHR$(3
2))+STR$(N%),3)
1760 DEFPROCcover:IF ND%+CN?8=0 PRINT"No
cards left in deck.":PROCany:ENDPROC
1770 IF ND%=0 T%=CL?141:CL?141=0:FOR I%
=2 TO CN?8:D?I%=CL?(142+CN?8-I%):CL?(142
+CN?8-I%)=0:NEXT:D?1=T%:ND%=CN?8:CN?8=0:
PROCdeck:G%=G%+1:IFG%=4 E$=" No progress
after turning pack.":F%=9:ENDPROC
1780 IFND%>3 NT%=3 ELSE NT%=ND%
1790 FOR I%=1 TO NT%:CL?(141+CN?8)=D?ND
%:D?ND%=0:ND%=ND%-1:CN?8=CN?8+1:PROCdeck
:NEXT:PROCace(8):ENDPROC
1800 DEFPROCauto:COLOUR1:PRINT"AUTOMATI
C FINISH.":COLOUR3:VDU7,7,7
1810 I%=0:CH=FALSE:REPEAT:I%=I%+1:IFCN?
I%=0 GOTO1840
1820 M%=1:F%=I%:T%=0:PROCcheck2:IFM%=0
GOTO1840
1830 PROCstack(N%,F%):G%=0:CH=TRUE
1840 UNTILI%=7
1850 IF CH GOTO 1810
1860 IFND%+CN?8=0 ENDPROC ELSEIFCN?8=0
GOTO1890
1870 M%=1:F%=8:T%=0:PROCcheck2:IFM%=0 G
OTO1890
1880 PROCstack(N%,8):G%=0:GOTO1810
1890 PROCover:IF CH GOTO 1850 ELSE1870
1900 DEFPROCshuffle:IFSH THENCLS:PRINT"
Is pack to be shuffled? ":IFFNno ENDPRO
C
1910 CLS:FORI%=1TO2:PRINTTAB(9,10+I%);C
HR$129;CHR$141;CHR$136;"SHUFFLING.":NEXT
:X%=RND(-TIME)
1920 FORI%=52TO2 STEP-1:X%=RND(I%):IFX%
<I%THENT%=P?X%:FORJ%=X%TOI%-1:P?J%=P?(J%
+1):NEXT:P?I%=T%
1930 NEXT:SH=TRUE:CLS:ENDPROC
1940 DEF PROCoff
1950 VDU 23,1,0;0;0;0;
1960 ENDPROC
1970 REM Relocation code
1980 CLS:PRINTTAB(5,10);"Relocating pro
gram.":VDU21
1990 T%=HIMEM:*KEY0*TAPE|MD%=PAGE-&E00:
FORI%=PAGE TOT% STEP4:!(I%-D%)=I%:NEXT:
?(T%-D%)=255:PAGE=&E00|MOLD|MVDU6:RUN|M
2000 *FX138,0,128

```

B

HINTS HINTS HINTS HINTS HINTS

and tips and tips and tips and tips and tips

INVISIBLE CALCULATIONS

Ian Bishop

When arranging large spreadsheets with Computer Concepts' Intersheet, it is often convenient to hide intermediate calculations so that they are not displayed with the rest of the results.

To hide such calculations, use the hold facility (/H) to place blank rows or columns over them. It is then possible to reduce the column width to three characters. This will replace the intermediate calculations with blank rows or columns. Remember to answer YES to the question 'Print held lines?' when the sheet is to be printed.

COLOUR VIEW

Graeme Davidson

Although the original View guide explains how to redefine the foreground and background colours in View (all modes except 7), no mention is made of it in the documentation supplied with the Master, and in any case the technique is often overlooked. The foreground and background colours can only be changed from within Command mode. The format of the command is as follows:

Ctrl-S followed by ab000 where 'a' represents a logical colour in the selected mode (foreground or background). For example, in the commonly used mode 3, 'a' would be 0 to change the background, 1 to change the foreground. The

letter 'b' represents the physical colour that you want. A complete list of physical colours can be found in your User Guide. For example, to change the background in mode 3 to blue (colour 4) type: Ctrl-S 04000.

