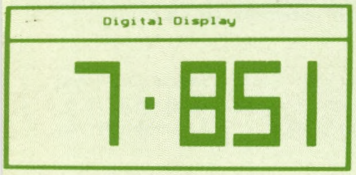
£1.00

BEEBUG

FOR THE BBC MICRO

Vol 2 No 8 JAN/FEB 1984



LARGE DIGITAL DISPLAY

DANCING LINES

MACHINE CODE GRAPHICS

BLOCK BLITZ

PLUS
* DISASSEMBLER
* PROTECTING PROGRAMS
* FORTH INTRODUCTION
* DOUBLE DENSITY DFS REVIEWS
* GRAPHICS TABLETS REVIEWED
* And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 20,000

EDITORIAL

THIS MONTH'S MAGAZINE

In this issue, we publish the first part of a new series introducing machine code graphics, which is the basis for the superb screen displays of many of the good commercial games now on the market. The series will introduce many of the techniques involved and will also show you how to include these routines in your Basic programs.

We are also very enthusiastic about Block Blitz, one of the very best games written in Basic that we have seen for some time. The effort of typing this into your micro is well worthwhile. We have also included a detailed review of the two double density disc controllers that have so far been released.

Apart from the many other excellent programs and articles this month, we have provided a short introduction to the computer language FORTH, as a prelude to a comparative review of FORTH implementations for the BBC micro, that we shall be publishing in the next issue. Your views on the extent to which we should cater in BEEBUG for languages other than Basic would certainly be of interest to us.

O.S. 1.2

May I just remind you, for one more time, that this is the last issue of the magazine for which we will be testing all programs with both O.S. Ø.1 and O.S. 1.2. From the next issue, we shall only be testing programs with O.S. 1.2, though we shall continue to test on Basics I & II.

SHOWS

We had a busy time before Christmas, as we not only produced two magazines (BEEBUG and ELBUG), but we were also represented at two shows, the Micro User Show and the Your Computer Christmas Fair, each over four days. If you wondered why our stand at the Micro User Show had the name Datel, it was because we could not get a stand when we called ourselves BEEBUG. This all seems rather silly. Anyway, we very much enjoyed meeting and talking to those of you who visited us at the shows. We shall also be at the Acorn Educational Exhibition in January (see Events in the Supplement).

NEXT ISSUE

This issue of BEEBUG covers both January and February 1984, and the next issue that you will receive will be for March 1984.

ADVERTISING SUPPLEMENT

As from the next issue all advertising in the supplement will be handled by Computer Marketplace Ltd. This will relieve a burden on our production team. If you are interested in booking space then give them a ring on Ø1-93Ø 1612.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAI

HINT WINNERS

This month's hint winners are D.Pooley who wins the £1Ø prize, and R.Duebel who wins the £5 prize.

MAGAZINE CASSETTE

In response to requests from a number of members, we have also included, on this month's magazine cassette, a copy of the machine code Compactor program by David Tall published in BEEBUG Vol. 1 No.9.

BEEBUG MAGAZINE

GENERAL CONTENTS

NEWS

New Educational Software from Acornsoft

Acornsoft have announced the release of a range of educational software developed by Applied Systems Knowledge, which is now being marketed by Acornsoft. Ten different software packages are available, covering concepts in numeracy and literacy, and designed for use with children from the age of 3 to 11 years. The programs are available now at £9.95 each and Electron versions will also be available shortly. [See review of FACEMAKER in this issue.]

Marriage Guidance by Computer

Acornsoft have released two new programs that are unusual, and perhaps controversial. Both programs make use of "personality questionnaires" developed by Professor Hans Eysenck and Dr.Glenn Wilson. "The dating game" enables individuals to examine their personalities and help them find partners that are likely to be compatible. "I do" helps couples to assess the good and bad points of their relationships, and pin-point danger areas which could lead to trouble. Both packages cost £12.65 inc VAT and will run on either the BBC micro or the Electron.

All Acornsoft programs are available through computer shops and dealers. For further information contact Acornsoft at 4A Market Hill, Cambs, CB2 3NJ.

Low Cost Printer Interface

For £32.25 (+£1 post) you can obtain an interface for your Beeb which allows your to use a Sinclair Printer, or £71.95 (+£2 post) for a package of the interface and printer together. Available from:
I.T.M. ltd., E3 New Enterprises, South West Brunswick Dock, Liverpool, L3 4AR.

Beaver Plotter

An A4 flat-bed plotter with Centronics interface, which accepts most types of pens costs £449 (plus VAT) from Linear Graphics Ltd., 34a Brook Road, Rayleigh Weir Industrial Estate, Rayleigh, Essex, SS6 7XN.

8" Drive DFS

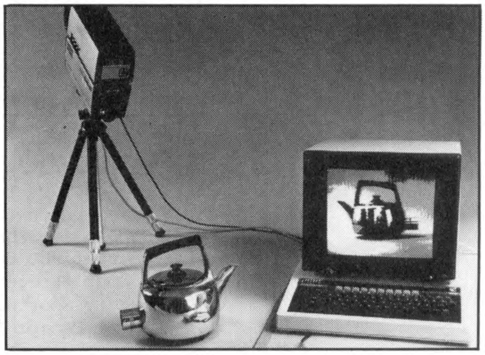
An 8" disc drive DFS is available from Vogan Products, "The White House", 21 Grove Road, Hazlemere, Bucks., HP15 7QY, for £44.95 (+£1.50 post) including manual. The DFS is supplied on EPROM and allows nearly 1.2 Mbytes of storage on the 8" drives, with 62 files per surface. It will even allow you to have up to 248 active files.

Residential Computer Courses

Horncastle Residential College, Mareham Road, Horncastle, Lincolnshire, LN9 6BW, is offering 2 short residential courses for anyone with a reasonable knowledge of Basic, called "Writing Games Programmes" on 17-19 February 1984, and "Programming Your Micro" on 20-22 February 1984. Contact the College for further details.

Video Camera Interface

This new Video Camera Interface allows a standard video signal to be digitised and captured by the Beeb, with a resolution of 220 by 312 pixels and is capable of converting up to 64 levels of grey. When used with a video camera (black and white or colour), a command given to the micro enables the video image to be stored in the micro. This image may then be manipulated further under user control. The final image can then be saved on disc (or cassette). The interface costs £174 (+VAT) from Educational Electronics, 30 Lake Street, Leighton Buzzard, Beds., LU7 8RX.



ested on O.S. 0·1 and 1·2
nd on Basics I and II

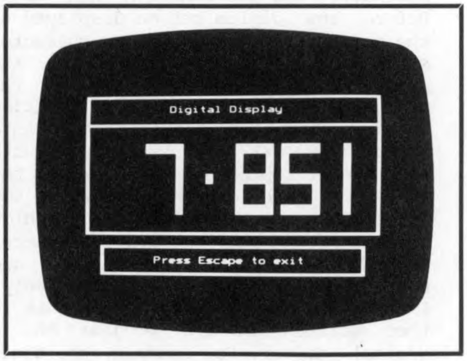
LARGE DIGITAL DISPLAY IN MODE 7

by Mike Williams

There are many applications where it can be very attractive to display numerical information on the screen in the form of a large digital display, similar to that produced by many digital readout meters and other types of equipment. We present here some interesting routines that you can incorporate in your own programs to do just that, in mode 7. The display can be further enhanced by using some of the procedures described in this month's Teletext Mode article.

Using the routines included here will enable you to produce a large digital display on the screen. The display may contain up to 6 characters (including digits, decimal point and minus sign) and there is room on the screen for two such digital displays, one above the other, if you need to display two values concurrently. The digits are changed quickly and realistically, and provide an ideal way of displaying rapidly changing numbers in a clear and readable way, particularly if they are to be visible from a distance or to a large group of people. Techniques very similar to this have been used to turn the Beeb into an accurate voltmeter, temperature gauge, pressure gauge, air speed indicator etc.

The routines to produce the display consist of two procedures, PROCdigit1 and PROCdigit2, together with a function FNchar. The use of these procedures is illustrated by a short program which also uses two of the procedures, PROCmsg1 and PROCbox, described in this month's article on the Teletext Mode, to give a neat and professional look to the results. The value being displayed on the screen is obtained from channel 1 of the A/D converter. This will provide a value, which floats around a predetermined mean, even with nothing connected to


Digital Display

7·851

Press Escape to exit

the analogue port. If you are able to connect anything to the analogue port, such as a joystick or a low voltage input, then you should be able to display a much wider range of values on the screen. The instructions in lines 200 to 220 simply convert the A/D value to lie within the range 0 to 10 (see page 202 of the User Guide) and to conform to the required 6 character format, before calling PROCdigit2 to display the value in large digits.

The analogue port is only available on the Model B. For a Model A, change line 200 to read:

```
  200 v$=STR$(RND(10000)-1)
```

This will generate random values for display instead, and the program will then run. Indeed, you can display any number, up to a maximum of six characters, on the screen by assigning the number in character form to the variable v$ as in line 200.

PROGRAM NOTES

In the program presented here, each digit is made up on a grid 7 characters high and 5 characters wide (though other sizes could also be designed) and each of the 7 rows is preceded by the graphics control code 145 (for red graphics). Thus in total, each digit consists of 42 characters, and the character string for each digit is assigned to the corresponding element of an array DIG$, so that element 0 contains the characters that form '0', element 1 contains the characters that form '1' and so on.

To display a digit on the screen, a text window 6 characters wide and 8 characters high (to avoid problems of unwanted scrolling) is defined, and

»

then the character string for that digit is simply poured into the defined area, which constrains the characters to produce the correct shape. The screen width can accommodate a maximum of six large digits of this size, and the procedures have been written to correctly display decimal points and minus signs should they occur.

To achieve these results two procedures and one function are used. Before any digits can be displayed on the screen, the corresponding character strings have to be assigned to the array DIG$. This is done by the procedure PROCdigit1 and the function FNchar. Each large digit is coded as a set of 7 numbers (to save space), with each number representing, in order, the state of each of the 7 rows of the digit. Each row is made up of graphics character 255 and the space character 32. These two characters are represented by 1 and Ø respectively, forming a 5 digit binary number that is then written in the program as a decimal number. For example, the digit 8 is coded:

```
   11111
   1ØØØ1
   1ØØØ1
   11111
   1ØØØ1
   1ØØØ1
   11111
```

This gives the set of numbers 31,17,17,31,17,17,31 which appear in a data statement (since binary 11111=31 decimal, and 1ØØØ1=17). The procedure PROCdigit1 reads the data numbers and calls FNchar to decode each number into the corresponding string of ASCII 255 and 32 characters.

To place the digits on the screen requires the use of the procedure called PROCdigit2. This expects a number with a maximum of 6 digits (including any decimal point or minus sign) will be passed as a character string (v$), together with the position (x%,y%) of the top left corner of the screen area in which the six characters will be placed. The procedure then places the corresponding large digits on the screen, working from right to left. This ensures that the rightmost digits, which in practice are the ones most likely to change, are the ones which are re-displayed first. The positions of the large digits on the screen are determined by the parameters x% and y% which help to define the text window at line 2030.

Although the large digital display could be used, as here, to measure some external input, the same techniques are much wider in application, and can be used whenever such a digital display is needed.

Note that in the listing below, the two procedure definitions for PROCmsg1 and PROCbox are copied exacly from the program TEXTED that forms part of the article on the Telext Mode that appears elsewhere in this issue.

```
  1Ø REM Program DIGITS
  2Ø REM Version B1.4
  3Ø REM Author Mike Williams
  4Ø REM BEEBUG Jan/Feb 1984
  5Ø REM Program subject to Copyright
  6Ø:
 1ØØ MODE 7
 11Ø ON ERROR GOTO 9ØØ
 12Ø PROCdigit1
 13Ø VDU23,1,Ø;Ø;Ø;Ø;
 14Ø PROCbox(5,1,1,4,39)
 15Ø PROCbox(5,1,4,11,39)
 16Ø PROCbox(5,3,16,3,35)
 17Ø PROCmsg1("Digital Display",2,12,2)
 18Ø PROCmsg1("Press Escape to exit",Ø
,1Ø,17)
 19Ø REPEAT
 2ØØ v$=STR$(INT(1ØØØØ*ADVAL(1)/6552Ø))
 21Ø v$=STRING$(4-LEN(v$),CHR$48)+v$
 22Ø value$=""+LEFT$(v$,1)+"."+RIGHT$(
v$,LEN(v$)-1)
 23Ø PROCdigit2(value$,2,6)
 24Ø UNTIL FALSE
 25Ø END
 26Ø:
 9ØØ ON ERROR OFF:MODE 7
 91Ø IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
 92Ø END
 93Ø:
1ØØØ DEF PROCmsg1(msg$,c%,x%,y%)
1Ø1Ø PRINT TAB(x%-1,y%);CHR$(129+c%);m
sg$;
1Ø2Ø ENDPROC
1Ø3Ø:
15ØØ DEF PROCbox(c%,x%,y%,h%,w%)
151Ø LOCAL c$,i%,y1%
152Ø y1%=y%+h%-1
153Ø FOR i%=y% TO y1%
154Ø PRINT TAB(x%-1,i%);CHR$(145+c%);
```

»

```
  1550 IF i%=y% THEN PRINT CHR$183; ELSE
  IF i%=y1% THEN PRINT CHR$245; ELSE PRI
 NT CHR$181;
  1560 IF i%=y% THEN PRINT STRING$(w%-2,
 CHR$163);
  1570 IF i%=y1% THEN PRINT STRING$(w%-2
 ,CHR$240);
  1580 IF i%=y% THEN c$=CHR$235 ELSE IF
 i%=y1% THEN c$=CHR$250 ELSE c$=CHR$234
  1590 IF i%>y% AND i%<y1% THEN PRINT TA
 B(x%+w%-2,i%);CHR$(145+c%);
  1600 PRINT TAB(x%+w%-1,i%);c$;
  1610 NEXT i%
  1620 ENDPROC
  1630:
  1700 DEF PROCdigit1
  1710 LOCAL d%,r%,x%
  1720 DIM DIG$(9)
  1730 FOR d%=0 TO 9
  1740 FOR r%=1 TO 7
  1750 DIG$(d%)=DIG$(d%)+CHR$145
  1760 READ x%
  1770 DIG$(d%)=DIG$(d%)+FNchar(x%)
  1780 NEXT r%,d%
  1790 ENDPROC
  1795:
  1800 DATA 31,17,17,17,17,17,31
  1810 DATA 4,4,4,4,4,4,4
  1820 DATA 31,1,1,31,16,16,31
  1830 DATA 31,1,1,31,1,1,31
  1840 DATA 17,17,17,31,1,1,1
  1850 DATA 31,16,16,31,1,1,31
  1860 DATA 31,16,16,31,17,17,31
  1870 DATA 31,1,1,1,1,1,1
  1880 DATA 31,17,17,31,17,17,31
  1890 DATA 31,17,17,31,1,1,1
  1895:
  1900 DEF FNchar(x%)
  1910 LOCAL i%,z$:z$=""
  1920 FOR i%=4 TO 0 STEP -1
  1930 IF x% AND 2^i% THEN z$=z$+CHR$255
 ELSE z$=z$+CHR$32
  1940 NEXT i%
  1950 =z$
  1960:
  2000 DEF PROCdigit2(v$,x%,y%)
  2010 LOCAL d%,d1%
  2020 FOR d%=LEN(v$) TO 1 STEP -1
  2030 d1%=5-(LEN(v$)-d%):VDU28,x%+6*(d1
 %),y%+7,5+x%+6*(d1%),y%
  2040 d1%=ASC(MID$(v$,d%,1))
  2050 IF d1%=46 THEN CLS:PRINT TAB(2,3)
 ;CHR$145;CHR$255; ELSE IF d1%=45 THEN C
 LS:PRINT TAB(0,3);CHR$145;STRING$(5,CHR
 $255); ELSE IF d1%>47 THEN VDU30:PRINT
 DIG$(VAL(MID$(v$,d%,1))); ELSE CLS
  2060 NEXT d%
  2070 ENDPROC
```

AMS DRIVES UPDATE

Last month we carried a review of the 3 inch AMS disc drives for the BBC micro. These seemed to us to be excellent drives, but as we noted in our review, were marred by the shortness of the supply cable, and the fact that it was untethered at the disc drive end. This cable carries the supply for the unit and is plugged into the low voltage power output socket underneath the Beeb. These faults on our review model have been cured on current production models, and in view of this we have arranged a special offer on these drives to members.

Drive type	Normal price	Members price
3" single (100K)	£255	£225
3" double (200K)	£399	£369

This offer is not available to members outside the U.K.

All prices include VAT and carriage. Send cheque and membership number to

BEEBUGSOFT
P.O. BOX 109
High Wycombe
Bucks
HP11 2TD

AN INTRODUCTION TO FORTH

by John Yale

Many members will have seen alternative languages to Basic advertised, and may be wondering which, if any,to choose. Next month, we will be publishing a comparative review of three versions of FORTH which are available for the BBC micro, but first, what is FORTH and what does it offer compared with Basic?

FORTH is different from any other language in common use, although it does share common features with many. The key to FORTH is that it is an extensible interactive language. Starting with a few dozen primitives defined in machine language, FORTH is extended by defining new operations (called 'words') in terms of the primitives. As soon as a new word is defined, it may be used in further definitions or executed immediately as if it were a primitive. Thus much of the language is actually written in FORTH itself!

The systems currently available for the BBC micro have around 200 words already defined. Programming consists of defining new words in terms of previously defined ones until one of them performs all the functions of the program which is required. Communication between words is not via parameter lists or variables (although variables may be used) but via a stack. This leads to very easy debugging, as most words can be executed from the keyboard by typing in their test arguments (which places them on the stack), then typing the name (which executes it) and finally displaying the stack for results.

The key then to FORTH is its building block approach, allowing new facilities to be added to the system, including new data types and compiler extensions (eg. if a CASE construct is not supplied, it may be added by the user). There is no sharp division between system and application in FORTH, rather the system grows towards the users application.

The following short example of a FORTH program shows how this works in practice. The purpose of the program is to display the dots and dashes comprising the morse code for S.O.S. Note how new words are defined in terms of primitives and previous definitions. First, the program defines DI, DA and GAP, then uses these to define S and O, then SOS and HELP. Each new definition can be tested as it is typed in and finally, typing HELP, will continually display on the screen the morse code for S.O.S.

```
: DI   ." ." ;
: DA   ." -" ;
: GAP   SPACE ;
: S    DI DI DI GAP ;
: O    DA DA DA GAP ;
: SOS   S O S GAP GAP ;
: HELP   BEGIN   SOS   Ø UNTIL Ø ;
```

To describe in a few words what FORTH is and why you should want it on your BBC Computer, I can do no better than to quote from the back of 'The Complete FORTH', a new book about FORTH just published by Sigma Technical Press:

"FORTH is a new, unusual and exciting computer language. Originally developed to control telescopes, it has since been applied in many diverse fields including the animation sequences for 'Star Wars'.

