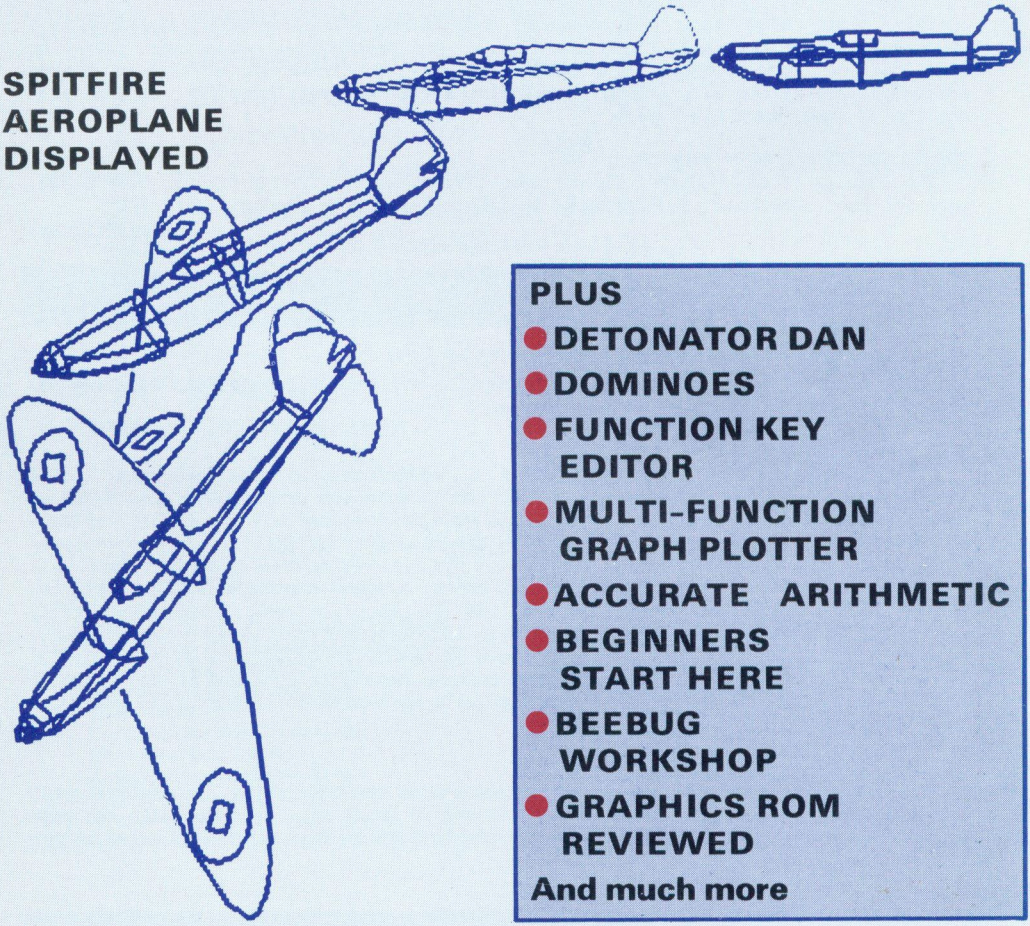
£1.00

BEEBUG

FOR THE BBC MICRO

Vol 3 No 1 MAY 1984



SPITFIRE AEROPLANE DISPLAYED

PLUS

- **DETONATOR DAN**
- **DOMINOES**
- **FUNCTION KEY EDITOR**
- **MULTI-FUNCTION GRAPH PLOTTER**
- **ACCURATE ARITHMETIC**
- **BEGINNERS START HERE**
- **BEEBUG WORKSHOP**
- **GRAPHICS ROM REVIEWED**

And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 25,000

EDITORIAL

SECOND ANNIVERSARY ISSUE

This issue of BEEBUG marks the second anniversary of the magazine. To mark this occasion, we are including with this issue a voucher worth £1 against any order to BEEBUG or BEEBUGSOFT, except for magazine subscriptions and back copies. The magazine itself contains a number of first-rate articles and programs that we have been saving just for this issue.

The Spitfire aeroplane display is particularly interesting in that its author, Mr I.C.Grant, worked for over 20 years in the design office of Supermarine Swift who built the original plane. It also shows what can be achieved with BEEBUG, your BBC micro and, of course, a lot of hard work. The magazine contains a listing of the data to be used with the 3D Rotation program by James Hastings that was first published in BEEBUG Vol.1 No.10. The complete program together with the data is included on this month's magazine cassette, and as a special bonus, we are hoping to arrange for the transmission of this program as part of the BBC's Telesoftware service for two weeks in early May.

This month also sees the start of two new features. The BEEBUG Workshop is a practical series for all those interested in programming, and any ideas or contributions from readers will be very welcome. For those who are new to micros and computing, we are also starting a special series of articles under the general heading of 'Beginners Start Here'.

FOURTH SOFTWARE COMPETITION

The results of the Fourth Software Competition, with prizes totalling over £1000, have now been announced. Full details appear in this month's supplement.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAI

HINT WINNERS

This month's hint winners are M.Robson who wins the £10 prize and P.Davies who wins the £5 prize. Keep sending in those good ideas.

MAGAZINE CASSETTE

This month's magazine cassette contains, in addition to all the programs listed in the magazine, the complete version of the 3D Rotation program by James Hastings first published in BEEBUG Vol.1 No.10. This version includes all published updates and other minor improvements together with the data for the Spitfire.

We have also included as a special bonus, a super-smooth Mode 7 action game, called SHAPES, by C.J.Fitch. The program is in machine code to provide truly impressive screen movement, resulting in an addictive and challenging game. Several of this month's programs require moving down in memory in order to run correctly on a normal disc machine. To assist with this we have also included on the magazine cassette a general 'Move-down' routine that will work with both Basic and machine code programs.

MAGAZINE DISC

In response to a growing number of requests from BEEBUG members, we shall shortly be launching a magazine disc (both 40 and 80 track) to complement the magazine cassette already available. More details on this next month, including arrangements for existing subscribers.

BEEBUG MAGAZINE

GENERAL CONTENTS

2 Editorial
4 News
5 Spitfire Aeroplane Displayed
9 Beginners Start Here
An Introduction to Using Procedures
11 Screen to Printer Dumps
The Latest ROMs Reviewed
13 Multi-Function Graph Plotter
17 Computer Concepts' Graphics ROM Reviewed
18 BEEBUG Workshop
An Elegant Data Entry Routine
19 Errata to BEEBUG Vol 2 No 10
20 Function Key Editor
24 Accurate Arithmetic
26 Machine Code Graphics (Part 4)
30 Testing Out Your Micro (Part 3)
Random Access Memory
31 Points Arising
32 VASM – A 6502 Disc-based Assembler Reviewed
34 Dominoes
38 Detonator Dan

HINTS, TIPS & INFO

Page Contents

8 Obtaining Negative Numbers in Hex
8 Preventing the Screen from Scrolling
8 Switching Basics
16 Defining your own Function Keys in VIEW
16 Resetting the Computer during a Program
23 Swapping Filing Systems Utilities
29 Relocating Character Definitions
33 WORDWISE Hints
Extra Function Keys in Wordwise
Inaccurate Word Count
Automatic Deletion of Markers
41 Teletext Downloader Clash
41 Accurately Filling Rectangular Areas
41 Super-condensed Characters on an Epson Printer

PROGRAMS

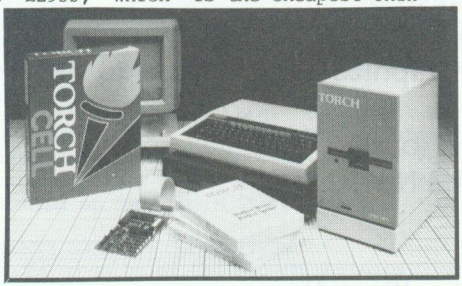
Page Contents

5 Spitfire Aeroplane Data
9 Procedure Examples
14 Multi-Function Graph Plotter
18 Workshop Data Entry Procedure
24 Accurate Arithmetic – Powers
24 Accurate Arithmetic – Division
26 Machine Code Graphics
Examples 9 and 10
30 Random Access Memory Tester
34 Dominoes Game
38 Detonator Dan Game

NEWS NEWS NEWS

20 MEGABYTE DISC SYSTEM FROM TORCH

Torch computers, who were the first to make a Z80 second processor and dual disc pack for the Beeb, have now expanded their range with a 16 bit second processor, called the Unicorn. The Unicorn is the top of the range with an 8MHz 68000 processor, 256K RAM, System III Unix, 20Mbyte hard disc, 400K single floppy disc drive and a Z80 processor to run existing Torch software. The price of this little lot is £2900, which is the cheapest Unix



system in the world! With the full system, a whole range of software is supplied including word processor, spelling checker, spreadsheet and a database. If you require any further details of Torch's range, contact them at Abberley House, Great Shelford, Cambridge, CB2 5LQ.

6502 SECOND PROCESSOR FROM ACORN

The most important extension, so far, for the BBC micro was officially released by Acorn on the 14th March. It is the long awaited 6502 Second Processor running at 3MHz with 44K of memory free for Basic programs, or 60K for machine code. The complete unit is supplied with two ROMs, one containing the new 'Hi-Basic' which is automatically copied across the Tube when the Beeb is powered up, and the second the new Disc and Network filing systems. Both ROMs are fitted into the BBC micro's sideways ROM sockets. It is claimed that this second processor will increase the Beeb's speed by as much as 50%, but this will depend on the task being performed. The complete unit costs £199 (inc.). Delivery of the Second Processor was confidently quoted as two months for new orders.

CAD SYSTEM FROM ACORN

Acorn are also selling their comprehensive CAD package called Bitstik for the BBC micro. The Bitstik is a high quality triple axis joystick, which requires a 6502 Second Processor system with dual 80 track disc drives to operate. The software for Bitstik consists of a ROM and a disc of utilities. Some of the facilities of the system include drawing in four colours, with up to twelve shades, freehand or computer drawn lines (circles, arcs, or straight lines), zoom up to 2^120 times larger with the ability to add details at any level, pan in all directions to display details 'off' the screen and manipulate any point or character on the screen (e.g. enlarge, shrink, stretch, reverse). The drawing, once created, can be saved onto disc, with each disc being able to store up to 48 drawings. An index for the pictures is maintained by the system, consisting of miniature versions of the drawings for rapid identification.

This system is the best CAD package available for any micro at the moment, and at the price, is much cheaper than many other professional systems. The price for the Bitstik and associated software is £375 (inc.). There should be no delay for this particular product if you order one now.