BASIC PRIORITY ON THE B+

Neil Stephens

On the BBC model B machines and the Master series the language which the machine enters upon a Reset or power-up can be easily defined, either by physically changing the order in which the ROMs are plugged in for the former, or by using the *CONFIGURE LANGUAGE command with the latter. Unfortunately neither of these two methods can be used on the BBC B+ because the *CONFIGURE command is not implemented, while Basic and the Operating System are both contained in the same chip.

Fortunately Acorn have provided a link to alter the logical ROM position which Basic occupies. Altering link S13 into the North position will make Basic resume the lowest priority position. The first language ROM encountered in a high priority socket will then be entered upon power-up.

POWER-UP CONFIGURATION

Brian Stein

Towards the righthand front corner of the keyboard PCB there is provision for an eight-

way DIP switch to be fitted. If the switch has not already been installed, one can be obtained from most electronic shops (inc. BEEBUG) and soldered in relatively easily. This switch determines various options set upon power-up or after a Break.

As you look at the bank of switches they are numbered 0 to 7 from right to left. Switches 0, 1 and 2 determine the screen mode, and this is represented by an inverted binary value between 0 and 7 (i.e. 101 would select screen mode 2). A switch in the up position represents 1, in the down position 0. Switch 3 reverses the effect of pressing Break and Shift-Break. In this condition the computer will boot the disc in the drive upon power-up. Switches 4 and 5 effect the disc drive speeds. These settings were covered in last month's Hints and Tips page. The last two switches, 6 and 7, are not used.

DOUBLE STEPPING

Micky Harford

For those people using 1770 disc interfaces, the *DRIVE command has been extended so that forty or eighty tracks may be specified. This means that forty track discs may be read on eighty track drives without the use of a hardware switch. The format of the command is:

*DRIVE n t

where n is the drive number and t is 40 or 80.

B

VIDEO CASSETTE CATALOGUER (continued from page 23)

```

3210 PROChead(11,mg$+"Delete"):PRINT'"N
o. of entries on card ";a$;" = ";ent
3220 PRINT'"Do you wish to : "
3230 PRINT'"1. Delete tape number."
3240 PRINT'"2. Delete all titles on a ta
pe.":*FX15,0
3250 IF ent>1 PRINT'"3. Delete a select
ed title on a tape."
3260 PRINT'"Action : ";:del=GET-48:IF(
ent>1 AND(del<1 ORdel>3)) OR(ent<2 AND(d
el<1 ORdel>2)) THEN3210
3270 IF del=3 ent=1
3280 FOR A%=1 TO Z%:IFLEFT$(N$(A%),3)=a
$ PROCprint:ELSE3230
3290 ENT=ENT+1
3300 IF ENT<>ent AND del<3 PROCspace(2)
:GOTO3320
3310 PROCdel
3320 NEXT
3330 IF F%=1 ORdel<3 ENDPROC
3340 FOR A%=1 TO Z%:IFLEFT$(N$(A%),3)=a
$ FO=A%
3350 NEXT:A%=FO:PROCleft:ENDPROC
3360 :
3370 DEFPROCdown
3380 FOR G%=1 TO Z%-A%:T$(G%+A%-1)=T$(G
%+A%):N$(G%+A%-1)=N$(G%+A%):NEXT
3390 Z%=Z%-1:ENDPROC
3400 DEFPROCcard
3410 PRINTTAB(1,23)CHR$148+STRING$(32,C
HR$163)TAB(1,9)CHR$148+STRING$(21,CHR$24
0)
3420 X%=10:REPEAT
3430 VDU31,1,X%,148,53,135,31,32,X%,148