FORTH is a compact and fast language, faster than BASIC yet more flexible. It is more than just a language, it is a programming language, editor, assembler and disc operating system all rolled into one. In short, a complete 'environment'."

DOUBLE DENSITY DISC CONTROLLERS

reviewed by Philip Le Grand and Mike Williams

This month we review two of the double density disc controllers that have recently become available. These are the Microware DDFS and the Kenda Professinal MDFS which enable you to double the storage capacity of most disc drives at a cost less than half the price of a second drive. They also provide a number of other features not available with the Acorn DFS. Too good to be true? Maybe not.

Product: Kenda Professional MDFS
Supplier:The Kenda Group
Nutsey Lane, Totton,
Southampton SO4 3NB.
Cost: £138 inc VAT

Product: Microware DDFS
Supplier:Microware
637a Holloway Road,
London N19 5SS.
Cost: £100.58 inc VAT.

The standard BBC micro, with disc interface and Acorn DFS, will allow a maximum of 31 files per surface using 40 or 80 track discs in single density mode. This provides a storage capacity of 200k or 400k on one double sided disc. Several firms already produce their own DFS system, often extending the limit on directory size from 31 to 62 but this does not increase the overall storage capacity of the disc. Double density discs allow twice as many sectors to be packed on each track and are readily available, so why not use the extra capacity?

Unfortunately, the 8271 disc controller used in the BBC micro is incapable of providing double density format, and so another controller must be used instead. This is usually the 8272 which can support both single and double density modes. Since this chip is not pin compatible with the 8271, it must be used with a special carrier board which will plug into the normal 8271 socket. Two systems that follow this course of action and provide true double density on the BBC micro are now available.

These are the Microware DDFS controller, and the Kenda Professional MDFS system. Both systems contain all the necessary parts to implement a complete disc upgrade on a Model B, though if you already have a disc interface, most of the extra chips will be redundant and you will need to remove your Acorn or Watford DFS. Both systems were comparatively easy to install and both performed to specification. However, the two systems offer completely different filing systems, and it is a comparison of these that will most likely form the basis of your own choice.

MICROWARE DDFS

This controller is supplied on a 2" by 2" pcb and the construction is of a high quality. The kit consists of the controller board, DDFS in EPROM, a dil switch and all the necessary interface chips. In order to review the Microware package in this issue, we had to work without the full manual that will be provided to purchasers, as this was still being printed. We did have sufficient information supplied to fit the system and this task proved quick and easy - no soldering is required.

Initially we were unable to obtain satisfactory results but discussions with Microware revealed that the double density system was designed for use with issue 4 and higher boards. On issues 1 to 3 boards, it is necessary to solder the carrier board in position or fit an alternative socket. Apart from this there were no other problems.

»

The dil switch supplied is fitted to the keyboard and can be set to optimise disc access (different makes of disc drive require different settings). This was a useful extra which is not normally provided with the Acorn DFS.

Switching the micro on reveals that Microware MDFS is now installed. All the standard Acorn DFS commands are available, and there is one useful addition, a built in formatter on the DDFS ROM. The start of memory (PAGE) is set to &1900 as for Acorn DFS systems.

The system is able to detect automatically whether a single or double density formatted disc is being used, and allows up to 31 files to be stored per surface (but see notes at end of review). The storage capacity of the disc will now be 400k on 40 track and 800k on 80 track double sided discs. It appears to be fully compatible with Acorn's DFS, faultlessly saving and loading programs, data and memory dumps. We have been able to interchange without difficulty, single density discs on a variety of disc drives and with three different filing systems in use, including Acorn DFS and Microware DDFS. In timing tests, the DDFS was marginally slower at loading Wordwise files from double density discs than from single density discs, but Basic programs were up to 30% faster at loading from double density discs as a result of a process called interleaving.

The system worked reliably during our tests, allowing all the normal operations with discs to be carried out. There would appear to be few, if any, disadvantages with the Microware system. It immediately doubles the storage capacity of all normal disc drives (including AMS 3" drives). The system will read and write standard single density discs with no apparent problems, though the Acorn filing system itself is not very sophisticated.

KENDA PROFESSIONAL DMFS

The first thing you notice about this system, is that the whole controller is neatly encapsulated in an epoxy resin unit including the disc filing system. No fitting problems were encountered with this unit, which simply plugs into the disc controller socket, and seemed quite stable. The ribbon cable coming from the unit plugs into one of the ROM sockets in place of the usual DFS chip. This needs a little care as the plug does not fill all the pin connections, and the ribbon cable runs over adjacent ROMs.

Like the first system reviewed, the Kenda Professional is a complete disc upgrade. It is accompanied by a high quality manual, that is both well written and well produced, and a separate utilities disc. This system takes up no user memory, since it contains its own 2k workspace. Thus PAGE is still set to &0E00, enabling the longest cassette based programs to be loaded and run, and of course, giving you an extra 2k of memory for your own programs compared with other disc systems for the BBC micro.

Unlike the Microware upgrade, Kenda's system is completely different to Acorn's, and is based on the widely used CP/M system, using CP/M commands. This is a much more sophisticated

system as disc blocks are dynamically allocated, both for files and for the directory, as required. Files can always be extended, unless the disc is full, which is also the only limit on the size of the directory. This is both efficient and flexible, and allows a disc to store a large number of very small files or fewer much larger files.

Moreover, the system treats both sides of a double sided drive as being continuous, so that once you have used up all the space on one side the system continues automatically on the other. In fact, you won't usually know (or care) which side of a disc a file is stored on (and it may be on both!). This also means that a single file on an 80 track disc could extend up to nearly 800k in size! The Kenda Professional MDFS automatically selects single or double density modes when reading discs. You can still access Acorn format discs by using the GROW utility on the supplied utilities disc, and likewise, you can make a file readable to an Acorn DFS, by using the SHRINK utility. The formating procedure is also a disc based utility.

The Professional MDFS has its own set of commands based on CP/M but all standard Acorn commands can be recognised, and those which have an equivalent MDFS command will be executed. Other Acorn commands will be ignored but no errors will be generated. Because of the disc filing structure being used, Acorn commands like *COMPACT are quite unnecessary.

The Professional MDFS offers a number of facilities not available with Acorn format discs. Any bad sectors encountered are automatically marked and avoided. Files that are initially erased can still be found and recovered. It is possible to use both the disc filing system and another filing system (say cassette or Teletext) together without switching between the two.

The Professinal MDFS comes with a clear concise manual, and the whole system was very pleasing to use. Although the controller became rather warm in use, no problems were encountered.

To summarise, The Kenda Professional MDFS will, like the Microware system, double the capacity of any disc drive it is used with, and offer a filing system with much more sophistication than will any system based on the Acorn DFS. It also has the significant advantage of saving the disc user 2k of memory compared with other disc systems currently available. The major drawback, of course, is that Acorn format discs cannot be read or created directly but must be converted to and from the Kenda format using the supplied disc-based utilities.

CONCLUSIONS

Both systems performed well, and the only way to choose between them is to look at the facilities which they offer. If you want Acorn compatibility with up to 128 files (version 0.98) within the constraint of an Acorn style filing system, then the Microware system should be the choice. However, if Acorn compatibility is not very important to you, and you want an intelligent system which takes up no user RAM, and handles files dynamically using CP/M type file commands, then the Kenda system may be the one to choose. This is particularly good for those with significant file handling applications. There are two points to bear in mind if you are going to buy either one of these upgrades:

1. Neither of the two systems reviewed allowed a ROM expansion board to be plugged in at the same time, except for the Watford Electronics board, although we believe that Kenda are modifying their system so that an ATPL board may be installed together with the MDFS.
2. Acornsoft discs with disc protection could not be read by either of the double density controllers reviewed even in the single density mode, as the protection involves direct access to the 8271 controller chip.

We have been testing version 0.9 of the Microware DDFS. Version 0.98 will be available early in 1984, and will, it is claimed, simulate the registers of an 8271, in such a way as to allow protected Acorn discs, of the kind currently produced, to be read. We understand that this version will also allow a maximum of 128 files per disc surface. Microware have told us that version 0.98 will be supplied free in exchange for version 0.9. A version 1.0 is also planned which will contain its own RAM, thus releasing about 2k of user memory for programs. No firm date is available for the release of this enhancement.

We would like to thank both Microware and Kenda for their help and cooperation while compiling this review.

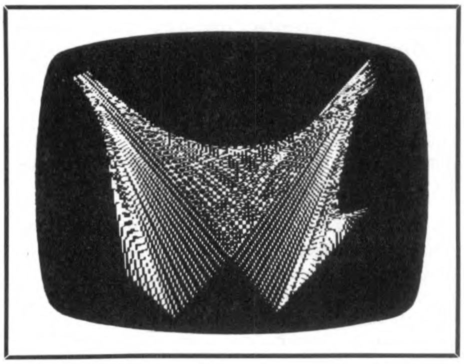
Tested on O.S. 0·1 and 1·2 and on Basics I and II

DANCING LINES

by David A. Fell

This program not only produces an interesting graphics display, but is very interesting in showing how a sequence of random numbers generated in Basic is not really random at all.

This is a short program that produces an attractive screen display by simulating the bouncing of two points around the screen at random, with a variable coloured line drawn between the two. An apparently random sequence is built up on the screen, and then erased in the same order as it was created. Once the screen has 'emptied', the program repeats, producng a different sequence for each screen.



RANDOM NUMBERS

The two key facts that enable the program ,to precisely erase the lines, without any co-ordinates being stored, are the plotting technique used, and the repeatability of Basic's random number generator. This is not a true random number generator, but produces what are termed pseudo-random numbers. This means that the numbers that it produces may appear random, and indeed will satisfy most tests for randomness, but are actually produced by a fixed set of calculations which can be made to repeat. Whenever a 'seed', or starting point, for the random number generator is entered, it is possible to regenerate a particular series of 'random' numbers.

To seed the Beeb's random number generator, you use the form:

```
   X=RND(N)
```

where N is a negative number. As an example, try this:

```
   PRINT RND(-100)
```

When 'seeding' the Basic random number generator, the value that RND returns is the same as was passed to it, in this case -100. Once a negative value has been passed, the generator is 'seeded', and thus now predictable. Moreover, if you now type:

```
   PRINT RND(10)
```

for example, immediately after seeding the generator as above you will always get a 9 printed, which is always the first value to be returned in the particular sequence chosen.

PROGRAM NOTES

In the program, line 120 of the program creates a random seed, and this is used at lines 130 and 150 to start the generator off producing the same sequence of numbers. Lines 140 and 160 call the procedure to produce a screen of lines, and both the calling and seeding sections are contained within a REPEAT ... UNTIL loop that cycles endlessly, or until Escape is pressed.

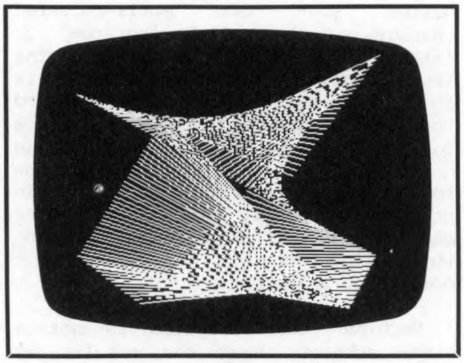
At the start of the program, line 90 turns off the cursor, as this only serves as a distraction in graphical displays. Line 100 effectively turns off the flashing colours, which are inevitably going to be generated as a result of both the usage of exclusive-or plotting and the random colour range that is selected, and which would appear unattractive if left flashing. This is achieved by setting the 'mark' period to zero, and thus leaving all of the flashing colours permanently in only one of their two possible colours.

The display itself is produced by the procedure PROCpicture. Lines 210 to 280 generate random starting points and random increments for the two ends of the line. The increments are fixed so that they will start the two ends moving off in opposing directions. The FOR ... NEXT loop starting at line 290 will determine how many lines will be drawn for each pattern, and adjusting the value of 500 to, say, 100 will give a shorter pattern.

The procedure uses the exclusive-or GCOL plotting option (selected at line 300), and the random colour. Exclusive-or plotting has the unusual property in that, if any graphics with the same colour and co-ordinates are plotted an even number of times, then they 'cancel' out, leaving whatever happened to be on the screen before. As we start of with a blank screen, and all plotting is done twice, at the same co-ordinates, and in the same colour, the net result is that there is nothing left on the screen by the end of the suequence.

Lines 330 to 440 are concerned with the mechanics of bouncing a point around an enclosed rectangle (in this case the screen).

```
 10 REM Program Dancing Lines
 20 REM Author David A. Fell
 30 REM Version v1.2
 40 REM BEEBUG JAN/FEB 1984
 50 REM Program subject to Copyright.
 60 :
 70 ON ERROR GOTO 470
 80 MODE 2
 90 VDU23;11,0;0;0;0
100 *FX9
110 REPEAT
120 SEED%=ABSRND
130 A%=RND(-SEED%)
140 PROCpicture
150 A%=RND(-SEED%)
160 PROCpicture
170 UNTIL0
180 END
185 :
190 DEF PROCpicture
200 X1%=RND(1279)
210 Y1%=RND(1023)
220 X2%=RND(1279)
230 Y2%=RND(1023)
240 XI1%=RND(4)*5
250 YI1%=RND(4)*5
260 XI2%=-RND(4)*5
270 YI2%=-RND(4)*5
280 FORI%=0TO500
290 GCOL3,RND(16)-1
300 MOVE X1%,Y1%
310 DRAW X2%,Y2%
320 IF X1%+XI1%>1279 XI1%=-RND(4)*5
330 IF X1%+XI1%<0 XI1%=RND(4)*5
```



```
340 IF Y1%+YI1%>1023 YI1%=-RND(4)*5
350 IF Y1%+YI1%<0 YI1%=RND(4)*5
360 IF X2%+XI2%>1279 XI2%=-RND(4)*5
370 IF X2%+XI2%<0 XI2%=RND(4)*5
380 IF Y2%+YI2%>1023 YI2%=-RND(4)*5
390 IF Y2%+YI2%<0 YI2%=RND(4)*5
400 X1%=X1%+XI1%
410 Y1%=Y1%+YI1%
420 X2%=X2%+XI2%
430 Y2%=Y2%+YI2%
440 NEXT
450 ENDPROC
460 :
470 ON ERROR OFF: MODE 7
480 IF ERR<>17 REPORT:PRINT" at line
";ERL
490 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BIRD SOUNDS - I.Evans

Here is a short program to generate bird-like sounds. It can of course be modified to taste - although the duration parameter in the SOUND statement has to be 3 for best effect:

```
10 ENVELOPE 1,1,5,5,-10,30,30,30,50,0,0,1,100,100
20 FOR I=0 TO 100:SOUND 1,1,130+RND(40),3:NEXT
```

Tested on O.S. 0·1 and 1·2 and on Basics I and II

MACHINE CODE GRAPHICS (16K)

by Peter Clease

Although the BBC micro is capable of excellent graphics when programmed in Basic, any form of animation can seem very slow and uneven beacause of the time taken to change the display. The solution to this problem, and one which has been used very successfully in many of the commercial computer games available, is to use machine code. This month we start a series of articles which will introduce the basic principles of machine code graphics.

In this series we will show you how the screen area of memory is structured for the different modes and how to define your own mutli-coloured characters (often called 'sprites' or 'slabs'). We will also describe the techniques which allow these characters to be moved swiftly and smoothly around the screen. Although the main routines for doing this will mainly be in machine code, we will also show you how to incorporate these routines in Basic programs as well. Some knowledge of machine code will help in understanding the routines but you certainly won't need to be an expert.

Machine Code Graphics is not a simple subject, though the results as seen in many commercial computer games can be outstanding. This month we will look at the basic screen structure and the definition of characters in mode 4 (easy because only two colours are possible) before moving on next month to multi-coloured characters in modes 1 and 2 and later to animation techniques.

* * * * * * * * * * * *

INTRODUCTION

To create graphics, the computer looks at numbers stored in the area of its memory known as its 'screen memory' and converts those numbers into pixel patterns and colours. A pixel is the smallest plottable point possible in any mode. To create graphics then, all the programmer has to do is to store the numbers corresponding with the image he wants to produce in that area of memory. To produce graphics in Basic in a similar way would be extremely slow, and hence a range of graphics commands and definable characters are available for this purpose, and these themselves are produced by machine code subroutines.

One consequence of writing programs that directly access screen memory (or any other part of memory) is that such programs will not work across the Tube. In many cases commercial software for the BBC Micro has ignored this potential disadvantage in favour of the benefits in increased speed that result.

If you would like to see the Basic version of a machine code program that plots a short line on the screen, type in the following program and run it.