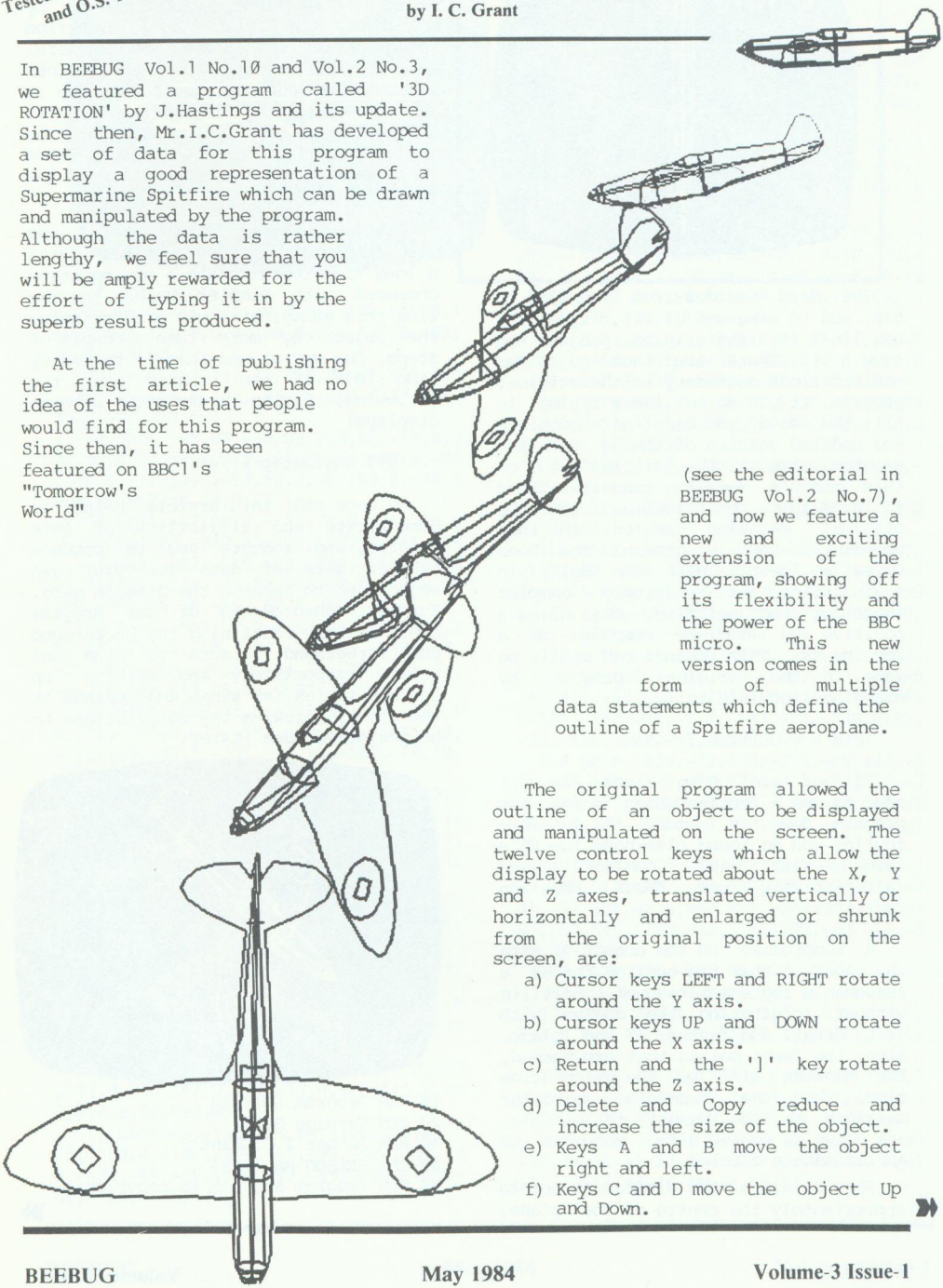


For further details on any of Acorn's products, contact your local dealer or Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN. All of Acorn's products can be ordered from Vector Marketing, Denington Estate, Wellingborough, Northamptonshire, NN8 2RL. Tel: 0933 79300

Tested on Basic I & II and O.S. 1.2

SPITFIRE AEROPLANE DISPLAYED

by I. C. Grant

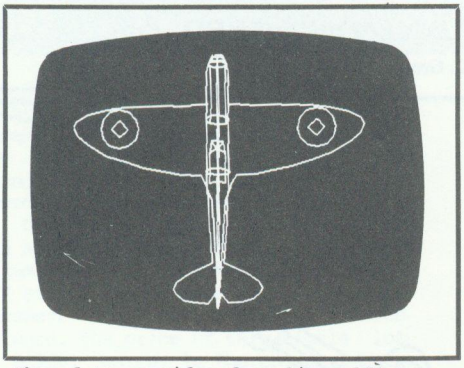


In BEEBUG Vol.1 No.10 and Vol.2 No.3, we featured a program called '3D ROTATION' by J.Hastings and its update. Since then, Mr.I.C.Grant has developed a set of data for this program to display a good representation of a Supermarine Spitfire which can be drawn and manipulated by the program. Although the data is rather lengthy, we feel sure that you will be amply rewarded for the effort of typing it in by the superb results produced.

At the time of publishing the first article, we had no idea of the uses that people would find for this program. Since then, it has been featured on BBC1's "Tomorrow's World" (see the editorial in BEEBUG Vol.2 No.7), and now we feature a new and exciting extension of the program, showing off its flexibility and the power of the BBC micro. This new version comes in the form of multiple data statements which define the outline of a Spitfire aeroplane.

The original program allowed the outline of an object to be displayed and manipulated on the screen. The twelve control keys which allow the display to be rotated about the X, Y and Z axes, translated vertically or horizontally and enlarged or shrunk from the original position on the screen, are:

a) Cursor keys LEFT and RIGHT rotate around the Y axis.
b) Cursor keys UP and DOWN rotate around the X axis.
c) Return and the ']' key rotate around the Z axis.
d) Delete and Copy reduce and increase the size of the object.
e) Keys A and B move the object right and left.
f) Keys C and D move the object Up and Down.



The data resides from lines 100 to 570, and to make use of it, type the new lines into the original program and save it. There are thus no other modifications necessary to the original program. If you do not fancy typing in all the data, you can find a complete and updated version of the 3D Rotation program, plus all the Spitfire data, on this month's magazine cassette. There is a slight modification to the original magazine version, in that whenever a key is pressed the Beeb emits a 'beep'. With the delay in manipulating the relatively complex image of the spitfire, this gives a positive and immediate response to a key press. This change can easily be made to the original program by changing line 1810 to:

```
1810 key=GET:VDU 7
```

If you are using discs, you will need to use a routine which moves the program down in memory. A suitable routine can be found elsewhere in this issue with the DOMINOES game (alternatively use *MOVE E00 on Toolkit!).

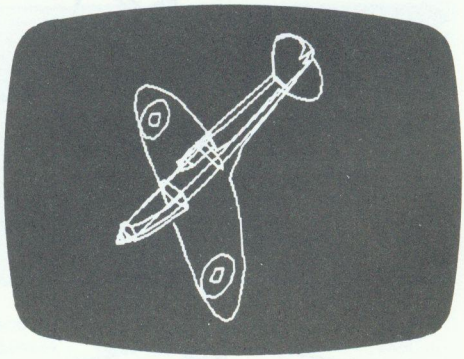
A compromise on the amount of data was made, so that the outline formed a reasonable representation of a Spitfire without cluttering the design with every detail and feature of the plane. Also, the fewer points that are needed, the shorter will be the calculation time. Thus the roundels (circular markings on the wings), for example, are drawn as twelve sided polygons to approximate a circle.

The origin of the diagram is set to approximately the centre of the plane, just below the cockpit canopy, but could equally have been set to any point. The data was obtained by dividing the plan and elevation drawings of the plane into several frames, and marking all the points where the outline deviated from a straight line (see diagram 1). The 262 co-ordinates thus calculated form the start and end positions for the lines which make up the representation of the plane's outline.

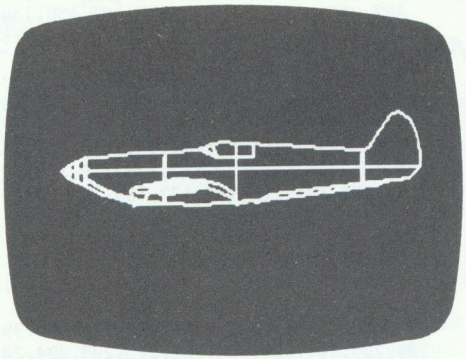
With the extra data entered, you will find that the program takes quite a long time to calculate a new position compared with simpler shapes. You may find this annoying if you are rotating the object by more than a couple of steps. One way around this is to modify line 1090 of the program to the following, giving larger steps between displays:

```
1090 anglestep=PI/4
```