,234
3440 X%=X%+1:UNTILX%>22
3450 VDU31,22,9,62,31,22,8,148,62,31,33
,9,237,31,32,8,237,31,23,7,148:PRINTSTRI
NG$(8,CHR$240)
3460 VDU31,25,9,135,32,32,32,148
3470 PRINTTAB(4,10)cy$+"E-"TAB(4,11)"Ca
teory : "TAB(4,13)"Title : "TAB(5,18)"Dur
ation : "TAB(6,20)"Counter : "TAB(4,22)"Ti
me left : "
3480 ENDPROC
3490 DEFPROCclear(c,d,e,f)
3500 LOCALT
3510 FORT=d
TOe:PRINTTAB(c,T)SPC(f):NEXT:ENDPROC
3520 :
3530 DEFPROCdel
3540 IF del=1 f$="tape" ELSEf$="title"
3550 PRINTTAB(0,3)"Is this the ";f$;" t
o"TAB(0,4)"be deleted (y/n) ?":Y$=GET$
3560 IFINSTR("Nn",Y$)>0 ENDPROC

```

```

3570 PRINT"Sure ?":Y$=GET$:IFINSTR("Nn
",Y$)>0 ENDPROC
3580 fi=1:FO=1
3590 IFdel=2 fi=2:N$(A(1))=a$+STRING$(4
,CHR$32)+"-"+STRING$(4,CHR$32)+L$+"000"+
L$:T$(A(1))=STRING$(4,CHR$32):IFent=1 GO
TO3630
3600 FOR F=ent TO fi STEP-1
3610 IFdel<3 A%=A(F)
3620 PROCdown:NEXT:F%=0
3630 ENDPROC
3640 :
3650 DEFFNeditor(J)
3660 PRINTTAB(4,J+1)"Title : ";
3670 IF K=2 i$=LEFT$(t$+STRING$(80,CHR$
32),80) ELSEi$=STRING$(80,CHR$32)
3680 FOR X=1 TO 4:PRINTTAB(12,J+X)"<"SP
C20">"
3690 IF K=2 PRINTTAB(13,J+X)MID$(t$, (X-
1)*20+1,20)
3700 NEXT
3710 PRINTTAB(2,J+6)"CTRL-L = CLS"TAB(2
,J+7)"f0 = insert/over"
3720 X=1:Y=1:over=FALSE:tog$=fl$+yl$+"I
NSERT"+wh$+fo$
3730 REPEAT:PRINTTAB(3,J+5)tog$TAB(X+12
,Y+J);:*FX21,0
3740 A=GET:IFA=128 VDU7:over=NOTover
3750 IFover tog$=fl$+qr$+"OVER"+wh$+fo$
ELSEtog$=fl$+yl$+"INSERT"+wh$+fo$
3760 IFA=12 PROCclear(13,1+J,4+J,20):i$
=STRING$(80," ")
3770 IFA=136 X=X-1:IFX=0 ANDY>1 X=20:Y=
Y-1 ELSEIF X=0 ANDY=1 X=1
3780 IFA=137 X=X+1:IFX=21 ANDY<4 X=1:Y=
Y+1 ELSEIF X=21 ANDY=4 X=20
3790 IFA=139 ANDY>1 Y=Y-1
3800 IFA=138 ANDY<4 Y=Y+1
3810 IFA>31 ANDA<123 PROCadd:PROCpri
3820 IFA=127 PROCde:PROCpri
3830 UNTILA=13:IFI$=STRING$(80,CHR$32)
i$=""
3840 =i$
3850 :
3860 DEFPROCpri
3870 FOR X%=0 TO 3:PRINTTAB(13,X%+J+1)M
ID$(i$,X%*20+1,20):NEXT:ENDPROC
3880 :
3890 DEFPROCadd
3900 temp$=LEFT$(i$, (Y-1)*20+X-1):tem$=
RIGHT$(i$,80-(Y-1)*20-X+1+over):i$=LEFT$
(temp$+CHR$A+tem$,80):X=X+1:IFX=21 ANDY<
4 X=1:Y=Y+1 ELSEIF X=21 ANDY=4 X=20
3910 ENDPROC

```

B



POSTBAG



POSTBAG

MASTERING THE KEYWORD GENERATOR

Having recently upgraded from a B to a Master, I found that the Keyword Generator program (BEEBUG Vol.3 No.6) only produced garbage. I was able to determine that the following alteration to line 370 enabled the program to work as before. I trust this program may be of help to others who use this program to ease the tedium of inputting Basic programs.

```
370 LDA#&56:STA&80:LDA#&84:
    STA&81:LDY#0
```

B.P.Weller

We are always interested in publishing amendments to older BEEBUG programs to enable them to run on the Master and Compact.

SINGLE KEY AUTO-SAVE

Although I do not want to detract from the merits of the Auto-Save utility in BEEBUG Vol.6 No.2, I find the following single key press even more useful, particularly if I am called away from the machine in a hurry. The only rule is that the first line of the program under development should be a REM statement with one space followed only by a five letter name and a one digit number plus a space, or a two digit number.