```
10 MODE4:VDU23;11,0;0;0;0
20 ?&5800=255
30 ?&5808=255
40 ?&5810=255
50 REPEAT UNTIL FALSE
```

If no line is visible at the top of the screen then you may need to move this down one line by using *TV254,1 (you may find this unnecessary or that values other than 254 are better depending on how your TV or monitor is adjusted - see page 23 in the User Guide), so that the top line of the display is clearly visible. The first line of the program selects mode 4 and switches off the cursor before poking the value 255, coding a solid white line, into three consecutive positions across the screen. It can be seen that the memory locations are not consecutive. The reason for this will become apparent when we discuss the screen structure in detail.

The machine code equivalent of the above is as follows:

»

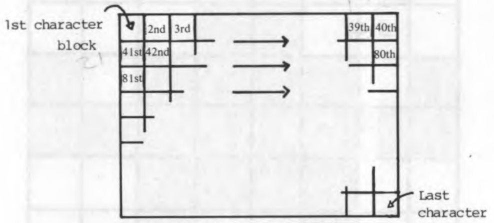
```
  10 FOR PASS=0 TO 3 STEP 3
  20 P%=&2000
  30 [OPT PASS
  40 .start
  50 LDA #22:JSR &FFEE:LDA #4:JSR &FFEE
  60 LDA #255:STA &5800:STA &5808:STA
&5810
  70 RTS
  80 ]
  90 NEXT PASS
  100 TIME=0:REPEAT UNTIL TIME>400
  110 CALL start
  120 VDU23;11,0;0;0;0
  130 REPEAT UNTIL FALSE
```

As is always the case on the BBC micro, the assembler code is embedded within a Basic program. In this particular instance we are loading at &2000. The first line of machine code changes to mode 4 by executing the equivalent of VDU22,4, and the second line stores the values that produce the displayed line at the appropriate memory locations. A time delay has been incorporated so that you can see the assembler listing of the machine code before the routine itself is called. Press down Shift and Ctrl together if you want to keep this on the screen for longer.

SCREEN MEMORY STRUCTURE

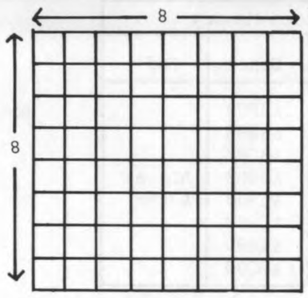
The basic screen structure in any of the graphics modes is that of lines of character blocks running across the screen.

see 3/84 p.35



Layout of character blocks in a 40 character mode. (modes 1 and 4)

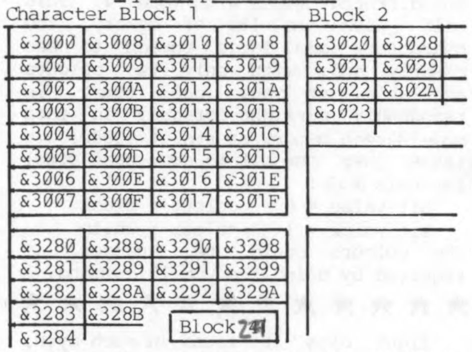
Each character block is always made of a combination of pixels arranged as an 8 by 8 matrix.



Layout of pixels within a character block.

Depending on the mode, each byte of screen memory represents either two, four or eight pixels horizontally (see the sections on the modes for detail about this). The screen's bytes are always arranged vertically down character block columns. The diagram below demonstrates this for mode two. From this you can see that each character block is made up of 4 bytes horizontally, and thus each byte represents two pixels in this mode. Screen memory starts from &3000 in this mode. The other modes have a similar structure.

Byte Map for Mode 2



Character Block 1				Block 2	
&3000	&3008	&3010	&3018	&3020	&3028
&3001	&3009	&3011	&3019	&3021	&3029
&3002	&300A	&3012	&301A	&3022	&302A
&3003	&300B	&3013	&301B	&3023	
&3004	&300C	&3014	&301C		
&3005	&300D	&3015	&301D		
&3006	&300E	&3016	&301E		
&3007	&300F	&3017	&301F		
&3280	&3288	&3290	&3298		
&3281	&3289	&3291	&3299		
&3282	&328A	&3292	&329A		
&3283	&328B	Block 24			
&3284					

It is important to note the increase in value from the bottom pixel of one character block to the top pixel of the block below, (&261) as you will see later. You can see that byte addresses change by eight as you move horizontally. Since each byte codes for two pixels (in this mode), this means that by moving a character one byte to the right, you move it two pixels, and this is the minimum increment for a character. Later, ways will be shown of moving characters one pixel only.

»

Screen Memory		
Mode	Start	End
Ø	&3ØØØ	
1	&3ØØØ	
2	&3ØØØ	
3	&3ØØØ	All at
4	&58ØØ	&7FFF
5	&58ØØ	
6	&6ØØØ	
7	&7CØØ	

The table above shows the start and end addresses of screen memory for the different modes.

Of these, only modes Ø, 1, 2, 4 and 5 are graphics modes. This series does not deal with modes 3, 6 and 7 which have limited graphics facilities (user defined characters in modes 3 and 6, Teletext graphics in mode 7).

The best way to deal further with machine code graphics is to examine each mode on its own, now that you know about the screen structure.

MODE 4

We will start our detailed look at the different modes with mode 4. This will work in 16k of memory, the characters are large enough to see clearly and, with only two colours available at any one time, is reasonably straight-forward. In this mode screen memory starts at &58ØØ. It takes just one bit to code each pixel for this mode:

```
   Bit value = Ø     colour = black
   Bit value = 1     colour = white
```

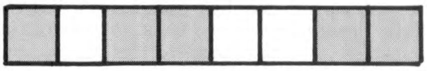
The colours could be changed if required by using the VDU 19 command.

Eight bits are found in each byte, or memory location, and so each memory location codes 8 pixels. Since character blocks consist of 8 by 8 pixels, a whole character in this mode is coded by eight consecutive bytes of memory. Each screen line consists of 4Ø characters, equivalent to 8 by 4Ø (=32Ø) bytes of memory. Thus to move horizontally by one character the screen memory address changes by 8 and to move vertically the screen memory address changes by 32Ø. You may find it helpful to draw a memory map showing the start address of the first character in each row and each column.

Each byte codes just one line in a character block as in the following example:

1	Ø	1	1	Ø	Ø	1	1

Each bit can be a 'Ø' or '1'. This sequence would give a line of pixels like this:

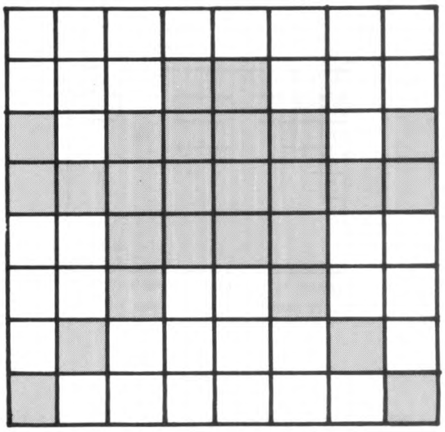


Thus poking the binary number 1Ø11ØØ11 into memory location &58ØØ will give the above pattern as a line in the top left memory location. Try it

```
   MODE ~~Ø~~ 4
   ?&58ØØ=179
```

The value 179 is the decimal for the binary number 1Ø11ØØ11. This could equally well have been put in hex as &B3.

An example, the plotting of a space invader, should make the use of this mode clearer. Draw the invader in an 8 by 8 grid.



Space invader.

»

Now convert each line to binary and thence to decimal:

```
Line 1  ØØØØØØØØ = Ø    (decimal)
Line 2  ØØØ11ØØØ = 24   (decimal)
Line 3  1Ø1111Ø1 = 189  (decimal)
Line 4  11111111 = 255  (decimal)
Line 5  ØØ1111ØØ = 6Ø   (decimal)
Line 6  ØØ1ØØ1ØØ = 36   (decimal)
Line 7  Ø1ØØØØ1Ø = 66   (decimal)
Line 8  1ØØØØØØ1 = 129  (decimal)
```

Here now is the Basic version of the program that will draw the space invader on the screen in the top left character position:

```
1Ø MODE4
2Ø VDU23;11,Ø;Ø;Ø;Ø
3Ø FOR I%=Ø TO 7
4Ø READ bytevalue
5Ø ?(&58ØØ+I%)=bytevalue
6Ø NEXT I%
7Ø REPEAT UNTIL FALSE
8Ø DATA Ø,24,189,255,6Ø,36,66,129
```

5800 (3/84 p.35)

We can put the same thing in machine code. Here the code has been put in the form of two procedures called PROCassemble and PROCinvader that could be incorporated into any Basic program.

```
 1ØØ MODE4:VDU23;11,Ø;Ø;Ø;Ø
 11Ø PROCassemble
 12Ø FOR I%=&6ACØ TO &6BFØ STEP 16
 13Ø PROCinvader(I%)
 14Ø NEXT I%
 15Ø END
 16Ø :
 1ØØØ DEFPROCassemble
 1Ø1Ø FOR PASS=Ø TO 2 STEP 2
 1Ø2Ø P%=&CØØ
 1Ø3Ø [OPT PASS
 1Ø4Ø .start
 1Ø5Ø LDY #Ø
 1Ø6Ø .loop LDA &7Ø,Y:STA (&8Ø),Y:INY:C
PY #8:BNE loop:RTS
 1Ø7Ø ]
 1Ø8Ø NEXT PASS
 1Ø9Ø FOR I%=Ø TO 7
 11ØØ READ A
 111Ø I%?&7Ø=A
 112Ø NEXT I%
 113Ø DATA Ø,24,189,255,6Ø,36,66,129
 114Ø ENDPROC
 115Ø :
 116Ø DEFPROCinvader(pos%)
 117Ø ?&8Ø=pos% MOD 256:?&81=pos% DIV 2
56
 118Ø CALL start
 119Ø ENDPROC
```

The data contains the decimal numbers that will produce the space invader. The data is poked into the locations from &7Ø to &77 and the machine code stored at &CØØ. In effect, the data is a character stored from &7Ø instead of the usual area of memory from &CØØ to &CFF reserved for user defined characters.

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

Since the bytes are arranged as groups of 8, running downwards, the data at the end of the program represents the lines in the character from top to bottom. Thus the value of the top line is Ø, the second is 24 and so on. To produce your own character, work out the lines as we did for the space invader above (using a character definer such as that published in BEEBUG Vol2 No5 will make things easier), and then place the data for the lines from the top line to the bottom in order in the data statement.

PROCassemble must be called first to assemble the machine code ready for use. From then on, writing PROCinvader(address) where 'address' is a screen address (between &58ØØ and &7FFF), will place a space invader on the screen in a corresponding position. If you do this, you must make sure that the address marks the start of a character block on the screen or the space invader will be split up into two separate parts. In our example program the loop from line 12Ø to 14Ø puts a space invader in every other position in line 16 on the screen. Try working out the memory address to put invaders in other positions.

You might also like to try moving the invader about the screen. To do this you will have to find some way of deleting the space invader in one position before re-displaying it in another.

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

Next month we will cover the basics of machine code graphics in the other modes, including the generation of multi-coloured characters.

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

Tested on O.S. 0·1 and 1·2 and on Basics I and II

PROTECT YOUR BASIC PROGRAMS

by P. Boardman

This interesting little utility is for those of you who wish to protect their own programs by making them unusable by anyone who does not know a prearranged password. It comes in the form of a procedure which must be added onto the end of the Basic program to be protected.

To use it, first type the procedure in and save it away on cassette or disc. Then 'spool' it out to cassette or disc under a different name e.g.

```
*SPOOL LOCKSP   <return>
LIST            <return>
*SPOOL          <return>
```

This now means that you have a version of your program saved, ready to be *EXECed onto the end of your program to be protected (See User Guide for full explanation of *SPOOL and *EXEC).

When that has been done, load in your Basic program (make sure that no line numbers exceed 32000). Then type *EXEC LOCKSP and wait for the protection program to load. This has now been joined to the end of your program.

To lock the program, type PROCLOCK, and type in a password when prompted. The password may contain any characters and be up to 256 characters in length, but you will have to remember the password you choose, so keep it short and meaningful. Once the password has been entered, the machine will pause, while it scrambles your program, changing the bytes of code. When that is completed, LIST your program and look at the code - it should make horrific reading!

SAVE this scrambled version away and use it as your main copy (although it cannot be RUN in its scrambled state).

When you want to use the program, simply load it in and type PROCLOCK. Then type in the same password as before and your program will be 'un-scrambled'. If you type in the wrong password, the program will be scrambled again and at this stage the program cannot be recovered, so take care not to forget the password.

Note that &C1 at line 32090 represents the length (plus one) of the LOCK procedure and is included in order to prevent the LOCK procedure itself being scrambled! Do be careful to type the procedure in EXACTLY as listed, otherwise PROClock itself mey become scrambled. This will produce error messages, and program recovery will not generally be possible.

You can try out the LOCK procedure by typing in the whole program listed below. Then proceed as follows:

```
LIST
RUN
PROClock
LIST
PROClock
LIST
RUN
```

Before entering PROClock the first time the program should LIST and RUN as normal. After locking the program, LIST will display the scrambled version. Typing PROClock a second time should now unscramble the program, and you should be able to LIST and RUN the program as before.

```
    0 REM EXAMPLE PROGRAM
    1 PRINT "THIS PROGRAM IS UNLOCKED"
    2 END
32000 DEFPROCLOCK
32010 INPUT"PASSWORD ",KEY$
32020 I%=PAGE+3
32030 REPEAT
32040 FORX%=I%+1 TO I%-4+?I%
32050 Y%=&1F AND (ASC MID$(KEY$,X% MOD
LEN KEY$,1))
32060 ?X%=(?X% AND &E0)+((?X% AND &1F)
EOR Y%)
32070 NEXT
32080 I%=X%+3
32090 UNTILX%>TOP-&C1
32110 ENDPROC
```

FOUR NEW GAMES FOR THE BBC MICRO

Reviewed by Robert Barnes

PROGRAM : TRANSISTORS' REVENGE
SUPPLIER: SOFTSPOT
PRICE : £6.95
RATING : ***

This very novel game, written in machine code, is set within an imaginary computer whose insides are displayed on the screen. This shows the CPU as the heart of the computer, with all its connecting data lines radiating out to the sides of the screen.

The rest of the computer's more colourful components (resistors, transistors etc) are attacking the CPU, which you have to protect by firing pulses of electricity down the data lines. This will destroy any component moving along the data line selected, but if a component should reach the CPU, then a life is lost. There is also a 'zap' key, which destroys all the attacking components on the screen. This can only be used once every 2Ø seconds.

This somewhat fanciful game is surprisingly good to play. The graphics and sound are to a high standard, and the game represents good value for money.

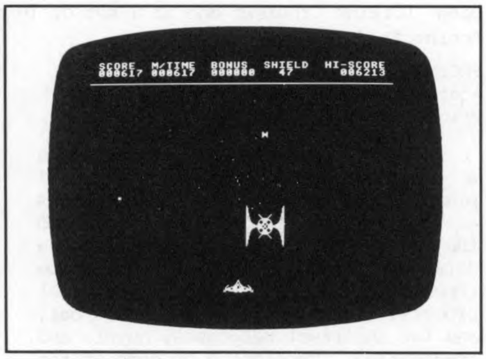
PROGRAM : NOC-A-BLOC
SUPPLIER: VIRGIN GAMES
PRICE : £7.95
RATING : ****

This new release from Virgin Games is an adaptation for the BBC micro of the arcade game Pengo. In this game the player is situated in the middle of a room filled with blocks which may either be vapourised or moved with a pushing action. The room also contains a number of characters called Spods, which must not be touched. These Spods must be destroyed by using one of two methods. The first, and best, method is to squash them between two blocks but they can also be destroyed by stunning them first.

The game increases in difficulty with each new screen, as the number of Spods also increases each time. To achieve a really good score, you need to try and position three special blocks in a line, and you gain a bonus by finishing a screen quickly. The need to think quickly and respond to a continually changing situation provides an interesting game.

PROGRAM : VORTEX
SUPPLIER: SOFTWARE INVASION
PRICE : £7.95
RATING : ****

Vortex is yet another very cleverly written game from Software Invasion, who are really building a high reputation for themselves in the writing of graphical games for the BBC micro. This game use high resolution graphics and is in black and white.



As you fly through space, with alien spacecraft attacking you from all sides, you must deploy laser torpedoes to destroy the aliens, while trying to avoid the asteroids and space debris which hurtle towards you. The feature that makes this game stand out is the way in which objects appear to get larger as they move towards you, only to shrink again as they move away (this seems to be almost a trademark of Software Invasion). [It's called 3D - Ed.]

This game is a very good example of its kind but suffers perhaps from the absence of any colour, while the game itself offers little in the way of variety.

PROGRAM : PINBALL ARCADE
SUPPLIER: KANSAS CITY SYSTEMS
PRICE : £10.35
RATING : *****

Pinball Arcade is an excellent new game from Kansas who have recently turned from producing software solely for the American Tandy TRS80 computer to producing programs for the Beeb. With Pinball, the screen displays a faithful representation of a pinball machine. You have contol over two flippers, which move just like the real thing. Colour is used to good effect, although it is a pity that the ball is white, as it can sometimes be difficult to see.

The feature of the program that makes it so good, is the facility to create your own design of pinball machine. There are six screen pages of shapes which can be placed anywhere on the board, and the layout can be saved to cassette for later use. The game has a lot of facilities, which unlike many other games, are not just gimmicky. The game is very fast, with such clever features as a ball that actually changes speed, just as with a real pinball machine.

This is a game which I am sure most people of any age will like. Mike Chalk, who wrote this program, has produced a winner. Pinball is highly recommended, even at its relatively high price.

EDUCATIONAL SOFTWARE

Reviewed by John Perkins

John Perkins reviews one of a set of 10 educational programs now being marketed by Acornsoft.

FACEMAKER
Acornsoft/ASK
Price: £9.95

Facemaker is a program which allows a child to build up an 'identikit' picture of a person. The instructions are clear, easy to understand and lead the user though the program in a straight-forward manner without undue repetition. I found the pictorial prompts, for the use of the space bar, and for incorrect responses, novel and imaginative. The length of some of the instructions and the vocabulary used would seem to put its main use in the upper half of the stated age range of 5 to 12, and even then, some of the terms would need explanation.

Sufficient opportunities are given, while building up the face, and on completion, to make changes and to experiment. The final results, however, can be a little disappointing since it is difficult to produce a result resembling what was in ones 'mind's eye', and one does get a certain feeling of sameness about the faces after a while. It is even quite difficult to distinguish between the

sexes if longer hair styles are selected for the men!

The documentation is simple but adequate. I am not totally convinced about some of the suggestions for classroom use but it could provoke some discussion and observation as well as being a vehicle for art-work. A.S.K. claim their "programs are designed, above all else, to make learning fun". Certainly the children I tried it out on found it fun to use, greeting their results with hoots of laughter, although I am not sure that they were actually learning much. In all, I would judge it as light-weight fun with limited classroom application.

Tested on O.S. 0·1 and 1·2 and on Basics I and II

THE TELETEXT MODE (PART 4)

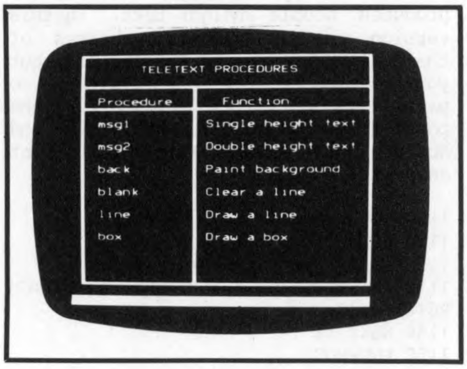
by Mike Williams

To conclude our series on the Teletext Mode, we describe this month a collection of useful procedures for creating Teletext Mode displays including one for large size digital displays.

The Teletext Mode is ideal for all manner of displays, particularly as the front page that introduces a program. All the colours are available, text may be in single or double height and there is a complete range of graphics characters as well. In addition, memory usage is unlikely to cause any problems, as the screen display uses only 1k of memory. If you generate Teletext displays for more than one program, you soon begin to realise that the same requirements keep on recurring and a small library of instant procedures undoubtedly saves time in the long run.

This month I propose to describe a number of procedures that I have found to be useful on many occasions when generating Teletext screens. Before describing them in detail there are a few general points to note first. For maximum flexibility, most of the procedures will create a display in any position of the screen. This position always refers to the visible display, and any control characters occupy preceding positions. Make sure you leave room for these. If any procedure refers to an area of the screen, then the reference position is always the top lefthand corner. Most of the precedures specify a colour (or colours) which is always specified as a value in the range Ø to 6. This is converted to the range 129 to 135 (text) or 145 to 151 (graphics) within the procedure itself.

Finally, if you try out some of the examples given in the text, you may occasionally find odd graphics characters appearing. This is usually because the prompt character '>' has appeared in a position where it is interpreted as a graphics character. In complete programs this can be easily avoided, and is often the reason for the 'REPEAT UNTIL FALSE' loop that you will see at the end of several of the example programs in this series.



TELETEXT PROCEDURES

Procedure	Function
msg1	Single height text
msg2	Double height text
back	Paint background
blank	Clear a line
line	Draw a line
box	Draw a box

If you type the procedures in as they are introduced, then you will be able to test them as we go along.

GENERAL TELETEXT PROCEDURES

The first procedure allows any text message to be displayed in any colour and in any position on the screen. This is defined as follows:

```
1ØØØ DEF PROCmsg1(msg$,c%,x%,y%)
1Ø1Ø PRINT TAB(x%-1,y%);CHR$(129+c%);ms
g$;
1Ø2Ø ENDPROC
```

This will display the text msg$ on the screen, in colour c% and in a position given by x% and y%, where y% is the line number and x% is the position on the line. Remember that the top line on the screen is always referred to as line Ø and that likewise, the first character on a line is in position Ø. For reference, a Teletext Mode line is 4Ø characters long (Ø to 39). You can test this procedure as soon as you have typed it in. Try typing as follows:

```
   CLS
   PROCmsg1("BEEBUG",2,16,1Ø)
```

You should see the word "BEEBUG" in yellow, displayed on line 1Ø and starting in position 16. The 'CLS' command is not strictly necessary, but will ensure that the screen is cleared of any existing information first.

»

When using the procedure, remember that a control code only applies to the line of the screen on which it is placed. If your text overflows onto the next line, it will normally default to white text on the second line.

The second procedure is similar, but produces double height text. In this version, the top and bottom halves of the characters are the same colour, but you could rewrite it with two parameters c1% and c2% to allow the possibility of the top half (c1%) and bottom half (c2%) having different colours.