We hope that this article helps to demonstrate the flexibility of this program, and induces you to produce further sets of data for your own objects or to enhance the diagram here. Further enhancements of the program could include redefining the background and foreground colours to blue and yellow respectively and setting up lookup tables for sines and cosines at the start, allowing the calculations to be performed much faster.



```
10 REM Program SPITFIR
20 REM Version B1.0
30 REM Author I.C.Grant
40 REM BEEBUG May 1984
50 REM Program subject to copyright
```



```
  70 REM Points data
 100 DATA 245,675,50,0,665,120,0,638,1
65,0,598,188,0,570,180,0,525,125,0,488,
70,0,468,60,0
 110 DATA 468,35,-15,468,0,-25,468,-35
,-15,468,-70,0,468,-35,15,468,0,25,468,
35,15,333,70,0
 120 DATA 168,80,0,0,89,0,0,86,-15,0,3
0,-50,0,0,-60,0,-30,-62,0,-65,-60,0,-90
,-50,0,-115,0,0,-90,50,0,-65,60,0,-30,6
2,0,0,60,0,30,50,0,86,15
 130 DATA -47,95,0,-132,70,0,-262,60,0
,-262,56,-28,-262,35,-46,-262,0,-61,-26
2,-43,-63,-262,-115,-46,-262,-115,0,-26
2,-115,46,-262,-43,63,-262,0,61,-262,35
,46,-262,56,28,-362,60,0,-502,50,0
 140 DATA -552,50,0,-552,43,-25,-552,2
5,-43,-552,0,-50,-552,-25,-43,-552,-43,
-25,-552,-50,0,-552,-43,25,-552,-25,43,
-552,0,50,-552,25,43,-552,43,25
 150 DATA -582,35,0,-582,30,-18,-582,1
8,-30,-582,0,-35,-582,-18,-30,-582,-30,
-18,-582,-35,0,-582,-30,18,-582,-18,30,
-582,0,35,-582,18,30,-582,30,18
 160 DATA -612,22,0,-612,19,-11,-612,1
1,-19,-612,0,-22,-612,-11,-19,-612,-19,
-11,-612,-22,0,-612,-19,11,-612,-11,19,
-612,0,22,-612,11,19,-612,19,11,-637,0,
0
 170 DATA -502,-75,0,-362,-115,0,-132,
-120,0,-47,-115,0,168,-100,0,333,-80,0,
503,-65,0,518,-60,0
 180 DATA 538, 50,0,588,-45,0,638,-34,
0,673,0,0
 190 DATA 600,0,-15,650,0,-50,645,0,-1
20,630,0,-180,597,0,-220,570,0,-230,539
,0,-210,498,0,-150,445,0,-30
 200 DATA 445,0,30,498,0,150,539,0,210
,570,0,230,597,0,220,630,0,180,645,0,12
0,650,0,50,600,0,15
 210 DATA 85,-110,-60,48,-108,-72,0,-1
15,-100,0,-104,-200,-15,-90,-300,-32,-7
8,-400,-59,-65,-500,-90,-55,-600,-145,-
45,-700,-186,-40,-755
 220 DATA -230,-35,-780,-265,-38,-765,
-300,-45,-700,-325,-55,-600,-340,-65,-5
00,-347,-78,-400,-352,-90,-300,-356,-10
4,-200,-365,-115,-100,-365,-115,-60
 230 DATA -365,-115,60,-365,-115,100,-
356,-104,200,-352,-90,300,-347,-78,400,
-340,-65,500,-325,-55,600,-300,-45,700,
-265,-38,765,-230,-35,780
 240 DATA -186,-40,755,-145,-45,700,-9
0,-55,600,-59,-65,500,-32,-78,400,-15,-
90,300,0,-104,200,0,-115,100,48,-108,72
,85,-110,60
 250 DATA 65,40,40,65,82,15,65,85,0,65
,82,-15,65,40,-40,0,40,-40,-70,40,-40,-
132,70,-30,-132,70,30,-70,40,40,0,40,40
 260 DATA -80,65,35,-90,89,0,-80,65,-3
5,-112,76,-18,-112,76,18
 270 DATA 445,0,-30,245,0,-45,135,0,-5
0,85,0,-60,-365,0,-60,-502,0,-60,-552,0
,-50,-582,0,-35,-612,0,-22,-637,0,0
 280 DATA-612,0,22,-582,0,35,-552,0,50
,-502,0,60,-365,0,60,85,0,60,135,0,50,2
45,0,45,445,0,30
 290 DATA -362,-100,-40,-440,-80,-40,-
502,-55,-40
 300 DATA -502,-55,40,-440,-80,40,-362
,-100,40
 310 DATA -225,-45,-630,-275,-49,-617,
-312,-50,-580,-325,-57,-530,-312,-65,-4
80,-275,-69,-443,-225,-70,-430,-175,-69
,-443,-138,-65,-480,-125,-57,-530,-138,
-50,-580,-175,-49,-617
 320 DATA -225,-50,-572,-241,-52,-558,
-253,-53,-546,-267,-56,-530,-253,-60,-5
14,-241,-62,-502,-225,-65,-488,-209,-62
,-502,-197,-60,-514,-183,-56,-530,-197,
-53,-546,-209,-52,-558
 330 DATA -225,-70,430,-275,-69,443,-3
12,-65,480,-325,-57,530,-312,-50,580,-2
75,-49,617,-225,-45,630,-175,-49,617,-1
38,-50,580,-125,-57,530,-138,-65,480,-1
75,-69,443
 340 DATA -225,-65,488,-241,-62,502,-2
53,-60,514,-267,-56,530,-253,-53,546,-2
41,-52,558,-225,-50,572,-209,-52,558,-1
97,-53,546,-183,-56,530,-197,-60,514,-2
09,-62,502
 350 DATA 130,-90,-35,130,-90,35
 360 REM Lines data
 370 DATA 262,1,2,2,3,3,4,4,5,5,6,6,7,
7,8,8,9,9,10,10,11,11,12,12,13,13,14,14
,15,15,8,8
 380 DATA 16,16,17,17,18,18,19,19,20,2
0,21,21,22,22,23,23,24,24,25,25,26,26,2
7,27,28,28,29,29,30,30,31,31,18,18
 390 DATA 32,32,167,167,33,33,34,34,35
,35,36,36,37,37,38,38,39,39,40,40,41,41
,42,42,43,43,44,44,45,45,34,34
```

```
  400 DATA 46,46,47,47,48,48,49,49,50,5
0,51,51,52,52,53,53,54,54,55,55,56,56,5
7,57,58,58,59,59,48,48
  410 DATA 60,60,61,61,62,62,63,63,64,6
4,65,65,66,66,67,67,68,68,69,69,70,70,7
1,71,60,60
  420 DATA 72,72,73,73,74,74,75,75,76,7
6,77,77,78,78,79,79,80,80,81,81,82,82,8
3,83,72,72,84,84
  430 DATA 78,78,66,66,54,54,85,85,86,8
6,40,40,87,87,88,88,25,25,89,89,90,90,9
1,91,92,92,93,93,94,94,95,95,96,96,1
  440 DATA 96,97,97,98,98,99,99,100,100
,101,101,102,102,103,103,104,104,105
  450 DATA 106,107,107,108,108,109,109,
110,110,111,111,112,112,113,113,114,114
,96
  460 DATA 115,116,116,117,117,118,118,
119,119,120,120,121,121,122,122,123,123
,124,124,125,125,126,126,127,127,128,12
8,129,129,130,130,131,131,132,132,133,1
33,134
  470 DATA135,136,136,137,137,138,138,1
39,139,140,140,141,141,142,142,143,143,
144,144,145,145,146,146,147,147,148,148
,149,149,150,150,151,151,152,152,153,15
3,154
  480 DATA 155,156,156,157,157,158,158,
159,159,160,160,161,161,162,162,163,163
,164,164,165,165,155
  490 DATA 164,166,166,167,167,168,168,
161,162,163,163,170,170,167,167,169,169
,162
  500 DATA 171,172,172,173,173,174,174,
175,175,176,176,177,177,178,178,179,179
,180,180,181,181,182,182,183,183,184,18
4,185,185,186,186,187,187,188,188,189
  510 DATA 39,190,190,191,191,192,192,5
2
  520 DATA 56,193,193,194,194,195,195,4
1
  530 DATA 196,197,197,198,198,199,199,
200,200,201,201,202,202,203,203,204,204
,205,205,206,206,207,207,196
  540 DATA 208,209,209,210,210,211,211,
212,212,213,213,214,214,215,215,216,216
,217,217,218,218,219,219,208
  550 DATA 220,221,221,222,222,223,223,
224,224,225,225,226,226,227,227,228,228
,229,229,230,230,231,231,220
  560 DATA 232,233,233,234,234,235,235,
236,236,237,237,238,238,239,239,240,240
,241,241,242,242,243,243,232
  570 DATA 95,244,244,115,154,245,245,9
5
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

OBTAINING NEGATIVE NUMBERS IN HEX - R.Claridge

Sometimes you may need to display a negative hex value. One of the easiest ways of doing this is to put the value into a spare location in memory using '!', and recover the required hex number using a '?' to access the first location where the value is stored. (Numbers in BBC Basic are stored in successive locations with the least significant byte in the first address.) For example:

```
10 location=&70
20 variable=-10
30 !(location)=variable
40 PRINT " Normal form            = "; variable
50 PRINT " Least significant byte = "; ~?location
60 END
```

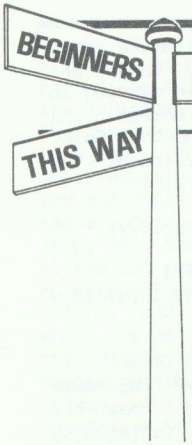
The value -10 as a four byte integer is FFFFFFF6, but the single byte returned for -10 is F6

PREVENTING THE SCREEN FROM SCROLLING - P.Davies

The screen will automatically scroll if a character is printed in the bottom right corner of the screen. This will have the effect of obscuring any text on the top line. An easy way to prevent this is to type in ?&D0=2. This sets the second bit of the location &D0, which controls the scrolling of the screen. If this bit is set, then the scrolling facility is inhibited. After printing a character in this position, the location must be returned to its former state by entering ?&D0=0.

SWITCHING BASICS - S.Williams

If you have both Basic I and Basic II in your machine, because you want to know if a program will work on both versions of Basic, you can use *FX142,n to select between them. The version of Basic being used remains active after pressing Break (but not necessarily after a Control-Break). The value of n to use in the FX command is the socket number of the required Basic.



PROCEDURES BEGINNERS START HERE

The BBC micro continues to remain well placed in the tables of top selling micros. With so many new users continuing to join BEEBUG we have decided to promote a regular series of articles that will look particularly at the needs of the new user, and maybe even old hands will learn a new trick or two along the way.

Of course, early issues of BEEBUG, published when the BBC micro was still in its infancy, contained many articles and programs for new users, as we were all beginners then. Now that is no longer the case, and although many of the early issues of the magazine remain a veritable fount of useful information, there is still a need for articles that will take the lid off some of the mysteries of the Beeb and explain all in plain and simple terms.

Peter Lewis starts our series with an introduction to the world of procedures for those still grappling with the problem of writing their own programs.

AN INTRODUCTION TO USING PROCEDURES

by Peter Lewis

Most people who buy themselves a micro to use at home, sooner or later start wanting to write their own computer programs. Learning a computer language like Basic is rather like learning any foreign language - you obviously need to learn some vocabulary, and you also need to know how to join the words together to say something meaningful.

Basic comes in many different varieties (probably at least 57) but BBC Basic is generally thought to be one of the best, certainly on a micro, and one of the reasons for this is its use of procedures. Many books, and indeed the User Guide itself, still tend to leave procedures till quite late on, giving the impression that this is a feature of the language which is only for the expert. This is not so. Even if you are just starting to program, you will find that using procedures will not only allow you to produce better programs, but will actually make the task of writing a program much easier as well.

Let's look first at the basic principles of procedures. Put at its simplest, a procedure is just a group of instructions, performing some particular task, to which we give a name. Whenever we want our program to carry out that task, we just have to refer to it by the name we have given.

For example, suppose we want a procedure to display a red filled square of side 400 centred on the screen. The procedure could be defined as follows:

```
1000 DEF PROCredsquare
1010 GCOL 0,1
1020 MOVE 440,312:MOVE 440,712
1030 PLOT 85,840,312:PLOT 85,840,712
1040 ENDPROC
```

Line 1000 defines the procedure with 'DEF PROC' followed by the name we choose to give the procedure, in this case 'redsquare'. All procedures are defined using the key words 'DEF PROC' (see page 102 of the User Guide). The following lines contain the instructions that make up this procedure, in this case the GCOL command to select the colour red for drawing, and then MOVE and PLOT instructions to display two triangles that together form a square (see the User Guide pages 56 and 162 for more information on these two instructions). Every procedure must be terminated, as in this case, by the instruction 'ENDPROC'.

Having 'defined' the procedure, as the above process is called, it is now time to use the procedure in a program. We could write:

```
100 MODE 2
110 PROCredsquare
120 END
```

The process of 'calling' the procedure, as in line 110 above, involves writing the keyword 'PROC' followed by the name of the procedure we want. If you enter the program and procedure definition into your micro, and run it, you should see a red square displayed in the centre of the screen.

Procedure definitions are usually numbered so that they follow the main program. When a procedure call is first encountered the computer searches through the program for that procedure, which is then carried out. The computer remembers where that procedure is located so that if it is called again, the computer can go directly to it.

Now although this is quite useful in itself there are some obvious limitations. The procedure as we have defined it will only display a red square, not a blue or a yellow square, and in only one fixed position on the screen. We could, of course, write other procedures to display other coloured squares and in different places, although the instructions in each case would all be very similar. BBC Basic has a further feature that helps in writing flexible procedures.

Let's suppose that, in general, we want to display a square of colour 'colour', and in position 'x' and 'y' on the screen. 'colour' is a variable to which we can give a value in the range 1 to 7 to specify the colour of the square (we shall assume we are using Mode 2, other modes allow fewer colours). The colours and their numbers are all listed on page 223 of the User Guide. You will see that we could use the range 0 to 15 if we are prepared to allow black and flashing colours as well.

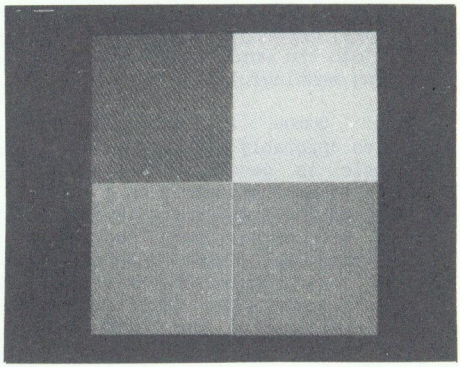
The values that we shall give to 'x' and 'y' eventually, will mark the position of the bottom left hand corner of the square, and will be in the range of 0 to 1279 and 0 to 1023. These are the normal ranges used in drawing in any graphics mode on the screen, and again are explained in the User Guide on page 56. Our revised version of the procedure would look like this:

```
1000 DEF PROCsquare(colour,x,y)
1010 GCOL 0,colour
1020 MOVE x,y:MOVE x,y+400
1030 PLOT 85,x+400,y:PLOT 85,x+400,y
+400
1040 ENDPROC
```

This follows exactly the same pattern as before, except that the values used then are now replaced by the variables 'colour', 'x' and 'y'. These are called the 'parameters' of this procedure.

Here now is a short program, to use this procedure, which draws one red, one yellow, one blue and one green square on the screen in mode 2:

```
100 MODE 2
110 PROCsquare(1,240,512)
120 PROCsquare(3,640,512)
130 PROCsquare(4,240,112)
140 PROCsquare(2,640,112)
150 END
```



In working out the numbers for 'x' and 'y' remember that x=640 and y=512 represents the centre point on the screen, and that our square has a side of length 400.

You might like to try rewriting the procedure definition for yourself, and making it even more useful, so that each time a square is displayed it can

be any size you choose. To do this you will need to use a fourth parameter, called for example 'size', and use this in defining the procedure instead of the fixed value of 400. Once you have done this, then there are all sorts of short example programs you can write to try out your new procedure. Most of these ideas use either a REPEAT-UNTIL loop, or a FOR-NEXT loop. One idea is to use the random number function (RND) to repeatedly calculate a random colour, and random position, for a whole series of squares. You could also try displaying a small coloured square in the bottom left hand corner of the screen, followed by larger and larger squares as you move towards the top right hand corner of the screen. If you know about sines and cosines then you could draw a sine (or cosine) curve, displaying a small coloured square in each position. I am sure you will be able to think of many other ideas.

One tip that is well worth remembering when you are developing a new procedure is that a procedure can be called in immediate mode as well as in a program, provided that you select the correct mode first. If, for example, you rewrite the procedure for the square to give a variable size as described above, then test it out by typing:

```
MODE 2
PROCsquare(640,6,0,0)
```

You should see a cyan coloured square on the screen that is exactly half the screen width and positioned at the bottom left hand corner. This does assume that you have defined your procedure so that the four parameters are in the order 'size', 'colour', 'x', and 'y'.

Now that you have seen how a procedure works, why not have a look at some of the other programs in the magazine, and the procedures that they use. Some of these will be quite complicated and difficult to understand, but you should be able to see how they all follow the same basic pattern we have described above. Looking at other peoples programs is a very good way learning how to write your own.

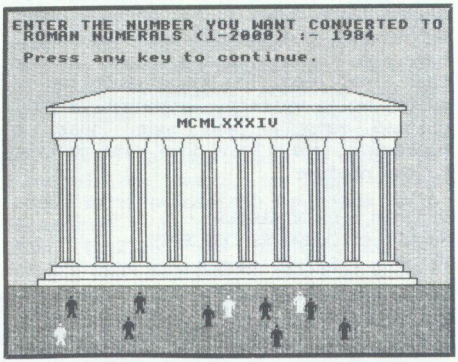
Next month, when you have had time to practise writing and using your own procedures a little, we will look at how the use of procedures makes the task of designing and writing a whole program so very much easier, for expert and non-expert alike.

SCREEN TO PRINTER DUMPS

The latest ROMs reviewed by Robert Barnes

We are frequently asked at BEEBUG for advice on a suitable program to produce a screen dump on a printer. Until recently the only answer was to have a screen dump routine incorporated with your program, or loaded from cassette or disc when required. Recently a number of companies have brought out screen dump programs in ROM. These provide a comprehensive range of screen dump facilities which are immediately available using simple commands.

Program : DUMPOUT2
Supplier: Watford Electronics
Price : £18.00 (inc.)



DUMPOUT2 is yet another ROM to join Watford's already large range of ROM based software. This one allows the user to produce variable size, two tone or eight tone dumps of the screen in any of the graphic modes (0,1,2,4,5). An eight tone screen dump is produced by using different patterns of dots giving the effect of grey tones, which can be used to replace the colours. The tones provided by this ROM were easy to distinguish and produced very pleasing results. The comprehensive manual supplied with the ROM has obviously had a lot of thought put into it. The screen dumps are accessed using simple commands followed by a number of optional parameters. The parameters can be used to select colour masking and the area of the screen to be dumped. This ROM is designed to work with the following printers: Seikosha GP80/100/250, Star, NEC, EPSON MX/RX/FX, TANDY LPVII and the DMP100/120/200/400.

Program : GDUMP
Supplier: D.A. Computers Ltd.
Price : £20.00 (inc.)

This is a similar product to the Watford DUMPOUT2 but does cater for a slightly different range of printers including Epson FX/MX/RX, Shinwa CP80, NEC8023, STAR DP510 and Seikosha GP80/100/250. GDUMP has only one command '*GDUMP', but up to nine parameters may be included. These allow everything that DUMPOUT2 does, plus the option to print the dump at an angle of 90 degrees, which is useful in that it allows large dumps to fit on to a page. GDUMP has the largest help facility I have seen to date, being pages rather than lines long. The manual supplied is quite well written, although a full printed manual, and not just a photocopy would have done more justice to the product.

Program : PRINTMASTER
Supplier: COMPUTER CONCEPTS
Price : £33.35 (inc.)

This ROM is based upon a slightly different concept from the other two, in that it is a combination of a printer dump ROM and a printer toolkit. At £33.35, Printmaster costs significantly more than the other two, but also has a lot more to offer. As with all of Computer Concepts ROMs, the manual is well written and easy to follow containing 32 pages of clearly printed information.

Printmaster at the moment is designed solely for use on Epson printers, although versions for other printers are being written, and is thus able to make full use of the features available. Printmaster allows you to design your own characters for use on the printer, spool text out to the printer (i.e. allows a file to be printed whilst you use your machine for another purpose. This is a real time saver, especially when using WORDWISE), and provides a screen to printer dump in any mode. It also provides support for other Epson printer features such as italic script, indeed any Epson font, line spacing, margins and a host of others. Not all of the facilities are available on every Epson of course. I was most impressed by this piece of firmware and the facilities it included.

All of these products can be thoroughly recommended although it is a great pity that a Mode 7 screen dump is only included with Printmaster and not with the other two ROMs. If you just want a screen dump in any graphics mode, then DUMPOUT2 and GDUMP can both be useful. My personal preference here was DUMPOUT2 which produced slightly better results then GDUMP, but this obviously depends on which printer you are using. However, if you have an EPSON printer I feel there is no better choice at present than Printmaster, which enables you to make the most of the features these printers offer.

PRINTMASTER ROM FROM COMPUTER CONCEPTS

We have decided to offer the Printmaster ROM at the special offer price of £30 inclusive. Further details appear in the Software Club pages in the supplement.

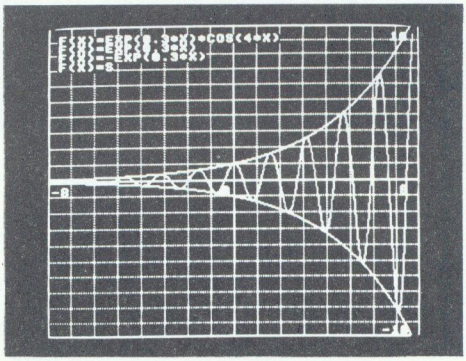
BEEBUG members with printers other than Epsons, will be pleased to know that Computer Concepts will be releasing Printmaster for other popular makes, and we expect to offer these to members in the same way.

Tested on Basic I & II and O.S. 1.2

MULTI-FUNCTION GRAPH PLOTTER

by Roel Grit

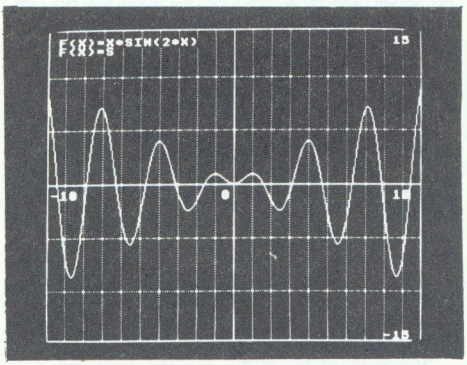
Following hot in the footsteps of last month's 'Colourful 3D Bar Chart' program, we present a program which plots continuous graphs of mathematical functions.



The bar chart generator provided a very colourful method of displaying non-continuous data, for example, daily rainfall, or monthly sales figures. However, if you want to plot a mathematical function (where Y is a function of X) then a quite different form of display is required and this is provided by the following program.

The unusual feature of this program is the inclusion of sound effects as the graph is being drawn. The Beeb emits a suitable sound whose pitch changes according to the rising and falling path taken by the graph. For the purists amongst you, there is the option of switching off the sound.

Initially the program allows you to set up various parameters, such as the minimum and maximum values for both the X and Y axes. Pressing Return without entering a value will select the default values of -3, 6, -2 and 5 for the minimum and maximum values of X and Y respectively. At this stage you can also choose to have sound effects, and whether you want marked lines or not. If the latter option is selected, then the graphs are drawn with dots attached at regular intervals. As you plot more functions on the same graph, the dot spacing is increased to distinguish between the different functions.



The grid is drawn initially and the scales are marked against the axes. You are then prompted to enter any function within the constraints listed below and the program will then plot the function. The function may be any valid Basic arithmetic expression based on the single variable X and may include powers, PI, LOG, SIN, COS, TAN, EXP and SQR. The program automatically handles difficult situations such as out of range logs, division by zero, and negative roots. The functions are entered in the same manner as would be expected for Basic, so that values passed to trigonometric functions will be assumed to be in radians, unless you have included the function RAD to convert from degrees to radians. Once a function has been plotted, the program prompts for another function. If another function is entered, its graph is plotted on the existing display. Pressing Return at this point forces the program to clear the screen, and re-draw the grid with the same

parameters. Entering 'S' will call up your own printer dump to make a hard copy of the graphs. (This must be supplied by the user and located in the procedure PROCscreendump defined at 11000. See BEEBUG Vol.1 No.9 for screen dumps for Epson and Seikosha printers.) Pressing Escape at any time will re-start the program.

Some functions which you could try out on this program are shown below:

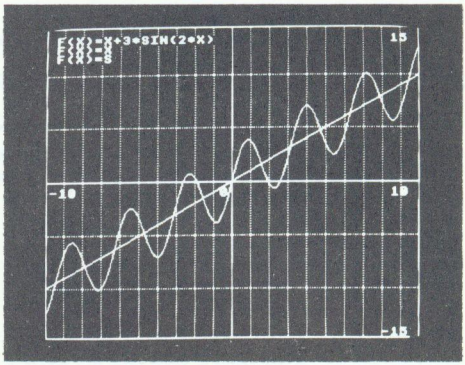
min.X	max.X	min.Y	max.Y	function
-3	6	-3	8	X^2
-3.14	3.14	-2	2	SIN(2*X)
-8	8	-10	10	EXP(0.3*X)* COS(4*X)
-8	8	-10	10	EXP(0.3*X)
-8	8	-10	10	-EXP(0.3*X)
-3	12	-2	11	10^(SINX)

It is well worth experimenting with any function that takes your fancy.

PROGRAM NOTES

The program is well structured, and it is easy to follow how the program operates. Below is a brief overview of the procedures and functions used.

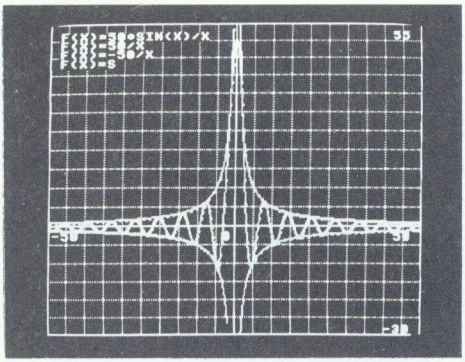
1000 PROCminmaxxy - sets up the minimum and maximum values for X and Y on the screen.
2000 PROCgrid - draws the grid.
3000 PROCxyaxis - draws the two axes.
4000 PROCscale - prints the minimum and maximum values of the scale on the grid.
5000 PROCplot - plots the function.
6000 PROCinstructions - displays the instructions for using the program.
7000 PROCtitle - sets up the screen titles.
8000 FNX - selects the next X value.
9000 FNY - selects the next Y value.
10000 PROCframe - draws the frame around the grid.
11000 PROCscreendump - room for the users printer driver screen dump.



The program could be altered at line 260 to run in Mode 4 for a Model A if required. In this case the resolution will be reduced, but screen dumps might turn out even better with thicker lines.