The key (f0) then inserts the next number into line one of the program, saves it under

that name and number, and also saves a copy under the name LAST. In the !BOOT file of my development disc I arrange to LOAD"LAST" which immediately puts me back into business when I can get back to the computer. I think your readers would find this useful.

```
*K.0$(PA.+11)=STR$(VAL$(PA.+11)+1)|Msave$=LE.$(PA.+6),
7)|MSA.save$|MSA."LAST"|MP.
$(PA.+5)|M
```

Francis W.Aries

DISC MANAGER BUG

I have been using the disc Manager from Vol.5 No.4 for some time and have found it extremely useful. Unfortunately there is a bug in the Rename option if an attempt is made to rename a file on other than drive 0. The problem can be overcome by changing line 2670 to read:

```
2670PROCos("RENAME ":"+D$+"
"+name$(n)+" ":"+D$+"."+a$:na
me$(n)=a$:ENDPROC
```

D.G.Beecham

It was only intended that Rename should work with drive 0, so this is a useful improvement.

IMPROVED KEYSTRIPS

With reference to Tony Briselden's letter in Vol.6 No.5 I would like to suggest the following additional changes to the Keystrip Generator (Vol.6 No.4). This amendment prints the legends in double

strike (line 2145), and the keystrip title in double-width characters (lines 1825 and 1835) on an Epson compatible printer. Add the following lines:

```
1825VDU27,64,14,27,69,27,71
1835VDU27,106,12,27,64,27,
    51,24
2145VDU27,71
```

D.J.L.van den Bergh

MS-DOS TO BEEB

I would like to suggest an idea for a program to be published in BEEBUG, one which I think may be of general interest to BEEBUG readers. Many of us who work in offices and elsewhere have access to PCs or similar during business hours using discs formatted to MS-DOS standard. My plea is for a program which would enable my BBC B to read a PC disc so that wordprocessor text could be transferred to my Wordwise at home.

Similarly, after amending the text at home, I would like to re-write the file back to a disc in PC format for use back in the office.

John Tupper

We have investigated the possibility of this and it certainly seems feasible. Even now one of our freelance contributors is working away to produce just such a utility which we hope to publish in the April issue of BEEBUG or soon after.

B

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

£ 7.50	6 months (5 issues) UK only
£14.50	1 year (10 issues) UK, BFPO, Ch.I
£20.00	Rest of Europe & Eire
£25.00	Middle East
£27.00	Americas & Africa
£29.00	Elsewhere

BEEBUG & RISC USER

£23.00
£33.00
£40.00
£44.00
£48.00

BACK ISSUE PRICES

Volume	Magazine	Cassette	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	£3.50	-
3	£0.70	£1.50	£4.00	£4.50
4	£0.90	£2.00	£4.50	£4.75
5	£1.20	£2.50	£4.75	
6	£1.30	£3.00		

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.

FURTHER DISCOUNTS

We will allow you a further discount:

Five or more:	deduct £0.50 from total
Ten or more:	deduct £1.50 from total
Twenty or more:	deduct £3.50 from total
Thirty or more:	deduct £5.00 from total
Forty or more:	deduct £7.00 from total

POST AND PACKING

Please add the cost of p&p:

Destination	First Item	Second Item
UK, BFPO + Ch.I	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

BEEBUG
Dolphin Place, Holywell Hill, St.Albans,
Herts. AL1 1EX
Tel. St.Albans (0727) 40303
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Assistant: Lance Allison
Production Assistant: Yolanda Turuelo
Membership secretary: Mandy Milleham
Editorial Consultant: Lee Calcraft
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, BEEBUG 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 £40 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.

BEEBUG Ltd (c) 1988

Printed by Head Office Design (0782) 717161 ISSN - 0263 - 7561

Magazine Disc/Cassette

MARCH 1988 DISC/CASSETTE CONTENTS

INSTANT LISTING OF FUNCTIONS & PROCEDURES - speeds up the listing of named functions and procedures.

NOW C HERE - a complete program in C to start off our new introductory series.

COCKTAIL OF 3D PROCEDURES - two complete programs to demonstrate how to draw 3D objects.