```
1100 DEF PROCmsg2(msg$,c%,x%,y%)
1110 LOCAL i%
1120 FOR i%=y% TO y%+1
1130 PRINT TAB(x%-2,i%);CHR$(129+c%);CH
R$141;msg$;
1140 NEXT i%
1150 ENDPROC
```

Note that this time, the text will be preceded by two control characters. Remember too, that with both these procedures, the text, msg$, could itself contain control characters. In this way the procedures could, for example, be used to produce flashing text. For example, try typing:

```
   CLS
   PROCmsg2(CHR$136+"BEEBUG",2,15,10)
```

This will not only display "BEEBUG" in double height, yellow letters but the text will be flashing because of the extra control characer added onto the front of the word. For this reason, the position of the text on the line has been given as 15, instead of 16 as in the previous example, in order to leave room for the extra control character and so have the text displayed in the same position as before. The odd character that appears immediately after "BEEBUG" on the bottom of the two lines is merely the bottom half of the normal prompt character '>', which is also affected by the 141 control character. Just press Return before continuing.

COLOURED BACKGROUNDS

The next procedure is concerned not with text, but with graphics and allows any number of lines on the screen to be given a coloured background:

```
1200 DEF PROCback(c%,y1%,y2%)
1210 LOCAL i%
1220 FOR i%=y1% TO y2%:VDU31,0,i%,145+c
%,157:NEXT i%
1230 ENDPROC
```

This will colour the background from line y1% down to line y2%. If y1% and y2% have the same value, then just that one line will be given a coloured background. You might also decide to protect the coloured background by redefining the text window as we have done several times in previous articles in this series using the VDU28 command. Since this procedure is only sending control characters to the screen, and not text, the use of the VDU command at line 1220 is more concise than the use of PRINT TAB. You can try out this procedure by typing:

```
   CLS
   PROCback(0,0,9)
   <return>
   PROCback(3,10,19)
```

You should find that the top 10 lines of your screen have a red background, and the next 10 lines have a blue backgound, leaving the bottom 5 lines as normal. The extra Return is needed in order to see the next line as you type it in.

CLEARING LINES

Now for a procedure that at first may seem a little surprising. It is called PROCblank and puts a line of spaces anywhere on the screen. Although many Teletext Mode displays serve just as an introduction to a program, in more serious applications, such as word processing, stock control, file handling etc, a formatted screen in Teletext Mode may be the main display. In such cases, it is often necessary to display, change and remove text in various positions on the screen. Different text messages will likely be of different lengths so that you can never rely on one message neatly replacing another. Instead, PROCblank can be used to wipe out one message first, by painting the relevant positions with spaces. The procedure is quite short:

```
1300 DEF PROCblank(x%,y%,f%)
1310 PRINT TAB(x%,y%);SPC(f%);
1320 ENDPROC
```

»

A version of this procedure was used in the LINK utility program last month. Its use is also illustrated in the program TEXTED later in this article.

LINES AND BOXES

The next procedure will display a horizontal line on the screen, one graphics pixel in width, by using the graphics character with code 172.

```
1400 DEF PROCline(c%,x1%,x2%,y%)
1410 LOCAL i%
1420 VDU31,x1%-1,145+c%
1430 FOR i%=x1% TO x2%:VDU172:NEXT i%
1440 ENDPROC
```

As in the other procedures, c% defines the colour and y% specifies the line on the screen. The line starts in position x1% and finishes in position x2% on that line.

Finally, I have included a somewhat longer procedure called PROCbox, which can make Teletext displays look very professional. Using Teletext Mode graphics, this will draw a rectangular box of any size and in any position on the screen. This can be used to enclose text, or provide space for a response, or by using several together, can present a complete table format on the screen. The details of this procedure are as follows:

```
 1500 DEF PROCbox(c%,x%,y%,h%,w%)
 1510 LOCAL c$,i%,yl%
 1520 yl%=y%+h%-1
 1530 FOR i%=y% TO yl%
 1540 PRINT TAB(x%-1,i%);CHR$(145+c%);
 1550 IF i%=y% THEN PRINT CHR$183; ELSE
 IF i%=yl% THEN PRINT CHR$245; ELSE PRI
NT CHR$181;
 1560 IF i%=y% THEN PRINT STRING$(w%-2,
CHR$163);
 1570 IF i%=yl% THEN PRINT STRING$(w%-2
,CHR$240);
 1580 IF i%=y% THEN c$=CHR$235 ELSE IF
 i%=yl% THEN c$=CHR$250 ELSE c$=CHR$234
 1590 IF i%>y% AND i%<yl% THEN PRINT TA
 B(x%+w%-2,i%);CHR$(145+c%);
 1600 PRINT TAB(x%+w%-1,i%);c$;
 1610 NEXT i%
 1620 ENDPROC
```

The parameters required are the colour (c%), the position of the box (x%,y%), and the height (h%) and width (w%). Various Teletext graphics characters are then arranged to display a box with its top lefthand corner in the position x%,y%.

When building up tables using this procedure it is important to overlap the edges of the boxes to achieve the correct effect, and to follow a sequence of placing the boxes on the screen from right to left and top to bottom. The use of this and several other of the procedures is illustrated in the following program which shows how easily a formatted screen can be achieved using just these few procedures.

```
 1Ø REM Program TEXTED
 2Ø REM Version B1.3  Ø8/12/83
 3Ø REM Author Mike Williams
 4Ø REM BEEBUG Jan/Feb 1984
 5Ø REM Program subject to Copyright
 6Ø:
1ØØ MODE 7
11Ø ON ERROR GOTO 9ØØ
12Ø VDU23;11,Ø;Ø;Ø;Ø
13Ø PROCbox(5,3,1,4,35)
14Ø PROCbox(5,16,4,3,22)
15Ø PROCbox(5,3,4,3,14)
16Ø PROCbox(5,16,6,16,22)
17Ø PROCbox(5,3,6,16,14)
18Ø PROCmsg1("TELETEXT PROCEDURES",2,1Ø,2)
19Ø PROCmsg1("Procedure",2,5,5)
2ØØ PROCmsg1("Function",2,2Ø,5)
21Ø PROCmsg1("msg1",Ø,5,7)
22Ø PROCmsg1("Single height text",Ø,18,7)
23Ø PROCmsg1("msg2",Ø,5,9)
24Ø PROCmsg1("Double height text",Ø,18,9)
25Ø PROCmsg1("back",Ø,5,11)
26Ø PROCmsg1("Paint background",Ø,18,11)
27Ø PROCmsg1("blank",Ø,5,13)
28Ø PROCmsg1("Clear a line",Ø,18,13)
29Ø PROCmsg1("line",Ø,5,15)
3ØØ PROCmsg1("Draw a line",Ø,18,15)
31Ø PROCmsg1("box",Ø,5,17)
32Ø PROCmsg1("Draw a box",Ø,18,17)
33Ø PROCback(Ø,23,23)
34Ø REPEAT
35Ø PROCmsg1("Press Escape to exit",2,1Ø,23)
36Ø Z%=INKEY(1ØØ)
37Ø PROCblank(5,23,3Ø)
38Ø Z%=INKEY(1ØØ)
39Ø UNTIL FALSE
4ØØ END
41Ø:
9ØØ ON ERROR OFF:MODE 7
91Ø IF ERR<>17 THEN REPORT:PRINT" at line
";E RL
92Ø END
```

This concludes our series on the use of the Teletext Mode. With the ideas and techniques described in previous articles in this series, and the set of procedures presented this month, you should now be able to exploit more fully the potential of the Teletext Mode when writing your own programs, and produce attractive screen displays. For some further ideas on the use of the Teletext Mode, see the article 'Large Digital Display in Mode 7' elsewhere in this issue of BEEBUG.

Tested on O.S. 0·1 and 1·2 and on Basics I and II

WHICH OPERATING SYSTEM?

by David A. Fell

Most of you will probably be aware that *FXØ will return a message which tells you which operating system you have. What happens, however, if you wish to write one piece of software that will automatically adapt itself to the currently installed operating system, and even tell if it is running in a Beeb or an Electron?

The answer is provided by means of an extension of the negative inkey function, implemented by OSBYTE call number 129 (&81). If this is called with -256 (X=&ØØ, Y=&FF on entry), then the value returned allows the user to check and see what operating system version is active in any machine. The following function could be adapted to any program:

```
   1ØØØ DEF FNos
   1Ø1Ø LOCAL A%,X%,Y%
   1Ø2Ø A%=129
   1Ø3Ø X%=Ø
   1Ø4Ø Y%=255
   1Ø5Ø A%=(&FFFFØØ AND USR&FFF4)DIV
&1ØØ
   1Ø6Ø IF A% AND &8ØØØ A%=A% OR
&FFFFØØØØ
   1Ø7Ø =A%
```

and returns a value indicating which operating system is installed. By comparing the value returned with the values in the table below, you can tell which operating system is running. If you type the function in and then enter

```
   PRINT FNos
```

you should see the result for own machine.

	Value Returned:	
M.O.S.	DECIMAL	HEX
======	=======	===
Ø.1	Ø	&ØØØØØØØØ
1.Ø/1.2	-1	&FFFFFFFF
Electron	1	&ØØØØØØØ1
USA MOS	254	&ØØØØØØFE

For software to modify itself to the appropriate standards, it merely needs to check the value returned, and set a flag accordingly. This can have real applications in telling the American operating system apart from the British one, because the American Beebs have less lines of display on each screen, and thus software would need to adjust its output so as not to scroll data off the screen. Another use could be to make software realise if it is running in an Electron, and if so, not to use Mode 7, as the Electron does not support this mode.

A further way to distinguish operating systems is via the value that is normally inserted into the 'unused' OSBYTE location for OSBYTE 24Ø. If this is a 'Ø', then it is a 'UK' operating system; '1' is currently allocated to 'USA' operating systems, and it is expected that other versions of the operating system for other countries will have other values here. (For example, there is a German version of the Beeb being developed, and this may well store, say, a '2' at the OSBYTE 24Ø location.)

We are grateful to Acorn Computers for the information on which these notes are based, although we have not been able to test the routine described on a current USA MOS.

Tested on O.S. 0·1 and 1·2 and on Basics 1 and II

A DISASSEMBLER FOR THE BBC MICRO

by D. Turner

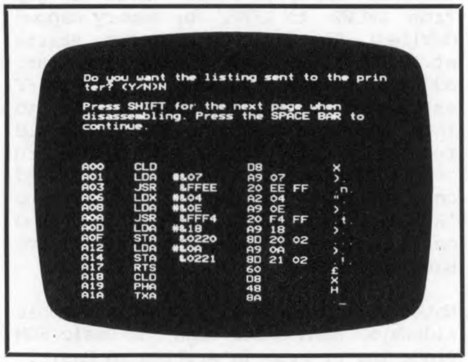
If you want to decipher machine code programs for which the source code is not available, then you need a disassembler. The program described here will help you to restore your own assembler programs, and will also allow you to look at the details of machine code games, or the workings of Basic and the Beeb's operating system.

Machine code programs are usually written in a format called 'Assembler' in which instructions are represented by easy to remember mnemonic codes, labels are usually simple but meaningful English words and variables are likewise very readable.

The assembler, or source code, is then converted into machine code by the Beeb's built in assembler, the result usually being presented as a sequence of hexadecimal numbers and hence thoroughly unreadable to all but the most technically minded expert.

Occasionally, through some mishap, you may find that the assembled machine code version is all you have left of a particular program. This is where a disassembler will come to your aid, by converting the hexadecimal code back into a fairly readable mnemonic format. With a disassembler you can try to fathom out machine code games and even try disassembling the Beeb's operating system or the Basic interpreter. This is certainly how some of the more detailed technical information about the Beeb has been obtained. On the other hand, no disassembler can precisely reproduce the original source code and you will still need to interpret some of the output for it to make sense.

The output from the disassembler consists of a listing of the program being disassembled, with mnemonic codes for instructions inserted where possible. The appropriate hexadecimal value is displayed alongside the disassembled machine code, as is an ASCII representation of the bytes that have been read in. When being displayed here, the ASCII value has the top bit removed. This is because some programs contain tables in which the setting of the top bit indicates the last letter



```
Do you want the listing sent to the prin
ter? (Y/N)N

Press SHIFT for the next page when
disassembling. Press the SPACE BAR to
continue.


A00     CLD            D8          X
A01     LDA  #&07      A9 07       >.
A03     JSR   &FFEE    20 EE FF    .n.
A06     LDX  #&04      A2 04       ".
A08     LDA  #&0E      A9 0E       >.
A0A     JSR   &FFF4    20 F4 FF    .t.
A0D     LDA  #&18      A9 18       >.
A0F     STA   &0220    8D 20 02    . .
A12     LDA  #&0A      A9 0A       >.
A14     STA   &0221    8D 21 02    .!.
A17     RTS            60          £
A18     CLD            D8          X
A19     PHA            48          H
A1A     TXA            8A          .
                                   _
```

in a word, but the remaining seven bits are still the valid ASCII code for that letter. The ASCII representation can be very helpful in identifying any text included as data within the machine code program.

Dissassemblers are quite widely available, often as a part of a larger package such as the BEEBUGSOFT EXMON. Here we present a straightforward example of a disassembler that will cope adequately with most situations. This one is in Basic, though some of those commercially available are themselves in machine code. You will see that a large part of the program is made up of all the mnemonic codes recognised by the assembler. Type the disassembler into your micro and save it away on cassette or disc.

To use the disassembler, *LOAD the machine code to be disassembled at its normal address. Then, reset PAGE to a value above (or below) the machine code, so that it will not be overwritten by the disassembler which should now be loaded using:

CHAIN"DISASS"

Once the disassembler is loaded and running, it will ask you for the start and end addresses of the machine code area of memory to be disassembled. The

➽

program will also ask whether the disassembled code is to be displayed only on the screen, or in addition sent to a printer.

If you want to examine the Beeb's operating system this starts at &C000 (although the first part of this is the default character definitions) and continues on up to &FFFF (with a gap from &FC00 to &FEFF for memory mapped devices). The Basic interpreter starts at &8000. If you give this as the start address to the disassembler, and &BFFF as the end address you will be able to look at Basic disassembled. You should recognise quite quickly the Acorn copyright message, which is displayed on the screen in response to typing 'REPORT' when you have just switched on, and also a complete table of the Basic keywords.

Note: This program will not disassemble sideways ROMs other than the Basic ROM since the program is written in Basic.