```
  10 REM Program GRAPHS
  20 REM Version B1.2
  30 REM Author R.Grit
  40 REM BEEBUG April 1984
  50 REM Program subject to copyright
  60 REM:
  99 flag=FALSE
 100 ON ERROR GOTO 340
 110 AMP=0:D=-2:DL=FALSE:C=0
 120 VDU 23,240,192,192,0,0,0,0,0,0
 130 MODE 7
 140 PROCtitle("MATHEMATICAL GRAPHS")
 150 IF flag THEN PRINT TAB(5,10)"DO Y
OU WANT TO STOP ?";:X$=GET$:PRINT X$:IF
 X$="Y" THEN MODE 7:END
 160 flag=TRUE
 170 PRINT TAB(5,12)"DO YOU WANT INSTR
UCTIONS ?";:X$=GET$:PRINT X$
 180 IF X$="Y" THEN PROCinstructions
 190 CLS:PROCtitle("MATHEMATICAL GRAPH
S")
 200 PRINT TAB(5,14)"DO YOU WANT SOUND
 (Y/N) ?";:X$=GET$:PRINT X$
 210 IF X$="Y" THEN AMP=-10
 220 PRINT TAB(5,16)"DO YOU WANT MARKE
D CURVES (Y/N) ?";:X$=GET$:PRINT X$
 230 IF X$="Y" THEN DL=TRUE
 240 PROCminmaxxy
 250 REPEAT
 260 MODE 0
 270 PROCgrid
 280 PROCxyaxis
```

```
  290 PROCscale
  300 PROCplot
  310 UNTIL FALSE
  320 END
  330 :
  340 ON ERROR OFF:IF ERR=17 THEN 100
  350 REPORT:PRINT" at line ";ERL
  360 END
  370 :
 1000 DEF PROCminmaxxy
 1010 MINX$="-3":MAXX$="6":MINY$="-2
":MAXY$="5"
 1020 PROCtitle("MIN. & MAX. X AND Y VA
LUES")
 1030 PRINT'"Now set the minimum and ma
ximum values"
 1040 PRINT"for the X and Y axes."
 1050 PRINT'"If you only press";CHR$(13
4);"<RETURN>";CHR$(135);"the computer"
 1060 PRINT"chooses the following value
s:"
 1070 PRINT'"MINIMAL X:";MINX$,"MINIMA
L Y:";MINY$
 1080 PRINT"MAXIMAL X: ";MAXX$,"MAXIMA
L Y: ";MAXY$
 1090 PRINT'';CHR$(131);"X VALUES"
 1100 INPUT TAB(5) "MIN. VALUE   " MINX$
 1110 IF MINX$=""THEN MINX$=MINX$:PRIN
T TAB(20,VPOS-1);MINX$
 1120 INPUT TAB(5) "MAX. VALUE   " MAXX$
 1130 IF MAXX$=""THEN MAXX$=MAXX$:PRIN
T TAB(21,VPOS-1);MAXX$
 1140 MINX=VAL(MINX$):MAXX=VAL(MAXX$)
 1150 IF MAXX<=MINX THEN VDU7:PRINTCHR$
129"Mistake - press SPACE BAR to try ag
ain":REPEAT UNTIL GET=32:GOTO 1010
 1160 PRINT;CHR$(131);"Y VALUES"
 1170 INPUT TAB(5) "MIN. VALUE   " MINY$
 1180 IF MINY$=""THEN MINY$=MINY$:PRIN
T TAB(20,VPOS-1);MINY$
 1190 INPUT TAB(5) "MAX. VALUE   " MAXY$
 1200 IF MAXY$=""THEN MAXY$=MAXY$:PRIN
T TAB(21,VPOS-1);MAXY$
 1210 MINY=VAL(MINY$):MAXY=VAL(MAXY$)
 1220 IF MAXY<=MINY THEN VDU7:PRINTCHR$
129"Mistake - press SPACE BAR to try ag
ain":REPEAT UNTIL GET=32:GOTO 1010
 1230 XUNIT=1279/(MAXX-MINX)
 1240 YUNIT=1023/(MAXY-MINY)
 1250 PRINT'CHR$134"Press any key":X$=G
ET$
 1260 ENDPROC
 1270 :
 2000 DEF PROCgrid
 2010 STX=1:STY=1
 2020 IF MAXX-MINX>25 THEN REPEAT STX=S
TX*5:UNTIL (MAXX-MINX)/STX<25
 2030 IF MAXX-MINX<4  THEN REPEAT STX=S
TX/5:UNTIL (MAXX-MINX)/STX>4
```



```
 2040 IF MAXY-MINY>25 THEN REPEAT STY=S
TY*5:UNTIL (MAXY-MINY)/STY<25
 2050 IF MAXY-MINY<4  THEN REPEAT STY=S
TY/5:UNTIL (MAXY-MINY)/STY>4
 2060 FOR Z=-MINX*XUNIT TO 0 STEP -XUNI
T*STX
 2070 MOVE Z,0:PLOT 21,Z,1023
 2080 NEXTZ
 2090 FOR Z=-MINX*XUNIT TO 1280 STEP XU
NIT*STX
 2100 MOVE Z,0:PLOT 21,Z,1023
 2110 NEXTZ
 2120 FOR Z=-MINY*YUNIT TO 0 STEP -YUNI
T*STY
 2130 MOVE 0,Z:PLOT 21,1279,Z
 2140 NEXTZ
 2150 FOR Z=-MINY*YUNIT TO 1024 STEP YU
NIT*STY
 2160 MOVE 0,Z:PLOT 21,1279,Z
 2170 NEXT Z
 2180 PROCframe
 2190 ENDPROC
 2200 :
 3000 DEF PROCxyaxis
 3010 MOVE -MINX*XUNIT,0:PLOT 5,-MINX*X
UNIT,1023
 3020 MOVE 0,-MINY*YUNIT:PLOT 5, 1279,-
MINY*YUNIT
 3030 ENDPROC
 3040 :
 4000 DEF PROCscale
 4010 VDU 5
 4020 MOVE 1240-LEN(MINY$)*32,32:PRINT;
MINY
 4030 MOVE 1240-LEN(MAXY$)*32,992:PRINT
;MAXY
 4040 MOVE 10,-MINY*YUNIT-20:PRINT;MINX
 4050 MOVE 1240-LEN(MAXX$)*32,-MINY*YUN
IT-20:PRINT;MAXX
 4060 MOVE -MINX*XUNIT-45,-MINY*YUNIT-2
0:PRINT;0
```