VIDEO CASSETTE CATALOGUER - a database application to catalogue all your video cassettes.

BARRY CHRISTIE VISUALS

SPRITE ANIMATOR - a total of four programs to animate the sprites created last month.

FIRST COURSE

CHARACTER CONTROL - a program illustrating the use of Basic's string functions for flexible input.

THE MASTER SERIES

TALKING TO THE ADFS (Part 2) - two complete programs to provide a customised ADFS menu, and a utility to catalogue all your ADFS directories.

CIRCUIT ANALYSIS (Part 2) - not only a program to display the results of your circuit analysis, but a copy of the extended analysis program and a selection of different circuit files as examples.

EXPLORING ASSEMBLER (PART 8) - three examples of the use of the CMP instruction including look-up tables.

PATIENCE - a most effective and addictive implementation of one of the oldest and most popular forms of this card game.

MAGSCAN - Bibliography for this issue of BEEBUG.

All this for £3 (cassette), £4.75 (5" & 3.5" disc) + 50p p&p.

Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES
6 months (5 issues)
12 months (10 issues)

5" Disc
£25.50
£50.00

UK ONLY
3.5" Disc
£25.50
£50.00

Cassette
£17.00
£33.00

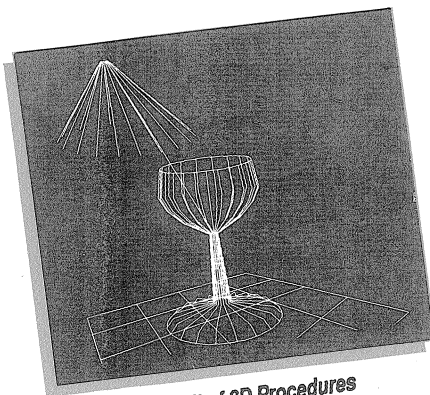
5" Disc
£30.00
£56.00

OVERSEAS
3.5" Disc
£30.00
£56.00

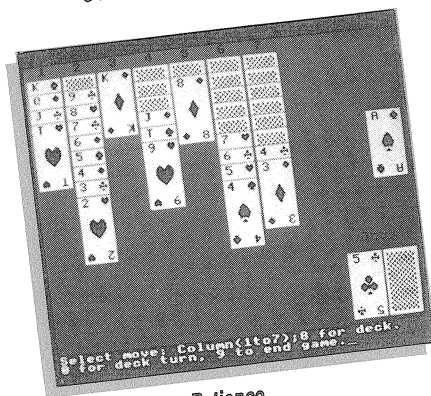
Cassette
£20.00
£39.00

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

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

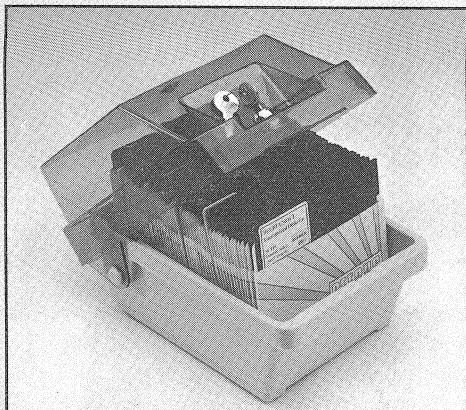


Cocktail of 3D Procedures



Patience

PRICES REDUCED!



50 Single Sided 40T **Only**
Double Density **£35.60**
Discs

50 Double Sided
80T Quad
Density Discs **Only**
£39.90

We confidently offer a lifetime data guarantee and will replace any disc with which you encounter problems. We have found that the standard of quality control at the factory makes this necessity very rare.

BEEBUG discs are manufactured to the highest specifications and are fully guaranteed.

	Price	Members Price	Order Code
10	£9.37	£8.90	0657
25	£23.00	£21.85	0661
50	£37.50	£35.60	0665

	Price	Members Price	Order Code
10	£10.42	£9.90	0660
25	£26.20	£24.90	0664
50	£42.00	£39.90	0668

I enclose a cheque for £ _____ /Please debit my Access/Visa card £ _____

[illegible]

Expiry date:

--	--

Name _____ Memb No. _____

Address _____

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

☎ (0727) 40303