```
   10 REM Program: DISASSEMBLER
   20 REM Version: B2.2
   30 REM Author : L.P.Baxter
   40 REM BEEBUG Jan/Feb 1984
   50 REM Program subject to copyright.
   60 :
   70 MODE 7
   80 PROCinit
   90 INPUT''''"Enter start address (pr
ecede by '&' for hex)"'start$
  100 REPEAT
  110 IF start$="" THEN N%=PAGE ELSE N%
=EVAL(start$)
  120 UNTIL NOT(N%<0 OR N%>&FFFF)
  130 INPUT'"Enter the end address (pre
cede by '&'   for hex)"'finish$
  140 REPEAT
  150 IF finish$="" THEN E%=&FFFF ELSE
E%=EVAL(finish$)
  160 UNTIL E%>N% AND E%<&10000
  170 PRINT'"Do you want the listing se
nt to the printer? (Y/N)";
  180 REPEAT
  190 Q%=GET AND &DF
  200 UNTIL Q%=ASC"Y" OR Q%=ASC"N"
  210 VDUQ%
  220 PRINT''"Press SHIFT for the next
page when"'"disassembling. Press the SP
ACE BAR to   continue.";
  230 REPEAT UNTIL GET=32
  240 IF Q%=ASC"Y" C1$=CHR$32:C2$=CHR$3
2:VDU2 ELSE C1$=CHR$131:C2$=CHR$134:VDU
14
  250 PRINT'''
  260 REPEAT
  270 X$=MID$(OPCO$(?N%),5,3)
  280 PRINT;~N%;C2$;SPC3; LEFT$(OPCO$(?
N%),3);
  290 IF X$="IMP" THEN PROCIMP
  300 IF X$="ACC" THEN PROCACC
  310 IF X$="ABS" THEN PROCABS
  320 IF X$="ZRO" THEN PROCZRO
  330 IF X$="IMM" THEN PROCIMM
  340 IF X$="ABX" THEN PROCABX
  350 IF X$="ABY" THEN PROCABY
  360 IF X$="IDX" THEN PROCIDX
  370 IF X$="IDY" THEN PROCIDY
  380 IF X$="ZRX" THEN PROCZRX
  390 IF X$="ZRY" THEN PROCZRY
  400 IF X$="REL" THEN PROCREL
  410 IF X$="IND" THEN PROCIND
  420 IF X$="HEX" THEN PROCHEX
  430 IF X$="ASC" THEN PROCASC
  440 UNTIL N%>=E%
  450 VDU3,15
  460 END
  470 DEFPROCIMP
  480 PRINTCHR$32;
  490 N%=N%+1:PROCHEXY(0)
  500 :
  510 ENDPROC
  520 :
  530 DEFPROCACC
  540 PRINT;SPC4;"A";
  550 PROCIMP
  560 ENDPROC
  570 :
  580 DEFPROCABS
  590 PRINT;SPC3;"&";:PROCBYTE(2):PRINT
CHR$32;:PROCHEXY(2)
  600 ENDPROC
  610 :
  620 DEFPROCZRO
  630 PRINT;SPC3;"&";:PROCBYTE(1):PRINT
CHR$32;:PROCHEXY(1)
  640 ENDPROC
  650 :
  660 DEFPROCIMM
  670 PRINT;SPC2;"#&";:PROCBYTE(1):PRIN
TCHR$32;:PROCHEXY(1)
  680 ENDPROC
  690 :
  700 DEFPROCABX
  710 PRINT;SPC3;"&";:PROCBYTE(2):PRINT
",X";:PROCHEXY(2)
  720 ENDPROC
  730 :
  740 DEFPROCABY
  750 PRINT;SPC3;"&";:PROCBYTE(2):PRINT
",Y";:PROCHEXY(2)
  760 ENDPROC
  770 :
  780 DEFPROCIDX
  790 PRINT;SPC2;"(&";:PROCBYTE(1):PRIN
T",X)";:PROCHEXY(1)
```

➧

```
  80Ø ENDPROC
  81Ø :
  82Ø DEFPROCIDY
  83Ø PRINT;SPC2;"(&";:PROCBYTE(1):PRIN
T"),Y";:PROCHEXY(1)
  84Ø ENDPROC
  85Ø :
  86Ø DEFPROCZRX
  87Ø PRINT;SPC3;"&";:PROCBYTE(1):PRINT
",X";:PROCHEXY(1)
  88Ø ENDPROC
  89Ø :
  9ØØ DEFPROCZRY
  91Ø PRINT;SPC3;"&";:PROCBYTE(1):PRINT
",Y";:PROCHEXY(1)
  92Ø ENDPROC
  93Ø :
  94Ø DEFPROCREL
  95Ø LOCALA,B
  96Ø A=?(N%+1)
  97Ø IFA>127THEN B=1 ELSE B=Ø
  98Ø PRINT;SPC3;"&";~(N%+A+2-B*256);
  99Ø N%=N%+2
 1ØØØ PROCHEXY(1)
 1Ø1Ø ENDPROC
 1Ø2Ø :
 1Ø3Ø DEFPROCIND
 1Ø4Ø PRINT;SPC2;"(&";:PROCBYTE(2):PRIN
T")";:PROCHEXY(2)
 1Ø5Ø ENDPROC
 1Ø6Ø DEFPROCHEX
 1Ø7Ø N%=N%+1
 1Ø8Ø PROCHEXY(Ø)
 1Ø9Ø ENDPROC
 11ØØ :
 111Ø DEFPROCASC
 112Ø N%=N%+1
 113Ø PROCHEXY(Ø)
 114Ø ENDPROC
 115Ø :
 116Ø DEFPROCBYTE(T%)
 117Ø LOCALB%
 118Ø B%=T%
 119Ø REPEAT
 12ØØ IF ?(N%+T%)<16 THEN PRINT;"Ø";
 121Ø PRINT;~?(N%+T%);
 122Ø T%=T%-1
 123Ø UNTIL T%<1
 124Ø N%=N%+B%+1
 125Ø ENDPROC
 126Ø :
 127Ø DEFPROCHEXY(Y)
 128Ø PRINTTAB(2Ø,VPOS);C1$;
 129Ø FORJ=N%-(Y+1)TO(N%-1)
 13ØØ VDU32
 131Ø IF?J<16THENPRINT;"Ø";
 132Ø PRINT;~?J;
 133Ø NEXT
 134Ø PRINTTAB(33,VPOS);
 135Ø FORJ=N%-(Y+1)TO(N%-1)
 136Ø PROCoutbyte(?J AND 127)
 137Ø NEXT
 138Ø PRINT
 139Ø ENDPROC
 14ØØ :
 141Ø DEF PROCoutbyte(I%)
 142Ø I%=I% AND 127
 143Ø IF I%=127 I%=Ø
 144Ø IF I%>32 VDU I% ELSE VDU 46
 145Ø ENDPROC
 146Ø :
 147Ø DEF PROCinit
 148Ø PROCnice("BEEBUG Disassembler.")
 149Ø PROCnice("By L.P.Baxter.")
 15ØØ DIM OPCO$(255)
 151Ø FOR I=ØTO255
 152Ø READ OPCO$(I)
 153Ø NEXT
 154Ø ENDPROC
 155Ø DATA BRK IMP,ORA IDX,??? HEX
 156Ø DATA ??? HEX,??? HEX,ORA ZRO
 157Ø DATA ASL ZRO,??? HEX,PHP IMP
 158Ø DATA ORA IMM,ASL ACC,??? HEX
 159Ø DATA ??? HEX,ORA ABS,ASL ABS
 16ØØ DATA ??? HEX,BPL REL,ORA IDY
 161Ø DATA ??? HEX,??? HEX,??? HEX
 162Ø DATA ORA ZRX,ASL ZRX,??? HEX
 163Ø DATA CLC IMP,ORA ABY,??? HEX
 164Ø DATA ??? HEX,??? HEX,ORA ABX
 165Ø DATA ASL ABX,??? HEX,JSR ABS
 166Ø DATA AND IDX,??? ASC,??? ASC
 167Ø DATA BIT ZRO,AND ZRO,ROL ZRO
 168Ø DATA ??? ASC,PLP IMP,AND IMM
 169Ø DATA ROL ACC,??? ASC,BIT ABS
 17ØØ DATA AND ABS,ROL ABS,??? ASC
 171Ø DATA BMI REL,AND IDY,??? ASC
 172Ø DATA ??? ASC,??? ASC,AND ZRX
 173Ø DATA ROL ZRX,??? ASC,SEC IMP
 174Ø DATA AND ABY,??? ASC,??? ASC
 175Ø DATA ??? ASC,AND ABX,ROL ABX
 176Ø DATA ??? ASC,RTI IMP,EOR IDX
 177Ø DATA ??? ASC,??? ASC,??? ASC
 178Ø DATA EOR ZRO,LSR ZRO,??? ASC
 179Ø DATA PHA IMP,EOR IMM,LSR ACC
 18ØØ DATA ??? ASC,JMP ABS,EOR ABS
 181Ø DATA LSR ABS,??? ASC,BVC REL
 182Ø DATA EOR IDY,??? ASC,??? ASC
 183Ø DATA ??? ASC,EOR ZRX,LSR ZRX
 184Ø DATA ??? ASC,CLI IMP,EOR ABY
 185Ø DATA ??? ASC,??? ASC,??? ASC
 186Ø DATA EOR ABX,LSR ABX,??? ASC
 187Ø DATA RTS IMP,ADC IDX,??? ASC
 188Ø DATA ??? ASC,??? ASC,ADC ZRO
 189Ø DATA ROR ZRO,??? ASC,PLA IMP
 19ØØ DATA ADC IMM,ROR ACC,??? ASC
 191Ø DATA JMP IND,ADC ABS,ROR ABS
 192Ø DATA ??? ASC,BVS REL,ADC IDY
 193Ø DATA ??? ASC,??? ASC,??? ASC
 194Ø DATA ADC ZRX,ROR ZRX,??? ASC
 195Ø DATA SEI IMP,ADC ABY,??? ASC
```

»

```
1960 DATA ??? ASC,??? ASC,ADC ABX
1970 DATA ROR ABX,??? ASC,??? HEX
1980 DATA STA IDX,??? HEX,??? HEX
1990 DATA STY ZRO,STA ZRO,STX ZRO
2000 DATA ??? HEX,DEY IMP,??? HEX
2010 DATA TXA IMP,??? HEX,STY ABS
2020 DATA STA ABS,STX ABS,??? HEX
2030 DATA BCC REL,STA IDY,??? HEX
2040 DATA ??? HEX,STY ZRX,STA ZRX
2050 DATA STX ZRX,??? HEX,TYA IMP
2060 DATA STA ABY,TXS IMP,??? HEX
2070 DATA ??? HEX,STA ABX,??? HEX
2080 DATA ??? HEX,LDY IMM,LDA IDX
2090 DATA LDX IMM,??? HEX,LDY ZRO
2100 DATA LDA ZRO,LDX ZRO,??? HEX
2110 DATA TAY IMP,LDA IMM,TAX IMP
2120 DATA ??? HEX,LDY ABS,LDA ABS
2130 DATA LDX ABS,??? HEX,BCS REL
2140 DATA LDA IDY,??? HEX,??? HEX
2150 DATA LDY ZRX,LDA ZRX,LDX ZRY
2160 DATA ??? HEX,CLV IMP,LDA ABY
2170 DATA TSX IMP,??? HEX,LDY ABX
2180 DATA LDA ABX,LDX ABY,??? HEX
2190 DATA CPY IMM,CMP IDX,??? HEX
2200 DATA ??? HEX,CPY ZRO,CMP ZRO
2210 DATA DEC ZRO,??? HEX,INY IMP
2220 DATA CMP IMM,DEX IMP,??? HEX
2230 DATA CPY ABS,CMP ABS,DEC ABS
2240 DATA ??? HEX,BNE REL,CMP IDY
2250 DATA ??? HEX,??? HEX,??? HEX
2260 DATA CMP ZRX,DEC ZRX,??? HEX
2270 DATA CLD IMP,CMP ABY,??? HEX
2280 DATA ??? HEX,??? HEX,CMP ABX
2290 DATA DEC ABX,??? HEX,CPX IMM
2300 DATA SBC IDX,??? HEX,??? HEX
2310 DATA CPX ZRO,SBC ZRO,INC ZRO
2320 DATA ??? HEX,INX IMP,SBC IMM
2330 DATA NOP IMP,??? HEX,CPX ABS
2340 DATA SBC ABS,INC ABS,??? HEX
2350 DATA BEQ REL,SBC IDY,??? HEX
2360 DATA ??? HEX,??? HEX,SBC ZRX
2370 DATA INC ZRX,??? HEX,SED IMP
2380 DATA SBC ABY,??? HEX,??? HEX
2390 DATA ??? HEX,SBC ABX,INC ABX
2400 DATA ??? HEX
2410 :
2420 DEF PROCnice(A$)
2430 A$=CHR$141+CHR$132+CHR$157+CHR$13
4+A$+"  "+CHR$156
2440 LOCAL I%
2450 FORI%=0TO1
2460 PRINTTAB((40-LENA$)DIV2);A$
2470 NEXT
2480 ENDPROC
```

INTERRUPT PROGRAMMING UPDATE

by David A. Fell

In the November issue of BEEBUG (Vol.2 No.6) we published an introductory article on Interrupt Programming. We are now able to provide some additional information on a number of points that were touched upon in the article.

Firstly, however, we would like to point out a small correction to our example program. Line 1030 should be changed to read:

```
1030 P%=&D00 (for cassette users.)
1030 P%=&A00 (for disc users.)
```

You will then find that the musical keyboard, for example, continues to function even when another program is loaded, as this no longer overwrites the machine code routine. This assumes that disc users will not be using cassettes or the RS423 port for input.

The main point concerns the saving of registers when using interrupts and events. When using interrupts, it is ESSENTIAL to save ALL registers correctly. This is achieved with the following series of assembler statements:

```
PHP:PHA:TXA:PHA:TYA:PHA
```

Note that the processor status register is saved first. The registers are subsequently restored by the following assembler statements:

```
PLA:TAY:PLA:TAX:PLA:PLP
```

The article was concerned primarily with events rather than interrupts. An event is a 'packaged' form of interrupt which is much easier to use. In this case there is no strict need to save all registers. The accumulator is set to the event number, essential if more than one event has been enabled. In our examples, only the X register needs to be preserved on entry to the routine:

```
TXA:PHA
```

and restored on exit: PLA:TAX

This is contrary to the information given in the Advanced User Guide which states, "The event handling routine should preserve all registers." (p.288). This is not quite correct, and the user should find that, if any, only the X register need be saved. We are grateful to Paul Beverley for bringing these points to our attention.

FOUR GRAPHICS TABLETS

reviewed by Russell Tubb

Although the BBC micro is capable of excellent graphics, the standard keyboard is not an ideal device for entering graphics designs and drawings. This is much more easily achieved by some form of digitizer, such as a graphics tablet, together with some good supporting software. This month, Russell Tubb reviews four of the graphics tablets currently available.

MODELS REVIEWED:
Beebplotter
Watford Electronics
Price :£ 84.00

Graphics Tablet
G.T. Norton
Price :£35.00

CAD-GET
Acedemic Software
Price :£78.50
(A left-hand version of the tablet is available for the same cost as the right-hand version. Educational orders will receive a 10% discount.)

P.L. Digitizer
B.S. Dollarmore Ltd.
Price :£149.95

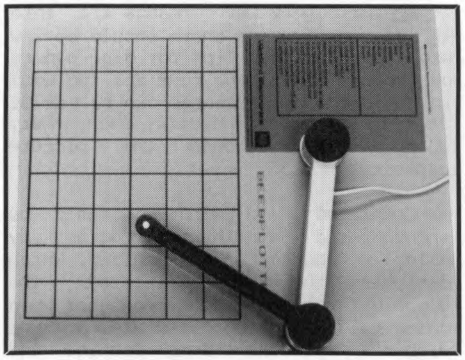
(All prices quoted are inclusive of VAT, where applicable, and delivery. For addresses of suppliers see at end of reviews.)

INTRODUCTION
Graphics tablets or digitizers are one relatively cheap and easy way of transfering freehand drawings or sketches into the computer for manipulation, storage and printing.

Nearly all those currently available work on the pantograph system, which consists of a pivoted arm with variable resistors at the 'shoulder' and 'elbow' whose resistance is directly proportional to the position of the 'hand' on the drawing area. Movements can be reproduced on the screen and special time saving routines such as drawing a rectangle from only two points can be incorporated into the software.

All the models use the analogue input port and need a calibration routine to set them up correctly (the CAD-GET is delivered already calibrated).

The useful drawing area of these tablets is just over A4 size and they all have a graticule related to that size printed on the baseboard (or in one case, as an overlay sheet). The quality of design and construction is important but the effectiveness of this type of device depends ultimately on the accompanying software. We shall look at both hardware and software in this review.



BEEBPLOTTER from Watford Electronics.
The striking thing about this model is the high quality construction. The baseboard is 40cm (16") x 48cm (19") smoked Perspex with a strong two-tone arm in similar colours to the BBC Micro. The movement is very smooth and positive.

The software is of a high standard and responds quickly, despite being in Basic (except the infill routine), and is easy to use. A drawing is produced by a sequence of commands, which are displayed in the bottom three lines of the screen as they are entered. Commands include: rectangle, line, circle, follow mode, print at.., move

➜

and redraw, define colours, alter drawing colour, wipe screen, save, edit and infill. These are all entered via the keyboard. When drawing lines, circles or rectangles, the shape is constantly plotted on the screen, so that it may be positioned exactly, and then 'fixed'. All eight steady colours are available, and logical colour changes also are possible (ie VDU 19).

Drawings may be saved and stored as a string of corresponding commands rather than a straight screen dump. This allows two very useful features: 'edit' and 'move and redraw'. 'Edit' clears the screen and displays a list of the commands which describe how each step of the picture was built up, giving the option of skipping over those steps no longer required. 'Move and redraw' allows all or some of the picture to be copied to another part of the screen.

Approximately 200 commands fit into 1k of memory, so there should be few problems of space. Tips for disc owners on how to obtain more room are included in the handbook. Pictures can be saved at any stage and overlay each other, which further extends the picture capacity.

Completed drawings can be incorporated into other programs by using a short routine supplied, which deciphers the command string. The Beebplotter need not be connected when doing this. Also supplied is a tape with a picture dump routine and programs in the manual for Acorn/Seikosha GP80/GP100 and Epson MX80 printers. In practice, this system proved both simple to understand and easy to use.

GRAPHICS TABLET from G.T. Norton

This tablet measures 46cm (18") x 41cm (16") and is finished in white Formica with a wood coloured, square section plastic arm. The whole unit does not appear to be too rugged and has a decidedly unprofessional appearance. The arm has a slight upward movement, allowing thick books to be traced easily through the sighting hole at the free end of the arm. There is also a utility button at this end of the arm which serves several functions.

The software for this tablet uses a combination of function keys, alpha-numeric and the utility key to select the desired option and at several stages either may be used. For example, there are coloured blocks down the left hand side of the tablet, which can be used to select background and foreground colours.

Unfortunately the program is slow to respond to all commands, as lengthy key depressions are needed to select the functions available (select colour, draw, line, triangle, rectangle, ellipse, fill, title, erase, abort, help, save and retrieve).

The picture is not saved as it is built up, so that clearing the screen accidentally will lose everything. The save routine simply *SAVEs the screen memory for the mode it is in, and can be retrieved for further modification. Editing is done, rather crudely, by a character sized, background coloured, block which clears anything it passes over. The fill routine works at machine code speed, but only if the background is initially black. However, on the review model, it did not appear to work at all.

The program is in Basic (apart from the fill routine) but is unLISTable. The completed screens can be used separately by *LOADing them in the appropriate mode, although the manual gives no advice on this point.

CAD-GET from Academic Software

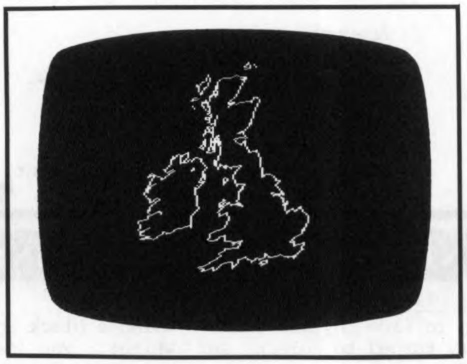
The Computer Aided Design Graphics Entry Tablet is, as the maker's name implies, aimed primarily at the education market. Indeed it includes many features that make it ideal for schools, but it appears equally well suited to home or office use.

Mounted on a clear Perspex base 46cm (18") x 31cm (12") with a brushed aluminium arm, each CAD-GET comes pre-calibrated, though it may be recalibrated, by software, for a different drawing area if required. It is also available as a left-handed model on request, and thumbwheels are an optional extra.

Due to the size of the operating program only modes 4 and 5 are available to the user. The colours are selected from a palette at the top of the baseboard.

All of the drawing commands are accessible from the tablet and a summary of them is also provided on the backplane. They include: erase, redraw, text, trace, line/dot, rectangle, arc/circle, polygon, fill/draw. When drawing, two buttons at the bottom left are used to select commands, move and plot. Editing is achieved by deleting previous steps.

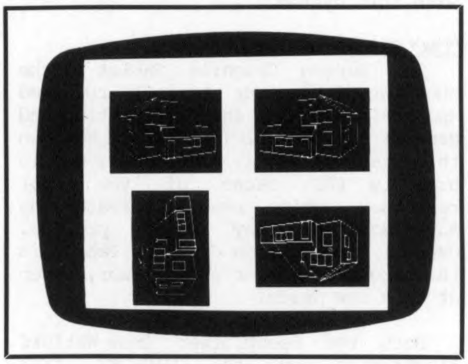
Other options provided by the function keys are run, joystick or thumb wheel, reset, print, save, load, accuracy, mode and end. If the optional thumbwheels are fitted, you can switch between these and the digitiser arm for input. The thumbwheels operate in the same way as the arm, but give much more accurate results for parallel lines and right angles.



Technical details of the program are given in the manual, together with information on using the designs produced in other programs. If hard copy is required, a printer dump routine must be appended to the program but the key to control this is already incorporated in the software. A number of demonstration displays are included with the software such as the outline map of the British Isles illustrated. Again this system worked well.

P.L. DIGITIZER from B.S. Dollarmore

The P.L. Digitizer measures 31cm (12") x 56 (22") and is finished in a simulated wood Formica. The top half, containing the mechanism, is enclosed and quite bulky in appearance. The base is accurately marked with the BBC micro graphics units with a plastic cover to place drawings under. The whole screen may be used for the design.



The software for the tablet is comprehensive and complex. Various documents are provided to explain its uses, including exercise diagrams, a very detailed manual, function key template and a quick reference card, which is essential considering the number of commands available.

Following the calibration routine, either mode 4 or 5 is selected, and a function key is pressed to start drawing or to read a file. Eight example files are provided on the tape and these may be loaded using a short display routine which could also be used in other programs.

There are well over fifty commands available to the user. These include line, box, draw, circle, fill (very fast), define colours (GCOL and VDU 19 - all colours, including flashing, available), text and line thickness. Some quite sophisticated manipulation routines, such as duplicate, reflect and rotate, are provided and the software is also able to remember a number of cursor positions, which you can then return to at the press of a key.

Other features include the ability to display user defined characters (a program is included for designing these using the tablet), halt markers making

➧

simple animation possible by delaying the next instruction, and parallel lines which can be quickly created and used for hatching and texture. The sequence of demonstration displays and animations is a good indication of the impressive results that can be achieved with this system.

CONCLUSIONS

The Norton Graphics Tablet is the cheapest of the four tablets reviewed and this shows in the construction and general appearance. The utility key on this tablet is a nice idea, but is probably the cause of the poor response, which really defeats any advantage the key might provide. Overall the Norton Graphics Tablet is the bare minimum for a digitizer, even at this low price.

Both the Beebplotter from Watford Electronics and the CAG-GET from Acedemic Software perform very satisfactorily. The Beebplotter looks very profressional and has good supporting documentation. The CAD-GET tablet is also well produced, though without quite the robustness of the Beebplotter, while the optional thumbwheels could be very useful in some applications. A compatible joystick is also available for use with the CAD-GET board as an alternative input device (the cost is £35.ØØ).

The P.L. Digitizer, from Dollarmore with its many advanced features is the 'Rolls Royce' amongst these graphics tablets, though you do have to pay substantially more for the extra software facilities provided. It is a pity that the general appearance of this unit is not more attractive, though the solid construction may appeal to schools and colleges. The manual and other aids to using this system have been produced to a high standard. The system certainly justifies its higher price though, unless the additional features are essential to your needs, the two tablets by Watford Electronics and Academic Software provide good value at a much lower price.

Suppliers:

Watford Electronics,
33/35 Cardiff Road,
Watford, Herts WD1 8ED.

G.T.Norton,
Dormers, Selsey Road,
Donnington, Chichester.

Academic Software,
Sourby Old Farm,
Timble, Otley, W.Yorks LS21 2PW.

B.S.Dollarmore Ltd,
Burton Road,
Castle Gresley, Burton upon Trent,
Staffordshire DE11 9HA.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FADING PICTURES IN AND OUT ON A MONOCHROME DISPLAY

If you want an object on the screen to fade in, and you are using a black and white TV or monitor (or indeed, a colour TV turned to black and white), you can successively change the object from dark (black) through to light (white) by changing the object's colour. The range of shading relating to the colours (from dark to light) is: black, blue, red, magenta, green, cyan, yellow, white

Try this short program which plots a square on the screen and continually fades it in and out using this technique. The colour is changed in line 9Ø using a VDU 19 call. See the User Guide p.382 for further details.