```
 4070 VDU 4
 4080 ENDPROC
 4090 :
 5000 DEF PROCplot
 5010 PRINT:B=1
 5020 VDU4:INPUTTAB(1,B) "F(X)=" F$:PRO
Cframe
 5030 IF F$="" THEN ENDPROC
 5040 IF F$="S" THEN PROCscreendump:GOT
O 5020
 5050 VDU 5:X=MINX:D=D+2
 5060 ON ERROR X=X+(MAXX-MINX)/320:ON E
RROR OFF:GOTO 5060
 5070 MOVE FNX(X),FNY(X)
 5080 ON ERROR IF ERR=18 OR ERR=21 OR E
RR=22 OR ERR=23 OR ERR=24 THEN GOTO 513
0 ELSE IF ERR=17 THEN 110 ELSE IF ERR=1
3 THEN 250
 5090 P=FNX(X):Q=FNY(X)
 5100 *FX21,5
 5110 IF Q<=0 OR Q>=1023 THEN MOVE P,Q:
ELSE SOUND 1,AMP,Q/5,5:DRAW P,Q
 5120 C=C+1:IF C>D AND DL=TRUE THEN C=0
:PRINT CHR$(240):MOVE P,Q
 5130 X=X+(MAXX-MINX)/320
 5140 IF X<=MAXX THEN 5090
 5150 B=B+1:*FX21,5
 5160 GOTO 5020
 5170 ENDPROC
 5180 :
 6000 DEF PROCinstructions
 6010 PROCtitle("INSTRUCTIONS")
 6020 PRINT'"This program allows you to
 plot graphs"
 6030 PRINT"of any mathematical functio
n."
 6040 PRINT'"You can select the minimum
 and maximum"
 6050 PRINT"values for the two axes."
 6060 PRINT'"You can mark the diferent
curves in the"
 6070 PRINT"diagram with the special dr
awing option"
 6080 PRINT"which places dots at regula
r intervals"
 6090 PRINT"along the curve, depending
on which"
 6100 PRINT"curve is drawn."
 6110 PRINT'"Pressing ESCAPE at any tim
e will start"
 6120 PRINT"the program again. Pressing
 RETURN in"
 6130 PRINT"plot mode clears the screen
."
 6140 PRINT'"Entering S will start the
screen dump"
 6150 PRINT"procedure, however you must
 insert your"
 6160 PRINT"own procedure relating to y
our printer."
 6170 PRINT'CHR$134"Press any key":X=GE
T
 6180 ENDPROC
 6190 :
 7000 DEF PROCtitle(TITLE$)
 7010 CLS:SP=INT((40-LEN(TITLE$))/2)
 7020 VDU 157,132,141:PRINT TAB(SP-4);T
ITLE$
 7030 VDU 157,132,141:PRINT TAB(SP-4);T
ITLE$
 7040 ENDPROC
 7050 :
 8000 DEF FNX(X)=(X-MINX)*XUNIT
 8010 :
 9000 DEF FNY(X)=(EVAL(F$)-MINY)*YUNIT
 9010 :
10000 DEF PROCframe
10010 MOVE 0,0:DRAW 1279,0:DRAW 1279,10
23:DRAW 0,1023:DRAW 0,0
10020 ENDPROC
10030 :
11000 DEF PROCscreendump
11010 REM **********************
11020 REM PLACE YOUR PRINTER DUMP
11030 REM     ROUTINE IN HERE
11040 REM **********************
11050 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DEFINING YOUR OWN FUNCTION KEYS IN VIEW - J.Wellings

The function key buffer is preserved while View is in use, although the definitions associated with them cannot normally be recalled by pressing the keys as they are used to perform special functions. The command *KEY still functions in View's command mode. The functions can be recalled by pressing Control and Shift in conjunction with the required function key, providing *FX228,1 is entered in command mode first.

RESETTING THE COMPUTER DURING A PROGRAM - A.Crowhurst

The reset vector can be directly called from within a program using CALL !&FFFC. This has exactly the same effect as pressing the Break key during the running of the program.

COMPUTER CONCEPTS GRAPHICS ROM

Reviewed by Mike Siggins

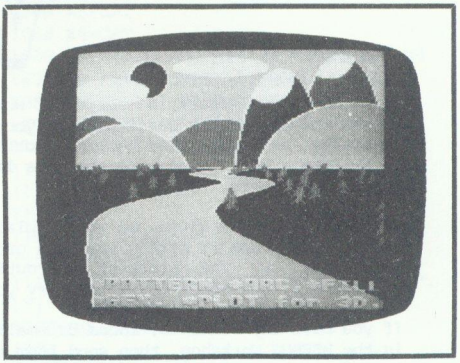
Product :Graphics Extension ROM.
Price :£33.35 (inc.)
Supplier:Computer Concepts, 16 Wayside, Chipperfield, Herts WD4 9JJ.

The Graphics Extension ROM is a welcome addition to the growing range of graphics software and add-ons now available for the BBC micro. The ROM provides many features, including sprites and turtle graphics, to supplement the already extensive routines available on the Beeb. There are, in addition to this, some general purpose commands which provide useful additions to the graphics capability of the Beeb. All the commands are accessible from Basic, in the manner of standard 'star' commands.

The ROM is packaged to the normal professional standards associated with previous products from Computer Concepts. Installation is simply a matter of inserting the ROM into a socket, preferably one of high priority for speed. The documentation supplied is of a good standard, providing detailed and clear explanations of all the commands and parameters.

The sprite commands are very comprehensive and provide the basis for animation techniques. Commands are provided to create, edit and display up to 32 multi-coloured sprites which can be up to 24 pixels square. They can then be combined to produce animated sequences using the powerful *FILM command. The only problem here is the knowledge of the memory map required to derive the best results, especially considering the limited memory available. The speed with which the sprites can be moved about is impressive, and can be extremely smooth. The worst results are experienced when the sprites are moved diagonally.

The Turtle graphics commands emulate those available in the Logo language. The range covered gives all the primitives for some very interesting graphics. The speed is again impressive, and would probably benefit



from some delay routines if the commands were used in the context of a Logo interpreter. As one would expect, all the colours are available making for some very spectacular displays.

There are many other commands which all go to provide a utility worthy of extended experimentation. They include a versatile plot command which can use the existing Basic parameters and combine them with 3-D co-ordinates. These routines can provide some amazing effects. There is also a print command which can display text at any angle, size or colour. The output is unfortunately somewhat untidy as it seems to merely magnify the existing character set. This causes unsightly 'steps' as the characters grow in size.

Other commands enable one to scale and rotate the screen, compose spiral patterns, rapidly draw arcs and circles. An extremely useful item is a system of 'GFX' commands which provide a suite of utilities for the chip itself, including a powerful sine and cosine look-up facility.

Overall, this product is good value for money and will provide anyone interested in graphics with a whole new field of possibilities. It is worth remembering that any programs developed using these features will only run on a machine in which the Graphics ROM is installed.

[Note: BEEBUG have arranged a special offer on this ROM for members at the 'all-in' price of £30. See elsewhere in this issue - Ed.]

Programmers Workshop

by SURAC

This month we start a series of workshops in which we will pass on all the best tricks and techniques for improving your programs, particularly in Basic. We will regularly be presenting helpful routines, functions and procedures that will build up to form a useful and comprehensive library, of interest to both beginners and experienced users.

For each routine given, we will explain clearly how it works, and include notes on how you could use it within your own programs, and how it could be tailored to your own needs. By the nature of the routines given, some of them will be more complex than others, but don't let this worry you!

If you have any comments, ideas or small routines that you would like to be included in the BEEBUG Workshop, then send them to SURAC, at the BEEBUG editorial address. Any of your contributions published will be paid for at the normal magazine rates.

Tested on Basic I & II and O.S. 1·2

AN ELEGANT DATA ENTRY ROUTINE

To start this series of workshops, we will look at a very useful function that replaces the Basic INPUT statement and then provides a number of helpful and user friendly facilities. When using INPUT, you are not restricted in the number of characters that you may enter. For example, when using INPUT to input a single number, you could easily hold down a single key, and the Beeb's auto repeat facility would then result in up to 254 extra characters being entered. This can have disastrous results during data entry in a program that relies upon tidy presentation of data. We now present a function that enables you to easily overcome these limitations. The facilities that the new input function provides are:

1. Text or data entry at any screen position.
2. Display of a guiding line of full stops to indicate the number of characters required.
3. Limits to the specified number of characters to be entered.
4. Program controllable conversion of lower case characters to upper case.
5. Program controllable deletion of the guide line once input has finished.

In order to use the function, you need to supply four parameters. The four parameters are:

1 Horizontal screen position of start.
2 Vertical screen position of start.
3 Maximum string length allowed.
4 Switchable options.

For example, if you wanted to input a string from the screen at the position 10,20 (i.e. 10 places from the left, on line 20), with a maximum of 13 characters, you could use:

```
reply$=FNinput(10,20,13,3)
```

(See the section later on for an explanation of the last parameter.) This example would produce the following prompt at the screen position 10,20:

```
[.............]
```

The user enters all of his text within the bounds of the surrounding brackets, with the number of full stops indicating how many characters are to be entered (at a maximum). The two enclosing square brackets could be omitted by leaving them out of line

1160. Using a ready-made function of this kind, makes the acquisition of text and data from the screen considerably easier than having to write your own routines and tests.

The method of inputting text is very similar to normal entry via Basic's INPUT statement. Pressing a key results in the character being added to the string (see later), and the cursor moved one place to the right. The Return key terminates the string, as it would normally. Delete can be used to delete a single character and CTRL-U removes the entire line. When a character is deleted, it is replaced by a full stop, to signify that there is no character there.

If you attempt to delete a character when there is nothing to delete, or add a character when the maximum number has already been entered, a beep will be sounded. If this is not required, then all occurrences of 'VDU 7' in the program should be deleted.

Two switchable options, which are controlled by the fourth parameter, are provided. These are the automatic conversion of lower case to upper case and the removal of the prompt line after entry of the text or data. In the example, all lower case letters entered will be converted to their upper case equivalents, and the prompt will be erased on exit from the function. To select the value of the fourth parameter that is applicable to your particular requirements, just match up the values given in the table below with the options that you require.

Fourth Parameter	Case Conversion	Prompt Deletion
0	No	No
1	Yes	No
2	No	Yes
3	Yes	Yes

The function should be appended to your program, with care being taken to ensure that the line numbers do not overlap. Note that all of the extra procedures should also be added, as these are essential.

```
1000 DEF FNinput(X%,Y%,L%,F%)
1010 IF F% AND 1 M%=&DF ELSE M%=&FF
1020 IF F% AND 2 F%=-1 ELSE F%=0
1030 PROCinit
1040 REPEAT
1050 K%=GET
1060 IF K%>96 AND K%<123 K%=K% AND M%
1070 IF K%=13 UNTIL-1:PROCend:=R$
1080 IF K%=127 PROCdel:K%=0
1090 IF K%=21 PROCinit:K%=0
1100 IF K%<32 K%=0
1110 IF K% PROCadd
1120 UNTIL0
1130 :
1140 DEF PROCinit
1150 C%=0:R$=""
1160 PRINTTAB(X%-1,Y%)"["STRING$(L%,".
")"]"TAB(X%,Y%);
1170 ENDPROC
1180 :
1190 DEF PROCend
1200 IF F% PRINTTAB(X%-1,Y%)SPC(L%+2);
1210 ENDPROC
1220 :
1230 DEF PROCdel
1240 IF C%=0 VDU7:ENDPROC
1250 C%=C%-1
1260 R$=LEFT$(R$,C%)
1270 VDU8,ASC".",8
1280 ENDPROC
1290 :
1300 DEF PROCadd
1310 IF C%=L% VDU7:ENDPROC
1320 C%=C%+1
1330 R$=R$+CHR$K%
1340 VDUK%
1350 ENDPROC
```

The function could easily be extended by allowing the routine to exit only if a string's length is between an upper and a lower limit. You could also extend it further by allowing the cursor keys to move along the current string on the screen, to allow insertion and deletion of characters in the string. If you think they are unnecessary, then you could alter the program so that the square brackets are not displayed.

ERRATA TO BEEBUG VOL.2 NO.10 (HOME ACCOUNTS - ANNUAL BUDGETS)

Line 1410 of this program on page 13 was regrettably omitted while the magazine was at the printers. The line, which is essential to the correct running of the program should read:
1410 PROCshowcell(NUM%(Y%,X%)):*FX15

Most of you will probably have noticed the break in the printed listing at this point. We apologise for any inconvenience caused.

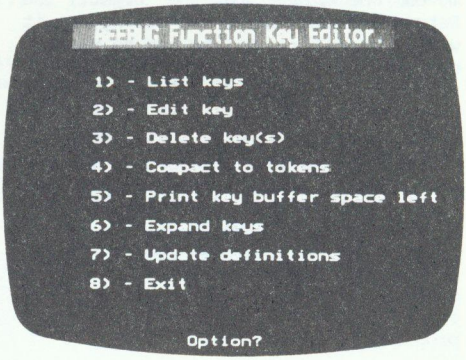
Tested on Basic I & II and O.S. 1.2

FUNCTION KEY EDITOR

by David Fell and Alan Webster

The Beeb allows you to define up to 16 different function keys with any characters in them but it does not, however, provide any easy method of editing them. We now present a very helpful utility that allows editing of any function key, and the display of some other very useful information about the current function keys settings.

The function key editor listed below allows you to list the current key definitions in their 'source' form, to edit them, to compact and expand from text to Basic tokens and back, and to display the amount of spare memory left in the key buffer. It is very easy to use, with a menu guiding the user through the various stages. When dealing with the tokens, the program automatically decides which version of Basic (I or II) is installed, and adapts to this. Because the program directly accesses I/O memory, it cannot be used with the program working in the second processor. To rectify this, just turn off your second processor, and then restart the loading process.



LIST THE KEYS

When run, the program pauses for a moment whilst it constructs a table of Basic keywords and tokens, and the current key definitions in their 'source' form. Once this is done, you are presented with a menu offering you eight possible options. From this list, the option most often chosen initially is the first one, i.e. list the key definitions. This allows you to check what the current definitions are. You might find it useful to use the editor to enter your function key definitions as well.

THE EDIT OPTION

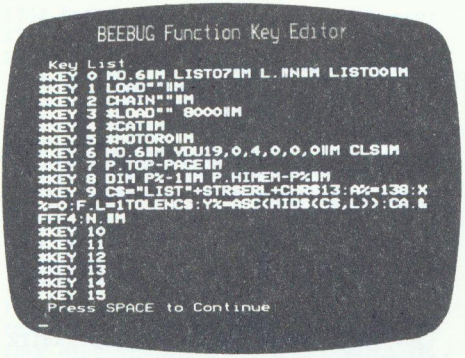
The 'edit' option first asks for the key number that you wish to change, and then displays the current 'source' definition of this key. You are then prompted to enter your new definition. The normal cursor key operations are enabled, and the same basic principles for editing a line of Basic apply: i.e. copy down those portions of the original line that you want, skip the ones that you do not want, and type in any new text. The string is entered when you press the Return key. Note that just pressing Return will not blank out a key definition in order to guard against accidental deletion. To delete a key, use the delete option described below.

THE DELETE OPTION

The 'delete' option asks you for a range, and this allows you to delete a single key (enter both the start and finish values as those of the single key which you wish to delete), or to delete a continuous group of keys (e.g. keys 4 to 8). Note that the actual key definitions are retained until the 'update' option is called. This may be called directly (option 7), or via the two token options (options 4 and 6), and by the memory display option (option 5).

COMPACTING THE KEYWORDS

The 'compact-to-tokens' option allows you to enter a string containing Basic keywords, and have these automatically changed into the corresponding single byte tokens that Basic uses to store them. When held in the key buffer, these take



significantly less space than the ASCII equivalent (see BEEBUG V.2 No.9 page 34). Note that this routine has been made as effective as is easily possible in so short a space, but it may occasionally not operate in the way expected. The keywords should be entered in full, although it is not absolutely essential for there to be spaces between the words. If not entered in full, the program will not compact them, and will leave them untouched; unless what you typed in contains sufficient to make another keyword (e.g. ENDP. for ENDPROC will be detected as END and P.).

ANY SPACE LEFT?

The operating system does not provide any easy method of discovering how much free space is left for key definitions. The message 'Bad key' can occur when an attempt to define past the available space is made, but no warning is given that this may happen. This option displays the amount of free space in the current definitions, and then displays the contents of the function key buffer (part of memory) in a 'hex dump' format.

EXPANDING A TOKEN

If you have previously saved some compacted key definitions, and then forgotten what the original keywords were, then this option is very helpful; for it reverses the effect of the compact option. Any values that might appear to correspond to a token, but in fact do not, (e.g. the OSCLI and OPENIN tokens when attempting to expand on Basic I) will be left in the format encountered.

UPDATING THE KEY DEFINITIONS

The program works with a copy of the key definitions, and it is necessary to use the update option to make the new versions active before leaving the program if you require these new definitions. This guards against accidental deletion of any definitions, and any mistakes that you may make whilst defining the keys. (Note though that the program does occasionally update the keys itself to aid in processing.)

THIS WAY OUT!