```
 1Ø MODE 2:VDU 23,1,Ø;Ø;Ø;Ø;:GCOLØ,1
 2Ø MOVE 1Ø24,Ø:PLOT85,1Ø24,128Ø
 3Ø MOVE Ø,128Ø:PLOT85,Ø,Ø
 4Ø REPEAT
 5Ø FOR Y=15Ø TO 16Ø STEP 1Ø
 6Ø RESTORE Y
 7Ø FOR X=Ø TO 7
 8Ø READ COL
 9Ø VDU 19,1,COL;Ø;
11Ø TIME=Ø:REPEAT:UNTIL TIME>1Ø
12Ø NEXT X,Y
13Ø UNTIL FALSE
14Ø END
15Ø DATA Ø,4,1,5,2,6,3,7:REM FADE IN
16Ø DATA 7,3,6,2,5,1,4,Ø:REM FADE OUT
```

(Note: Do not renumber this program otherwise the calculated RESTORE in line 6Ø will not work.)

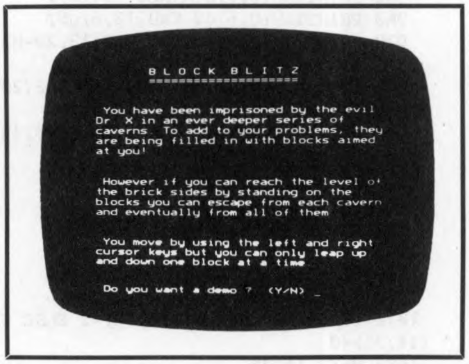
Tested on O.S. 0·1 and 1·2 and on Basics I and II

BLOCK BLITZ (32K)

by D. J. Pilling

The following program is one of the most entertaining games we have seen for some time and will well repay the effort spent in typing it into your micro. It uses colour, graphics and sound to produce a fast and original action game that compares very favourably with many of the commercial games currently available.

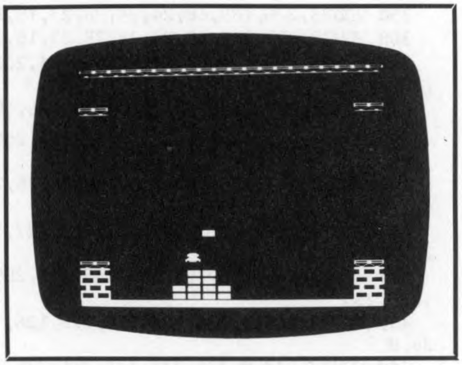
In this game, you are trapped in a series of caverns, from which you have to try and escape whilst being bombarded by blocks, dropped from above by a very mobile crane. As the blocks fall towards you, you must try to dodge them if you want to stay alive, but your only way of making good your escape is to climb onto the blocks, and up to the height of the walls. The crane is quite intelligent, and will try and drop the blocks on you, giving you the chance to practice some risky brinkmanship as you lure the blocks into the places that will best help you make your exit. If you succeed, a gantry extends from one of the walls, allowing you to escape. You are then thrown into an even deeper cavern!



```
B L O C K   B L I T Z
=====================

 You have been imprisoned by the evil
Dr. X in an ever deeper series of
caverns. To add to your problems, they
are being filled in with blocks aimed
at you!

 However if you can reach the level of
the brick sides by standing on the
blocks you can escape from each cavern
and eventually from all of them

 You move by using the left and right
cursor keys but you can only leap up
and down one block at a time

 Do you want a demo ?  (Y/N) _
```

You move the man using the left and right cursor keys, and he will also climb over any blocks in the way, provided he doesn't have to step up or down more than one block at a time. You will need to be very alert to avoid being trapped on a column, or between two high walls of blocks.

If you should find the program too slow for your reactions, try changing line 180 to:-



```
180 PROCbl:PROCm
```

You will now need to react to the rapidly changing situation with the speed of lightning!

```
 10 REM PROGRAM BLOCK BLITZ
 20 REM VERSION B0.4
 30 REM AUTHOR  D.J.PILLING
 40 REM BEEBUG  JAN/FEB 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
 70 ON ERROR GOTO 250
 80 DIM S%(19,31),MAN$(5,1)
 90 PROCchar
100 REPEAT
110 MODE 7
120 PROCsetupuser
130 MODE5
140 PROCsetup
150 REPEAT
160 PROCscreen
170 REPEAT
180 PROCbl:PROCm:PROCd(2)
190 UNTIL end%
200 PROCgame
210 UNTIL demo%
220 UNTIL FALSE
230 END
240 :
250 ON ERROR OFF:MODE 7
260 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
270 END
```

»

```
  280 :
  290 DEF PROCchar
  300 a$=CHR$8:b$=CHR$10:c$=CHR$11
  310 d$=CHR$17:e$=CHR$32
  320 f$=STRING$(14,CHR$32):g$=STRING$(
5,CHR$32)
  330 VDU23,224,28,48,62,20,28,60,126,1
89
  340 VDU23,225,189,60,24,24,56,104,200
,76
  350 VDU23,226,189,60,24,24,28,23,18,24
  360 VDU23,227,189,60,24,24,28,23,18,48
  370 VDU23,228,189,60,24,24,56,104,200
,88
  380 VDU23,229,28,6,62,20,28,60,126,189
  390 VDU23,240,60,90,126,60,36,126,255
,189
  400 VDU23,243,255,129,255,129,66,36,2
4,255
  410 VDU23,244,0,127,127,127,127,127,1
27,0
  420 VDU23,245,255,255,255,255,255,255
,255,255
  430 VDU23,246,0,126,126,126,126,126,1
26,0
  440 VDU23,247,0,231,231,231,231,231,2
31,0
  450 VDU23,248,189,189,60,36,36,36,36,
102
  460 VDU23,249,119,34,34,127,127,65,65
,99
  470 VDU23,250,189,219,255,189,165,255
,255,60
  480 VDU23,251,60,60,60,36,36,36,36,102
  490 ENVELOPE1,3,0,2,0,0,255,0,127,0,0
,-127,80,80
  500 R$=d$+CHR$1:Y$=d$+CHR$2
  510 W$=d$+CHR$3
  520 M$=W$+CHR$240+b$+a$+R$+CHR$248
  530 MJ$=b$+e$+a$+c$+R$+CHR$251+a$+c$+
W$+CHR$250
  540 MK$=c$+e$+b$+a$+W$+CHR$240+a$+b$+
R$+CHR$248
  550 f$=e$+W$+CHR$224+b$+a$+a$+e$+R$
  560 MAN$(1,0)=f$+CHR$225
  570 MAN$(1,1)=f$+CHR$226
  580 f$=e$+a$+a$+W$+CHR$229+b$+e$+a$+a
$+R$
  590 MAN$(0,0)=f$+CHR$227
  600 MAN$(0,1)=f$+CHR$228
  610 f$=b$+e$+c$+a$+e$+R$:g$=a$+c$+W$+
CHR$224
  620 MAN$(3,0)=f$+CHR$225+g$
  630 MAN$(3,1)=f$+CHR$226+g$
  640 f$=e$+a$+b$+e$+W$+CHR$224+a$+b$+R$
  650 MAN$(5,0)=f$+CHR$225
  660 MAN$(5,1)=f$+CHR$226
  670 f$=b$+e$+c$+a$+e$+a$+a$+R$:g$=a$+
c$+W$+CHR$229
  680 MAN$(2,0)=f$+CHR$227+g$
  690 MAN$(2,1)=f$+CHR$228+g$
  700 f$=e$+a$+b$+e$+a$+a$+W$+CHR$229+a
$+b$+R$
  710 MAN$(4,0)=f$+CHR$227
  720 MAN$(4,1)=f$+CHR$228
  730 B$=R$+CHR$249+a$+b$+Y$+CHR$244
  740 TL$=e$+a$+a$+R$+CHR$249
  750 TR$=e$+R$+CHR$249
  760 BL$=TL$+a$+b$+Y$+CHR$244+e$
  770 BR$=TR$+a$+a$+b$+e$+Y$+CHR$244
  780 BD$=Y$+e$+a$+b$+CHR$244
  790 WALL$=CHR$246+CHR$246+c$+a$+a$+CH
R$247+CHR$247+c$+a$+a$
  800 HI%=0:SC%=0
  810 ENDPROC
  820 :
  830 DEF PROCsetup
  840 VDU23;11,0;0;0;0:VDU29,640;512;
  850 IF HI%<SC% THEN HI%=SC%
  860 H%=4:N%=1:M%=0:MEN%=3
  870 SC%=FNsc(H%):SCT%=0
  880 ENDPROC
  890 :
  900 DEF PROCscreen
  910 CLS:COLOUR1
  920 PRINTTAB(0,0)"H ";HI%;TAB(9);"S "
;SC%;TAB(17)"M ";MEN%
  930 COLOUR2
  940 g$=STRING$(2,CHR$243)
  950 PRINTTAB(0,1)STRING$(10,g$);
  960 PRINTTAB(0,6)g$ TAB(18,6)g$;
  970 PRINTTAB(0,29-H%)g$ TAB(18,29-H%)
g$;
  980 COLOUR2:PRINTTAB(0,30)STRING$(20,
CHR$245);
  990 PRINTTAB(0,30)STRING$(20,CHR$245);
 1000 COLOUR131:COLOUR1
 1010 g$=STRING$(H%/2,WALL$)
 1020 PRINTTAB(0,29)g$;
 1030 PRINTTAB(18,29)g$;
 1040 COLOUR128
 1050 FOR J%=0 TO 31
 1060 FOR I%=0 TO 19
 1070 IF J%=30 THEN S%(I%,J%)=2 ELSE S%
(I%,J%)=0
 1080 NEXT I%,J%
 1090 B%=28:A%=10:NG%=FALSE:end%=FALSE
 1100 PRINTTAB(A%,B%)M$
 1110 Y%=3:IF RND(2)=1 THEN  X%=0 ELSE
X%=19
 1120 ENDPROC
 1130 :
 1140 DEF PROCm
 1150 IF M%=0 THEN M%=1 ELSE M%=0
 1160 IF S%(A%,B%)<>0 THEN PROCng:ENDPR
OC
 1170 IF demo% THEN 1200
 1180 IF INKEY-26 THEN IF A%>2 THEN 128
0
```

```
 1190 IF INKEY-122 THEN IF A%<17 THEN 1
230
 1200 IF NOT demo% THEN PRINTTAB(A%,B%)
M$;:ENDPROC
 1210 IF Y%<=3 OR X%<>A% THEN PRINTTAB(
A%,B%)M$;:ENDPROC
 1220 IF A%=10 THEN 1280
 1230 IF S%(A%+1,B%)<>0 THEN ENDPROC
 1240 IF S%(A%+1,B%+1)=2 THEN PRINTTAB(
A%,B%)MAN$(3,M%):A%=A%+1:B%=B%-1:GOTO 1
320
 1250 IF S%(A%+1,B%+2)=2 THEN PRINTTAB(
A%,B%)MAN$(1,M%):A%=A%+1:GOTO 1320
 1260 IF S%(A%+1,B%+3)=2 THEN PRINTTAB(
A%,B%)MAN$(5,M%):A%=A%+1:B%=B%+1:GOTO 1
320
 1270 ENDPROC
 1280 IF S%(A%-1,B%)<>0 THEN ENDPROC
 1290 IF S%(A%-1,B%+1)=2 THEN PRINTTAB(
A%,B%)MAN$(2,M%):A%=A%-1:B%=B%-1:GOTO 1
320
 1300 IF S%(A%-1,B%+2)=2 THEN PRINTTAB(
A%,B%)MAN$(0,M%):A%=A%-1:GOTO 1320
 1310 IF S%(A%-1,B%+3)=2 THEN PRINTTAB(
A%,B%)MAN$(4,M%):A%=A%-1:B%=B%+1
 1320 IF S%(A%,B%)<>0 THEN PROCng:ENDPR
OC
 1330 IF B%=27-H% THEN PROCwin:ENDPROC
 1340 ENDPROC
 1350 :
 1360 DEF PROCng
 1370 end%=TRUE:NG%=TRUE:MEN%=MEN%-1
 1380 SOUND&0011,0,0,0:SOUND0,-15,5,12
 1390 PROCbonk
 1400 ENDPROC
 1410 :
 1420 DEF PROCd(T%)
 1430 T%=TIME+T%
 1440 REPEAT UNTIL TIME>T%
 1450 ENDPROC
 1460 :
 1470 DEF PROCwin
 1480 IF S%(X%,Y%)=1 THEN PRINTTAB(X%,Y
%)e$;:SOUND&0011,0,0,0
 1490 PRINTTAB(A%,B%)M$;
 1500 end%=TRUE:COLOUR2
 1510 IF A%=2 OR A%=17 THEN 1540
 1520 D%=A%
 1530 IF A%<9 THEN FOR C%=2 TO A%-1:PRI
NTTAB(C%,B%+2)CHR$243:SOUND2,-10,100,1:
PROCd(15):NEXT ELSE FOR C%=17 TO A%+1 S
TEP-1:PRINTTAB(C%,B%+2)CHR$243:SOUND2,-
10,100,1:PROCd(15):NEXT
 1540 REPEAT IFA%<9 PRINTTAB(A%,B%)MAN$
(0,M%):PROCd(15):A%=A%-1 ELSE PRINTTAB(
A%,B%)MAN$(1,M%):PROCd(15):A%=A%+1
 1550 IFM%=0 THEN M%=1 ELSE M%=0
 1560 UNTIL A%=1 OR A%=18
 1570 IF D%=2 OR D%=17 THEN 1630
 1580 COLOUR2:PRINTTAB(A%,B%)M$;
 1590 IF D%<9 THEN FOR C%=D%-1 TO 2 STE
P-1:PRINTTAB(C%,B%+2)e$:SOUND2,-10,100,
1:PROCd(15):NEXT ELSE FOR C%=D%+1 TO 17
:PRINTTAB(C%,B%+2)e$:SOUND2,-10,100,1:P
ROCd(15):NEXT
 1600 COLOUR0:COLOUR131
 1610 IF A%=1 PRINTTAB(3,B%-1)"YIPPEE"
ELSE PRINTTAB(11,B%-1)"YIPPEE"
 1620 COLOUR128
 1630 FOR I%=1 TO 10:PRINTTAB(A%,B%)MJ$
;:SOUND0,-15,4,1:PROCd(8):PRINTTAB(A%,B
%)MK$;:SOUND0,-15,6,1:PROCd(8):NEXT
 1640 IF A%=1 PRINTTAB(3,B%-1)SPC(6) EL
SE PRINTTAB(11,B%-1)SPC(6)
 1650 PROCd(150)
 1660 ENDPROC
 1670 :
 1680 DEF PROCbl
 1690 IF Y%=3 THEN 1800
 1700 IF T%>9 THEN 1720
 1710 IF T%>0 PRINTTAB(T%,2)TL$:T%=T%-1
:GOTO 1730 ELSE PRINTTAB(0,2)e$:GOTO 17
30
 1720 IF T%<19 PRINTTAB(T%,2)TR$:T%=T%+
1 ELSE PRINTTAB(19,2)e$
 1730 PRINTTAB(X%,Y%)BD$:S%(X%,Y%)=0:Y%
=Y%+1:S%(X%,Y%)=1:IF S%(X%,Y%+1)<>2 THE
N ENDPROC
 1740 S%(X%,Y%)=2
 1750 SOUND&0011,0,0,0:SOUND0,-15,4,1
 1760 IF T%<>0 AND T%<>19 Y%=Y%-1:ENDPR
OC
 1770 IF RND(1)>.5 THEN  X%=19 ELSE X%=0
 1780 SC%=SC%-10:PRINTTAB(11,0)SPC(5)TA
B(11,0)R$;SC%
 1790 Y%=3:PRINTTAB(X%,2)B$;:ENDPROC
 1800 SOUND2,-10,60,1
 1810 IF (A%=X% AND RND(1)>.5) THEN 1860
 1820 IF demo% THEN 1840
 1830 IF ABS(A%-X%)<2 THEN IF RND(1)>.8
THEN 1860
 1840 IF A%<X% THEN PRINTTAB(X%,2)BL$:X
%=X%-1 ELSE PRINTTAB(X%,2)BR$:X%=X%+1
 1850 ENDPROC
 1860 IF X%<2 OR X%>17 ENDPROC
 1870 T%=X%:SOUND1,1,350,90:GOTO1730
 1880 :
 1890 DEF PROCgame
 1900 IF NG% AND MEN%=0 THEN PROClost:P
ROCsetup:ENDPROC ELSE IF NG% THEN PROCc
ng:ENDPROC
 1910 N%=N%+1:H%=H%+2
 1920 IF H%=18 THEN PROCover:PROCsetup:
ENDPROC
 1930 PROCccg
 1940 ENDPROC
 1950 :
 1960 DEF FNsc(H%)=160*(H%+1)
 1970 :
```

➧

```
 198Ø DEF PROCr:REPEAT UNTIL INKEY-74:E
NDPROC
 199Ø :
 2ØØØ DEF PROCsetupuser
 2Ø1Ø VDU19,1,2,Ø,Ø,Ø,19,3,6,Ø,Ø,Ø
 2Ø2Ø PRINTTAB(7,2)CHR$131;"B L O C K
B L I T Z"
 2Ø3Ø PRINTTAB(7,3)CHR$129STRING$(2Ø,"=
")
 2Ø4Ø PRINTTAB(Ø,6)CHR$134;" You have b
een imprisoned by the evil"'CHR$134;"Dr
. X in an ever deeper series of"'CHR$13
4;"caverns. To add to your problems, th
ey"'CHR$134;"are being filled in with b
locks aimed"'CHR$134;"at you!"
 2Ø5Ø PRINT''CHR$13Ø;" However if you c
an reach the level of"'CHR$13Ø;"the bri
ck sides by standing on the"'CHR$13Ø;"b
locks you can escape from each cavern"'
CHR$13Ø;"and eventually from all of the
m."
 2Ø6Ø PRINT''CHR$133;" You move by usin
g the left and right"'CHR$133;"cursor k
eys but you can only leap up"'CHR$133;"
and down one block at a time."
 2Ø7Ø *FX15,Ø
 2Ø8Ø PRINT''CHR$129;" Do you want a de
mo ?"CHR$136;" (Y/N) ";
 2Ø9Ø IK$=GET$:IF IK$="Y" THEN demo%=TR
UE ELSE IF IK$="N" THEN demo%=FALSE ELS
E GOTO 2Ø9Ø
 21ØØ ENDPROC
 211Ø :
 212Ø DEF PROCcng
 213Ø SC%=SCT%:CLS:COLOUR2
 214Ø PRINTTAB(4,5)"OH DEAR !"TAB(4,7)"
YOU LOST"TAB(4,9)"A MAN"
 215Ø PROCsctab:SC%=SC%+FNsc(H%)
 216Ø ENDPROC
 217Ø :
 218Ø DEF PROCccg
 219Ø SCT%=SC%:CLS:COLOUR2
 22ØØ PRINTTAB(2,5)"NEXT -->"TAB(2,8)"C
AVERN NUMBER ";N%
 221Ø PROCsctab:SC%=SC%+FNsc(H%)
 222Ø ENDPROC
 223Ø :
 224Ø DEF PROCsctab
 225Ø VDU19,1,6,Ø,Ø,Ø
 226Ø COLOUR1
 227Ø PRINTTAB(4,14)"MEN.......";MEN%
 228Ø PRINTTAB(4,16)"SCORE.....";SC%
 229Ø PRINTTAB(4,18)"HISCORE...";HI%
 23ØØ COLOUR2:PRINTTAB(1,28)"RETURN TO
CONTINUE"
 231Ø PRINTTAB(1,28)"RETURN TO CONTINUE"
 232Ø PROCr:VDU19,1,1,Ø,Ø,Ø
 233Ø ENDPROC
 234Ø :
 235Ø DEF PROCover
 236Ø CLS:COLOUR2
 237Ø SC%=SC%+MEN%*FNsc(18)
 238Ø PRINTTAB(2,3)"CONGRATULATIONS"TAB
(2,6)"YOU HAVE ESCAPED"TAB(2,8)"FROM AL
L THE"TAB(2,1Ø)"THE CAVERNS"
 239Ø PROCsctab
 24ØØ ENDPROC
 241Ø :
 242Ø DEF PROClost
 243Ø CLS:COLOUR2:SC%=SCT%
 244Ø PRINTTAB(4,4)"GAME OVER"TAB(4,6)"
YOU RAN OUT"TAB(4,8)"OF MEN"
 245Ø PROCsctab
 246Ø ENDPROC
 247Ø :
 248Ø DEF PROCbonk
 249Ø CLS
 25ØØ FOR X%=15Ø TO 5ØØ STEP 5Ø:GCOLØ,1
:MOVE X%,X%:DRAW X%,-X%:DRAW -X%,-X%:DR
AW -X%,X%:DRAW X%,X%:X%=X%+5Ø:GCOLØ,2:M
OVE X%,X%:DRAW X%,-X%:DRAW -X%,-X%:DRAW
 -X%,X%:DRAW X%,X%:NEXT
 251Ø COLOUR3:PRINTTAB(6,15)"B O N K!"
 252Ø FOR I%=1 TO 8:VDU19,1,3,Ø,Ø,Ø,19,
2,1, Ø,Ø,Ø:PROCd(8):VDU19,1,1,Ø,Ø,Ø,19,
2,3,Ø ,Ø,Ø:PROCd(8):NEXT
 253Ø ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SINGLE KEY CHARACTER DEFINER - D.Pooley

This single key definition allows you to define a character and is easily stored in a function key ready for immediate use.

Z -left	? -down
X -right	Space -insert block
* -up	S -print out the values

Moving the cursor over a square will delete it.

```
*KEYØMO.7:X=Ø:Y=Ø:REP.:P.TAB(X,Y);:Q=GET:X=(X+(Q=9Ø)-(Q=88))MOD8:Y=(Y+(Q=58)-(Q=47))
MOD8:X=X-(X<Ø)*8:Y=Y-(Y<Ø)*8:?(H.+X+Y*4Ø)=Q=32:U.Q=83:F.Y=ØTO7:C=Ø:F.X=ØTO7:C=C+(2^(
7-X)AND?(H.+X+Y*4Ø)=255):N.:P.TAB(1Ø,Y);C:N.|M
```

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

WORDWISE GOES LIKE A BOMB

Dear Sir,

Thank you for the WORDWISE package you supplied. It works extremely well.

Would you please note that any packages sent to servicemen overseas must have some identification on the outside and preferably a customs declaration. The WORDWISE package had none of this, and was therefore treated as a suspect package, i.e. the bomb disposal squad were called in to deal with it. Fortunately they declared it safe. They could just as easily have destroyed it!

J.M.Nicholson

Reply: We are not a subversive organisation though producing BEEBUG can be occasionally mindblowing! More seriously, we have noted Sgt. Nicholson's point about orders sent to servicemen.

BEEB IMPROVES AT MATHS

Dear Sir,

Mr McMillan should not blame his machine for scoring 1Ø.5% in his maths test! (Postbag Vol.2 No. 5). You pointed out that the problem arises from storing all numbers in binary and I would just add that there is no "rounding" function on the BBC micro.

However, if the following program is run, the machine will score 1ØØ%:

```
1Ø FOR N=Ø TO 1 STEP Ø.ØØ1
2Ø PRINT INT((N*1ØØØ)+Ø.5)
3Ø NEXT N
```

Robin A.Richmond

Reply: I thought that getting a friend to help you in exams wasn't allowed! - Ed.

PROBLEMS IN USING WORDWISE

Dear Sir,

I have now discovered that my WORDWISE ROM is not faulty. However, the reason for the apparent fault may be of interest to others. I recently wrote a routine which involved setting the base for the Ctrl-function-key codes to 14Ø (*FX227,14Ø). Unfortunately, WORDWISE appears to use the cursor keys as function keys and expects the standard setting of 144. Thus changing the setting upsets WORDWISE (it really ought to reset the base level itself).

S.Gaynor

WHY BOTHER WITH HEX?

Dear Sir,

A short article (BEEBUG Vol.1 No.8) explained clearly what hexadecimal is, and pointed out that one byte can hold any number up to &FF (decimal 255). However, it continued by saying that a 'page' of memory is &1ØØ bytes. While convenient, surely this is purely a convention, and a page could just as well be &1ØØ bytes to base 7, or base 163?

So why bother with hex? I can see that in the days of true binary machine code programming, hexadecimal was a useful tool to translate the mind boggling Øs and 1s into something human. But what, if any, is the advantage of hex when using a high level language like Basic?

J.Davies

Reply: Nearly all modern computers are binary machines and this is fundamental to their whole organisation. One byte, as Dr.Davies says, can address &1ØØ (decimal 256) memory locations. To address even one more memory location would require two bytes, but this would be very inefficient as two bytes could address as many as &1ØØØØ (65536 decimal) memory locations. So, keeping everything in binary (or hexadecimal) is simply the most efficient way of organising a binary machine.

In answer to Dr.Davies' second point, the answer is almost the same. Whenever Basic is used in connection with something that is essentially binary, then hexadecimal can, in practice, be a more useful and convenient representation than decimal (examples are in logical operations and in user defined characters). Decimal can be and often is used instead. We do try and ensure that the two are not unnecessarily mixed, but this is not always feasible with contributed programs.

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

TOOLKIT AND DISC DOCTOR CLASH

Dear Sir,

I have just been scanning the catalogue of BEEBUG software which accompanied the December issue of the magazine. As I looked at your TOOLKIT ROM and Computer Concepts' DISC DOCTOR, I found that several of the functions had the same name (e.g. EDIT, MOVE).

As the paged ROM system operates by offering commands to each ROM in turn until they are recognised, this means that if I have TOOLKIT in a higher socket than DISC DOCTOR I cannot get at 4 out of 20 of DISC DOCTOR's commands: in crude terms, that's six quid down the river.

I believe that anybody marketing a ROM should think of a nice neat name for each service he offers - and then immediately reject it as likely to cause this sort of clash. Could you not prefix each obvious command name with, for example, 'BEEB'?

A.H.Harker

Reply: Whilst it is true that several features of our TOOLKIT have the same name as those in Disc Doctor, we have tried to overcome such clashes, which might occur with any ROM based software, by allowing the commands in TOOLKIT to be prefixed by 'B', very much the solution Mr Harker has suggested. This means that if you put DISC DOCTOR and TOOLKIT in your machine together, and type *EDIT <return>, you will be taken into the editing mode of Disc Doctor, whilst typing *BEDIT takes you into TOOLKIT's editing mode. This ensures that you will be able to access all of its commands. Disc Doctor is configured to take priority, so it will reply to commands such as EDIT and MOVE irrespective of the ROM socket it is placed in. If there is no conflict, TOOLKIT can be given the ordinary non-prefixed call.

GUARANTEES ON DISC DRIVES

Dear Sir,

Reviews of disc drives in BEEBUG (Vol.1 No.6, Vol.2 No.7) do not mention the significant difference in the length of the guarantee periods between the various suppliers. i understand the Acorn disc drives are only guaranteed for 6 months while Microware, for example, guarantee all their drives for 12 months.

A further possible disadvantage to Acorn and other makes of disc drice that do not use an independant power supply is that in the event of the power supply unit on the BBC micro failing, it might "blow" the disc drive as well. The total repair bill could be very hefty indeed.

Ralph Erskine

SWINGING ON A STAR

Dear Sir,

Your October reviewer, Ian Gilbert, sounded very regretful that his STAR DP840 has no descenders. I certainly found the 'g' and 'q' unacceptable so I Wrote a crude little procedure to intercept these two on their way to the printer. Thus the 'g' is replaced by 'q' and later when the paper has been advanced slightly, overwritten with a 'j', so now you matrix dot printer owners know! This technique can also be used for underlining etc.

This is very crude. Perhaps some other reader has found how to do a screen dump for just one letter or found a way of defining one's own characters.

PS. I have never used my TAB key. Can anyone save this key from a life of loneliness?

PPS. To try out your ESP, make your space invaders invisible. Then try for a high score!

Martin Campbell

Reply: We can't think of a better fix for the DP840, but the TAB key is intended for use in applications software. Wordwise uses it rather like a typewriter TAB. If you want to bring it to life, there is an FX call which will define it to be any desired character. Use *FX219,x where x is the ASCII value to be returned. *FX219,27 will make it operate as an Escape key, or *FX219,12 will make it clear the screen. The TAB key may also be used as a programmable function key. Use *FX219,128+n to make it operate as function key n, though it is restricted to the range of 0 to 15. Program it as usual with *KEYn ...

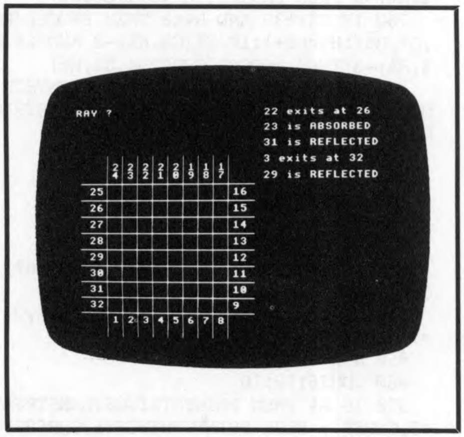
THE RAYBOX GAME (32K)

by Richard Stott

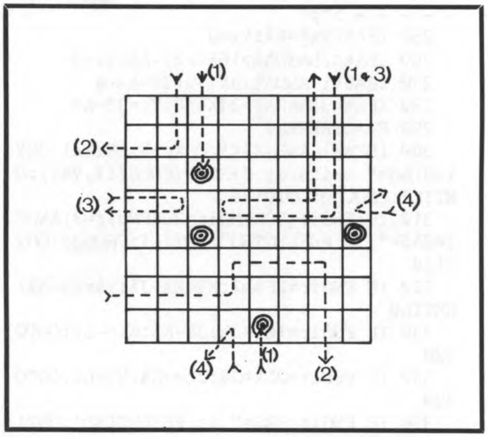
Test your powers of deduction with this very good implementation of the popular game 'Blackbox'.

Four pegs are hidden somewhere on an 8 by 8 grid, which is displayed on the screen. The object of the game is to discover where the pegs are located by sending in 'rays' from one of the 32 positions around the outside of the grid. The path of the ray is affected in different ways by different configurations of the pegs; it may be absorbed, reflected, or it may emerge at another point. By considering the possible ways in which the ray could have travelled from one point to another, you can find out where the computer has hidden the pegs.

You may try as long as you like to find the correct locations, but if you do want to give up and see where the pegs are, simply type in the number '99' in response to 'RAY?'. This will then reveal the locations of the pegs.



```
RAY ?
22 exits at 26
23 is ABSORBED
31 is REFLECTED
3 exits at 32
29 is REFLECTED
```



RULES

1) If a ray collides with a peg head on, it is said to be ABSORBED.
2) If a ray meets the corner of a peg, it will be DEFLECTED to one side.
3) If a ray meets two pegs at either side of its path, it will be REFLECTED.
4) If there is a peg immediately to the side of the cell at which a ray enters, the ray is also said to be REFLECTED. This is called 'reflection at an edge', but the computer sees all reflections as being the same.

```
10Ø REM PROGRAM  RAY-GAME
2Ø REM VERSION  BØ.3
3Ø REM AUTHOR   R.STOTT
4Ø REM BEEBUG   JAN/FEB 84
5Ø REM PROGRAM  SUBJECT TO COPYRIGHT
6Ø :
7Ø DIM S%(9,9)
8Ø ON ERROR GOTO 111Ø
9Ø MODE7:PROCvdu:PROCheader
```

```
 10Ø PROCclear
 11Ø MODE1:PROCvdu:PROCsetup
 12Ø PROCrand:PROCboard
 13Ø REPEAT
 14Ø A%=Ø:VDU26:PRINT''STRING$(24,CHR$
32):VDU11:PRINT"RAY ?";
 15Ø C%=GET:IF C%=13 AND A% THEN 22Ø
 16Ø IF C%=127 AND A% THEN VDU127:A%=A
%/1Ø:GOTO15Ø
```

```
  17Ø IF C%>135 AND C%<14Ø PROCcursor
  18Ø IF C%=32 OR C%=42 PROCspot((C%-42
)*-21.2+42):IF EOG THEN1Ø5Ø
  19Ø IF C%<48 OR C%>57 THEN15Ø
  2ØØ VDUC%:A%=A%*1Ø+C%-48
  21Ø GOTO15Ø
  22Ø UNTIL (A%>Ø AND A%<33) OR A%=99
  23Ø IF A%=99 PROCshow:VDU28,25,3Ø,39,
2,12:GOTO1Ø8Ø
  24Ø J%=FNJ(A%):K%=FNK(A%):PRINT" - -
- - - - - >";
  25Ø IFA%<9X%=A%:Y%=Ø
  26Ø IFA%<25ANDA%>16X%=25-A%:Y%=9
  27Ø IFA%<17ANDA%>8X%=9:Y%=A%-8
  28Ø IFA%<33ANDA%>24X%=Ø:Y%=33-A%
  29Ø F%=Ø:REPEAT
  3ØØ IFF%=1 AND(X%<1 ORX%>8 ORY%<1 ORY
%>8)A$=" exits at "+STR$(FNCB(X%,Y%)):U
NTIL1:PROCgo:GOTO13Ø
  31Ø IF(FNdir=1ORFNdir=2ORFNdir=3)ANDF
%=ØA$=" is REFLECTED":UNTIL1:PROCgo:GOT
O13Ø
  32Ø IF FNdir=ØF%=1:X%=X%+J%:Y%=Y%+K%:
UNTILØ
  33Ø IF FNdir=1L%=J%:J%=K%:K%=-L%:GOTO
32Ø
  34Ø IF FNdir=2L%=J%:J%=-K%:K%=L%:GOTO
32Ø
  35Ø IF FNdir=3A$=" is REFLECTED":UNTI
L1:PROCgo:GOTO13Ø
  36Ø A$=" is ABSORBED":UNTIL1:PROCgo:G
OTO13Ø
  37Ø :
  38Ø DEF PROCsetup
  39Ø *FX4,1
  4ØØ VDU19,2,11,Ø,Ø,Ø
  41Ø VDU23,254,Ø,Ø,Ø,24,24,Ø,Ø,Ø
  42Ø PROCPR(CHR$254,1,1):G%=1:H%=1
  43Ø ENDPROC
  44Ø DEFPROCPR(A$,X,Y)
  45Ø IF A$=CHR$254 THEN COLOUR2
  46Ø PRINT TAB(X*2+3,28-Y*2)A$:COLOUR3
  47Ø ENDPROC
  48Ø :
  49Ø DEFPROCboard
  5ØØ GCOLØ,1:FORX%=14ØTO652STEP64:MOVE
X%,36:DRAWX%,764:MOVE28,X%:DRAW764,X%:N
EXT
  51Ø GCOLØ,3:MOVE14Ø,14Ø:DRAW652,14Ø:D
RAW652,652:DRAW14Ø,652:DRAW14Ø,14Ø
  52Ø FORW%=1TO8:PROCPR(STR$W%,W%,Ø):NE
XT
  53Ø FORW%=9TO16:PROCPR(STR$W%,9,W%-8)
:NEXT
  54Ø FORW%=17TO24:PROCPR(LEFT$(STR$W%,
1),25-W%,9.5):PROCPR(RIGHT$(STR$W%,1),2
5-W%,9):NEXT
  55Ø FORW%=25TO32:PROCPR(STR$W%,-Ø.5,3
3-W%):NEXT
  56Ø ENDPROC
  57Ø :
  58Ø DEFPROCrand
  59Ø FOR T%=1 TO 4:REPEAT A%=RND(8):B%
=RND(8):UNTIL S%(A%,B%)=Ø:S%(A%,B%)=1:N
EXT
  6ØØ ENDPROC
  61Ø :
  62Ø DEF FNJ(Q%):IFQ%<9 OR(Q%>16ANDQ%<
25)THEN=ØELSEIFQ%<17THEN=-1 ELSE=1
  63Ø DEF FNK(Q%):IFQ%<9THEN=1ELSEIFQ%>
16ANDQ%<25THEN=-1ELSE=Ø
  64Ø DEF FNCB(M%,N%)
  65Ø IFN%=ØTHEN=M%
  66Ø IFN%=9THEN=25-M%
  67Ø IFM%=9THEN=N%+8
  68Ø IFM%=ØTHEN=33-N%
  69Ø DEF FNdir
  7ØØ =(S%(X%+J%,Y%+K%)AND1)*4+(S%(X%+K
%+J%,Y%-J%+K%)AND1)*2+(S%(X%-K%+J%,Y%+J
%+K%)AND1)
  71Ø DEF PROCcursor
  72Ø IF(S%(G%,H%) AND 2)=Ø THEN R$=CHR
$32 ELSE R$="*"
  73Ø IF C%=136 AND G%>1 THEN PROCPR(R$
,G%,H%):G%=G%-1:IF S%(G%,H%)-2 AND S%(G
%,H%)-3 THEN PROCPR(CHR$254,G%,H%)
  74Ø IF C%=137 AND G%<8 THEN PROCPR(R$
,G%,H%):G%=G%+1:IF S%(G%,H%)-2 AND S%(G
%,H%)-3 THEN PROCPR(CHR$254,G%,H%)
  75Ø IF C%=138 AND H%>1 THEN PROCPR(R$
,G%,H%):H%=H%-1:IF S%(G%,H%)-2 AND S%(G
%,H%)-3 THEN PROCPR(CHR$254,G%,H%)
  76Ø IF C%=139 AND H%<8 THEN PROCPR(R$
,G%,H%):H%=H%+1:IF S%(G%,H%)-2 AND S%(G
%,H%)-3 THEN PROCPR(CHR$254,G%,H%)
  77Ø COLOUR3:VDU26;1Ø;1Ø:IFA%PRINTSTRI
NG$(LENSTR$A%+5,CHR$9); ELSE PRINTSTRIN
G$(5,CHR$9);
  78Ø ENDPROC
  79Ø :
  8ØØ DEFPROCspot(W)
  81Ø EOG=1
  82Ø PROCPR(CHR$W,G%,H%)
  83Ø IF W=42 THEN S%(G%,H%)=S%(G%,H%)
OR 2 ELSE S%(G%,H%)=S%(G%,H%) AND 1
  84Ø FORT%=1TO8:FORU%=1TO8:IFS%(T%,U%)
=1 OR S%(T%,U%)=2 THEN EOG=Ø
  85Ø NEXT U%,T%
  86Ø VDU26;1Ø;1Ø
  87Ø IF A% THEN PRINTSTRING$(LENSTR$A%
+5,CHR$9); ELSE PRINTSTRING$(5,CHR$9);
  88Ø ENDPROC
  89Ø :
  9ØØ DEFPROCvdu
  91Ø VDU23,1,Ø;Ø;Ø;Ø;
  92Ø ENDPROC
  93Ø :
  94Ø DEFPROCheader
  95Ø FORT%=1TO2:PRINT TAB(12,T%);CHR$1
41;CHR$129;"RAYGAME":NEXT
```

```
  96Ø PRINT'" When the prompt 'RAY ?' a
 ppears on the"'"screen, type in the ent
 ry point as a"'"number (1-32) followed
 by RETURN."''" Using the cursor keys, m
 ove the"'"flashing dot about the grid,
 and"'"place or erase a marker by typing
  a '*'"
  97Ø PRINT"or a SPACE."''" These two k
 inds of input operate"'"simultaneously.
  The game will end when"'"you have only
  four markers on the grid"'"and all are
  in the correct places."'
  975 PRINT'" If you wish to see the pe
 g positions  at any time, then simply
 type '99' in   response to 'RAY?'."
  98Ø PRINT' CHR$145;CHR$157;SPC7;CHR$1
 36;CHR$131"Press SPACE to start";
  99Ø  REPEAT UNTIL GET=32
 1ØØØ ENDPROC
 1Ø1Ø :
 1Ø2Ø DEFPROCgo
 1Ø3Ø VDU7;28,25,3Ø,39,2:PRINTCHR$11;CH
R$11;STR$A%+A$:GU%=GU%+1:L%=INKEY(1ØØ)
 1Ø4Ø ENDPROC
 1Ø5Ø VDU28,25,3Ø,39,2:CLS:COLOUR2:*FX9
,25
 1Ø6Ø *FX1Ø,25
 1Ø7Ø PRINT''"  CORRECT !"':COLOUR1:PRI
NT" You used ";GU%'"  rays to"'"  fin
ish.":FORT%=1TO9:READV,P,D:SOUND1,V,P,D
:NEXT
 1Ø8Ø COLOUR3:PRINT''''"Another game?"'
"Y/N - ";:REPEATG=GET:UNTILG=89ORG=78:V
DUG:IFG=78PRINT''"Ok."ELSEI=INKEY5Ø:RUN
 1Ø9Ø DATA2Ø,96,4,2Ø,76,2,32,76,1,2Ø,76
,1,2Ø,84,4,2Ø,76,4,32,76,4,2Ø,92,4,2Ø,9
6,4
 11ØØ :
 111Ø ON ERROR OFF:MODE7:*FX4
 112Ø IF ERR<>17 REPORT:PRINT" at line
";ERL
 113Ø END
 114Ø :
 115Ø DEFPROCshow
 116Ø GCOL3,2
 117Ø FORT%=1TO8:FORU%=1TO8:PROCPR(" ",
T%,U%):IFS%(T%,U%)=1 OR S%(T%,U%)=3 THE
N PROCPR("*",T%,U%)
 118Ø NEXT:NEXT
 119Ø ENDPROC
 12ØØ :
 121Ø DEFPROCclear
 122Ø FORc1%=1TO8:FORc12%=1TO8
 123Ø S%(c1%,c12%)=Ø
 124Ø NEXT,
 125Ø GU%=Ø
 126Ø ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SHORTER VDU CALLS - David Fell

The following two commands are completely equivalent, although the second version saves 19 bytes:

```
VDU 23,24Ø,255,255,255,255,255,255,255,255
VDU 23,24Ø,-1;-1;-1;-1;
```

Similarly 254 could be represented as -2, 253 as -3 and so on.

FORMAT ERRORS

If you have the Watford Electronics DFS, and Disc Doctor in your machine at the same time, and you use Disc Doctor to format your discs in drives 1 or 3, then you will find it formating drive Ø. In addition, to format drive 2, you must select that drive first. Disc Doctor formats discs correctly when used with the Acorn DFS.

SAFER ESCAPE ACTION - R.Duebel

Safer ways of obtaining an Escape condition through the use of *FX22Ø are:

```
*FX22Ø,Ø   makes Ctrl-@ produce Escape
*FX22Ø,128 makes 'fØ' produce Escape
*FX22Ø,144 makes Shift-fØ produce Escape
*FX22Ø,16Ø makes Ctrl-fØ produce Escape
*FX22Ø,176 makes Shift-Ctrl-fØ produce Escape.
```

The last command is an extremely useful and safe 'three-key' Escape action.

MEMORY SHORTAGE ON DISC MACHINES

Placing both Acorn's and Watford's DFS in your machine at the same time, will result in PAGE being set to &1BØØ when you switch on, since both grab their own private workspace. This will limit even further the size of program that it is possible to run.

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues are kept in print (from April 1982). Send 90p per issue PLUS an A5 SAE to the subscriptions address. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

NOTE OUR NEW SUBSCRIPTIONS ADDRESS

Subscriptions & Software Address
BEEBUG
PO BOX 109
High Wycombe
Bucks
HP11 2TD

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:
£5.40 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
European Membership £16 for 1 year.
Elsewhere (Postal zones)
Zone A £19, Zone B £21, Zone C £23

SOFTWARE AND ROM OFFER (Members only)

These are available from the subscription address, which is our NEW software address also. (Note that this does not apply to Wordwise or Beebcalc - in this instance please see magazine for details).

IDEAS, HINTS & TIPS, PROGRAMS, AND LONGER ARTICLES

Substantial articles are particularly welcome and we will pay around £25 per page for these, but in this case please give us warning of anything that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "Minitext Editor" or other means. If you use cassette, please include a backup copy at 3ØØ baud.

Editorial Address
BEEBUG
PO Box 5Ø
St Albans
Herts
AL1 2AR

We will also pay £1Ø for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Edited by Mike Williams. Technical Editor: Philip Le Grand.
Production Editor: Phyllida Vanstone. Technical Assistant: Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft, John Yale, Michael Beesley, David Fell and Tim Powys-Lybbe for assistance with this issue.

BEEBUG NEW ROM OFFER

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the Series One Machine Operating System in ROM at the price of £5.85 including VAT and post and packing.

The ROM will be supplied with fitting instructions to enable members to install it in their machine.

If the computer does not subsequently operate correctly, members may take their machines to an Acorn dealer for the upgrade to be tested, which will be done at a charge of £6.00 plus VAT. This charge will be waived if the ROM is found to have been defective. If the computer has been damaged during the installation process, the dealer will make a repair charge.

Please note that we cannot accept EPROM-based operating systems (0.1 or 1.0) in lieu of payment. This can only be performed by Acorn dealers or by Acorn's service centre at Feltham, and applies only to users of the 0.1 O.S.

ADDRESS FOR 1.2 O.S. IF ORDERED ON ITS OWN:- ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD. PLEASE ALLOW 28 DAYS FOR DELIVERY.

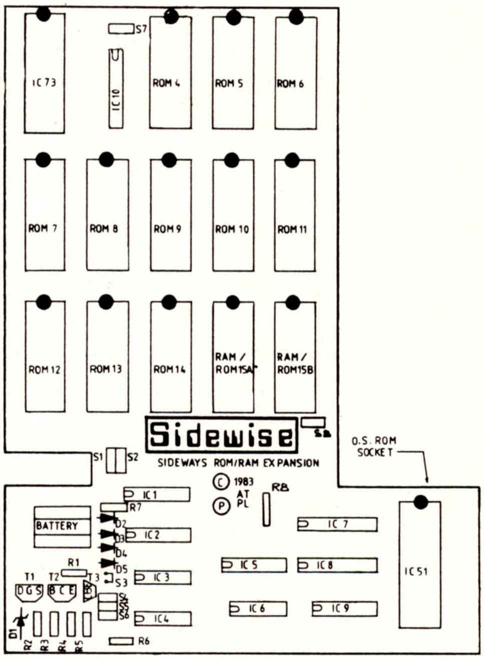
BEEBUGSOFT

ATPL'S SIDEWAYS ROM EXPANSION BOARD

SPECIAL PRICE TO MEMBERS £39.00 inc.

Save £5.70 on normal price of £44.70

* Simply plugs into the BBC Micro
* No soldering necessary
* Increases the sideways ROM capacity to 16
* Fully buffered - allows all sockets to be used
* Complete with full and detailed instruction booklet.
* Accepts 16K RAM in special sockets
* Battery back up facility for RAM
* As used at BEEBUG
* Reviewed in BEEBUG vol.2 number 6



HOW TO ORDER

Please send your order with a cheque / postal order made payable to BEEBUG, and enclose your membership number. We are unable to supply the board to overseas members.

The address for SIDEWAYS is:
BEEBUGSOFT, PO Box 109, High Wycombe, Bucks. HP11 2TD.

MAGAZINE CASSETTE OFFER

To save wear and tear on fingers and brain, we will be offering each month a cassette of the programs featured in the latest edition of BEEBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles. Ordering information, and details of currently available cassettes are given below.

Previous cassettes: Vol.1 No.1Ø, Vol.2 No.1, Vol.2 No.2, Vol.2 No.3, Vol.2 No.4, Vol.2 No.5, Vol.2 No.6, Vol.2 No.7.

This month's cassette (Vol.2 No.8) includes: Block Blitz game, Machine Code Graphics example programs,

Teletext procedures and program, Disassembler, Raybox game, Program for Large Digital Display in Mode 7, Procedure to protect Basic programs, Dancing Line program, plus Program Compactor (from BEEBUG Vol.1 No.9). All magazine cassettes cost £3.ØØ each. For ordering information see BEEBUGSOFT advertisement at the back of this month's magazine supplement.

MAGAZINE CASSETTE SUBSCRIPTION

We are able to offer members subscription to our magazine cassettes. Subscriptions will be for a period of one year and are for ten consecutive issues of the cassette. If required, subsriptions may be backdated as far as Vol.1 No.1Ø, which was the first issue available on cassette. This offer is available to members only, so when applying for subscription please write to the address below, quoting your membership number and the issue from which you would like your subscription to start.

CASSETTE SUBSCRIPTION ADDRESS:

Please send a sterling cheque with order, together with your membership number and the date from which the subscription is to run, to:
PO Box 1Ø9, High Wycombe, Bucks, HP11 2TD.

CASSETTE SUBSCRIPTION PRICE:
UK £33 inc VAT and p&p
OVERSEAS (inc Eire) £39 inc p&p
(no VAT payable).

BEEBUG BINDER OFFER

BEEBUG MAGAZINE BINDER OFFER

A hard-backed binder for the BEEBUG magazine is now available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to use the whole of the first volume of the magazine as a single reference book. Individual issues may be easily added or removed. The binders will also conveniently hold less than 1Ø issues, so that you can use it while you build up Volume 2 also.

BINDER PRICE
U.K. £3.9Ø inc p&p, and VAT.
Europe £4.9Ø inc p&p (VAT not charged)
Elsewhere £5.9Ø inc p&p
(VAT not charged)

Make cheques payable to BEEBUG.
Send to Binder Offer, BEEBUG, PO Box 1Ø9, High Wycombe, Bucks, HP11 2TD.
Please allow 28 days for delivery on U.K. orders.

Printed in England by Staples Printers St Albans Limited at The Priory Press.   ISSN 0263-7561