The eighth menu option allows you to exit from the program. Note the comments made in the above paragraph though, regarding the updating of key definitions.

```
  10 REM PROGRAM FUNCTION KEY EDITOR
  20 REM AUTHORS D.A. FELL /
                 A.R. WEBSTER
  30 REM BASED UPON A  PROGRAM FROM
                   M.A. SHELLEY
  40 REM BEEBUG MAY 1984
  50 REM VERSION 1.5
  60 REM PROGRAM SUBJECT TO COPYRIGHT.
  70 :
 100 DIM K$(16)
 110 MODE7:PROCtidy(1,CHR$157+CHR$131+
"BEEBUG Function Key Editor. "+CHR$156,
2):VDU28,0,24,39,3,14:PRINT''':PROCtidy
(2,"Please wait",2)
 120 PROCreadkeys:PROCsourcekeys
 130 PROCinit
 140 REPEATCLS
 150 ON ERROR GOTO 280
 160 PRINT'TAB(7)"1) - List keys"
 170 PRINT'TAB(7)"2) - Edit key"
 180 PRINT'TAB(7)"3) - Delete key(s)"
 190 PRINT'TAB(7)"4) - Compact to toke
ns"
 200 PRINT'TAB(7)"5) - Print key buffe
r space left"
 210 PRINT'TAB(7)"6) - Expand keys"
 220 PRINT'TAB(7)"7) - Update definiti
ons"
 230 PRINT'TAB(7)"8) - Exit"'''
 240 PROCtidy(6,"Option? ",1):REPEAT S
=VAL(GET$):UNTILS>0 AND S<9
 250 CLS:IFS=1 PROC1k:PROCspace ELSEIF
 S=2 PROCed ELSEIF S=3 PROCdel ELSE IFS
=4 PROCco ELSE IFS=5 PROCkeyspare ELSE
IFS=6 PROCexpand ELSE IF S=7 PROCupdate
 ELSE IF S=8 THEN PROC1k:END
 260 UNTILFALSE
 270 :
```

```
  280 CLS:ON ERROR OFF:IF ERR<>17 REPOR
T:PRINT" at line ";ERL
  290 IF ERR=251 THEN PRINT $&A00
  300 END
  310 :
 1000 DEF PROClk :LOCALT%,S:PRINTCHR$13
0"Key List.":FORT%=0 TO15:PRINT"*KEY ";
T%;" ";K$(T%):NEXT:ENDPROC
 1010 :
 1020 DEF PROCed :PROCtidy(3,"Use curso
r keys to copy",1):PROCtidy(3,"Use dele
te key to delete",1):PROCtidy(3,"Use re
turn to enter definition",1):PROCtidy(3
,"and return to menu.",1)
 1030 REPEAT INPUT'"Edit which function
 key ",K:UNTIL K>-1 AND K<16
 1040 PRINT"KEY ";K":-"K$(K)'"KEY ";K;"
:";:INPUTLINE""D$:IF D$<>"" THEN K$(K)=
D$
 1050 ENDPROC
 1060 :
 1070 DEF PROCdel :REPEAT PROCtidy(2,"K
ey Deletion.",1):INPUTTAB(1,5)"Which ke
ys to delete (from, to) ",f%,t%:UNTIL f
%>-1 AND t%<16 AND NOT(f%>t%):FORL%=f%T
Ot%:PRINT"*KEY ";L%;" ";K$(L%):NEXT:PRI
NT"Delete ?  (Y/N)";:S=GET:S=S OR96
 1080 IF S<>121 ENDPROC ELSE FORD%=f% T
O t%:K$(D%)="":NEXT:ENDPROC
 1090 :
 1100 DEF PROCcl :LOCALA%:A%=18:CALL&FF
F4:ENDPROC
 1110 :
 1120 DEF PROCco :LOCAL D$,F%,T%,Z%:PRO
Ctidy(2,"Key Compaction.",1):PRINT'"Whi
ch key(s) to compact "'"(from,to) ";:RE
PEATINPUT,F%,T%:UNTIL F%>-1 AND T%<16 A
ND NOT(F%>T%)
 1130 PROCupdate:PROCreadkeys
 1140 FOR Z%=F% TO T%:K$(Z%)=FNstring(K
$(Z%)):NEXT:PROCsourcekeys:ENDPROC
 1150 :
 1160 DEF PROCL(S,N) :LOCAL I,J,B$,R:PRI
NT'"'_' signifies a control character."
 1170 FORI=S TOS+N STEP8:@%=4:PRINT~I;:
B$="  ":FORJ=0TO7:R=I?J:PRINT~R;:@%=3:R
=R+(R<32 ORR>126)*(R-95):B$=B$+CHR$R:NE
XT:PRINTB$:NEXT:@%=10:PROCspace:ENDPROC
 1180 :
 1190 DEF PROCkeyspare :LOCAL T%,H%:PRO
Ccl:PROCupdate:PROCtidy(2,"Memory Displ
ay.",1):PRINT'
 1200 T%=&B00:H%=0:REPEATT%=T%+1:IF ?T%
>H% H%=?T%
 1210 UNTILT%=&B10
 1220 IF H%>=&FE THEN PRINT"No key spac
e left":VDU7,7 ELSE PRINT"There are ";S
TR$(&FF-H%);" bytes left"
 1230 VDU28,0,24,39,7:PROCL(&B00,&FF):V
DU28,0,24,39,3
 1240 ENDPROC
 1250 :
 1260 DEF PROCexpand
 1270 PROCupdate:PROCreadkeys
 1280 LOCAL I%,S%,F%
 1290 CLS
 1300 PROCtidy(2,"Keyword Expansion.",1)
 1310 PRINT'''
 1320 REPEAT
 1330 INPUT"Start and Finish keys :"S%,
F%
 1340 UNTIL S>-1 AND F%<16 AND NOT(S%>F
%)
 1350 FORI%=S%TOF%
 1360 IF K$(I%)<>"" K$(I%)=FNexpl(K$(I%
))
 1370 NEXT
 1380 PROCsourcekeys
 1390 ENDPROC
 1400 :
 1410 DEF PROCspace
 1420 PRINTCHR$130"Press SPACE to Conti
nue:"
 1430 *FX15,1
 1440 REPEAT UNTIL GET=32
 1450 ENDPROC
 1460 :
 1470 DEF FNexpl(A$)
 1480 LOCAL I%,B$:B$=""
 1490 IF A$="" ="":
 1500 FORI%=1TOLENA$
 1510 B$=B$+FNexp2(ASC(MID$(A$,I%,1)))
 1520 NEXT
 1530 =B$
 1540 :
 1550 DEF FNchar(X%)
 1560 LOCAL A$
 1570 IF X%<32 THEN A$=A$+"|"+CHR$(X%+6
4)
 1580 IF X%>31 AND X%<&7F THEN A$=A$+CH
R$(X%)
 1590 IF X%=&7F A$=A$+"|?"
 1600 IF X%=&7C THEN A$=A$+"||"
 1610 IF X%>&7F THEN A$=A$+"|!":X%=X%AN
D127:GOTO 1570
 1620 =A$
 1630 :
 1640 DEF FNreadkey(K%)
 1650 LOCAL I%,A$
 1660 IF &B00+?(K%+&B01) < &B01+?(K%+&B
00) THEN =""
 1670 FORI%=&B01+?(K%+&B00) TO &B00+?(K
%+&B01)
 1680 A$=A$+FNchar(?I%)
 1690 NEXT
 1700 =A$
 1710 :
 1720 DEF PROCupdate
 1730 LOCAL I%,X%,Y%:Y%=10
 1740 FORI%=0TO15
```

```
1750 $&A00="KEY"+STR$I%+" "+K$(I%)
1760 CALL&FFF7
1770 NEXT
1780 ENDPROC
1790 :
1800 DEF PROCreadkeys
1810 LOCAL T%,S%,F%,A%,A$
1820 FORT%=&B00 TO &B10:A$="":FL%=2:@%
=10:S%=?T%:F%=FNRV:FORA%=S%+1 TO F%:A$=
A$+CHR$(A%?&B00)
1830 NEXT:IF F%=S% THEN A$=""
1840 K$(T%-&B00)=A$:NEXT
1850 ENDPROC
1860 :
1870 DEF FNRV :LOCAL H%,L%,R%
1880 H%=255:FORL%=&B00TO&B0F:R%=?L%:IF
R%>S% AND R%<H% AND R%<>S% THEN H%=R%
1890 NEXT:IF H%=255 THEN H%=S%
1900 =H%
1910 :
1920 DEF PROCsourcekeys
1930 LOCAL I%,J%
1940 FORI%=0TO15
1950 IF LENK$(I%) B$="":FORJ%=1TOLENK$
(I%):B$=B$+FNchar(ASC(MID$(K$(I%),J%,1)
)):NEXT:K$(I%)=B$
1960 NEXT
1970 ENDPROC
1980 :
1990 DEF PROCinit
2000 DIM KW$(128)
2010 PROCbasic
2020 PROCkeyword:PROCblank
2030 ENDPROC
2040 :
2050 DEF PROCbasic
2060 REM CHECK TO SEE WHICH VERSION OF

2070 REM BASIC IS IN THE MACHINE.
2080 IF ?&8015=50 B%=2:S%=&8071 ELSE B
%=1:S%=&806D
2090 ENDPROC
2100 :
2110 DEF PROCkeyword
2120 G%=S%
2130 REPEAT:W$="":REPEAT
2140 W$=W$+CHR$(?G%)
2150 G%=G%+1:UNTIL ?G%>127
2160 T%=?G%
2170 KW$(T%-127)=W$
2180 G%=G%+2
2190 UNTIL G%>&836D
2200 ENDPROC
2210 :
2220 DEF PROCsearch(K$)
2230 PRINTK$
2240 found=FALSE
2250 FOR A%=1 TO 128
2260 IF KW$(A%)=K$ found=TRUE:T%=A%:A%
=128
2270 NEXT
2280 IF found PRINTCHR$129;KW$(T%);CHR
$130;TAB(20);T%+127
2290 ENDPROC
2300 :
2310 DEF FNexp2(N%)
2320 LOCAL N$
2330 IF N%<128 N$=CHR$(N%) ELSE N$=KW$
(N%-127)
2340 IF N$="" N$=CHR$(N%)
2350 =N$
2360 :
2370 DEF PROCblank
2380 KW$(14)="":KW$(79)=""
2390 IF B%=2 KW$(65)=""
2400 IF B%=1 KW$(10)="":KW$(15)="":KW$
(53)="":KW$(69)="":KW$(112)="":KW$(128)
=""
2410 ENDPROC
2420 :
2430 DEF FNstring(S$)
2440 IF S$="" =""
2450 FORT%=128 TO 1 STEP -1
2460 IF INSTR(S$,KW$(T%))<1 OR KW$(T%)
="" GOTO 2500
2470 IF S$=KW$(T%) S$=CHR$(T%+127):T%=
0:GOTO 2500
2480 S$=LEFT$(S$,INSTR(S$,KW$(T%))-1)+
CHR$(T%+127)+RIGHT$(S$,LEN(S$)-(INSTR(S
$,KW$(T%))+LEN(KW$(T%)))+1)
2490 GOTO 2460
2500 NEXT
2510 =S$
2520 :
2530 DEF PROCtidy(C%,A$,H%)
2540 LOCAL I%
2550 A$=CHR$(128+C%)+A$
2560 IF H%=2 A$=CHR$141+A$
2570 IF H%=2 FOR C%=1 TO 2
2580 PRINTTAB(19-LENA$DIV2)A$
2590 IF H%=2 NEXT
2600 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SWAPPING FILING SYSTEM UTILITIES - A.Still

Some of the normal DFS utilities, such as *TYPE and *DUMP, will work with the cassette filing system, and not just the DFS. In fact some, if not all, of the commands work with other filing systems such as the ROM and Teletext filing systems. Note, the DFS must be fitted since the utilities reside in this ROM.

Tested on Basic I & II and O.S. 1.2

ACCURATE ARITHMETIC

by J. B. Miller-Smith

Unassisted, a BBC micro will give its answers correct to 9 significant figures; but for some applications greater accuracy is required. The routines below permit ultra-high precision arithmetic with answers to 1000 significant figures if you choose - the degree of precision may be selected by the user.

Computers and, for that matter, pocket calculators are very good at providing fast and accurate results to almost all arithmetic problems, and any limitations on their accuracy are generally ignored by most users as being insignificant and affecting perhaps only the last decimal place in a given answer. However there are occasions (such as solving computer magazine puzzles!) for which it is necessary to evaluate very big numbers absolutely accurately (what, for example, are the middle three digits of 7 to the power 50?) - there is also a certain aesthetic satisfaction in having your computer display such large numbers perfectly, and avoiding such dead end messages as 'Too big'.

The BBC computer holds integer variables (those whose names end with a % sign) in 4 bytes, giving a range of whole numbers from -2,147,483,648 to +2,147,483,647. Real variables are held in a floating-point format which increases the range significantly to cover plus or minus values from 2 x 10 to the power -39 to 2 x 10 to the power +38, but the accuracy is limited to 9 significant digits (8 decimal places). To avoid these limitations the following program deals with integer numbers, up to several hundred digits long if necessary, by holding the digits in arrays: the numbers are then multiplied digit by digit very much as in 'long multiplication'. The multiplication routine used in this program to evaluate large powers can obviously be used in other applications - but note that it is not easy to make it a stand-alone procedure because of the difficulty of passing arrays as parameters on the BBC micro.

```
  10 REM Program ARITH
  20 REM Version B0.2
  30 REM Author  Ben Miller-Smith
  40 REM BEEBUG  May 1984
  50 REM Program subject to copyright
  60 :
 100 MODE 7
 110 ON ERROR VDU 26:END
 120 VDU 15
 130 head$=CHR$(131)+"Powers of N list"
 140 PRINTTAB((40-LEN(head$)) DIV 2,1)
;head$
 150 :
 160 REM Set reasonable array sizes
 170 REM and max number of digits in N
 180 :
 190 asize%=1500
 200 max%=20
 210 DIM A%(asize%),N%(max%)
 220 PRINT
 230 PROCinputN: REM Get N and nsize%
 240 psize%=asize%+nsize%+1
 250 DIM P%(psize%)
 260 ndigits%=nsize%
 270 VDU 28,0,24,39,5
 280 :
 290 REM Initialise all arrays to zero
 300 :
 310 FOR I%=0 TO asize%:A%(I%)=0:NEXT
 320 FOR J%=0 TO nsize%:P%(J%)=0:NEXT
 330 P%(0)=1: REM Initial product=1
 340 E%=1: REM Power count
 350 :
 360 REM Copy last product to array A%
 370 REM and reset product P% to zero
 380 :
 390 REM   *** Main Loop ***
 400 :
 410 REPEAT
 420 FORI%=0 TO ndigits%:A%(I%)=P%(I%)
:P%(I%)=0:NEXT
 430 :
 440 REM Multipliction Routine
 450 :
 460 FORJ%=0 TO nsize%
 470 carry%=0
 480 FORI%=0 TO ndigits%+1
```

```
  490 prodij%=N%(J%)*A%(I%)+carry%
  500 psum%=P%(I%+J%)+prodij% MOD 10
  510 P%(I%+J%)=psum% MOD 10
  520 P%(I%+J%+1)=P%(I%+J%+1)+psum% DIV
10
  530 carry%=prodij% DIV 10
  540 NEXT I%,J%
  550 :
  560 REM Skip leading zero's and print
  570 REM product array
  580 :
  590 I%=psize%
  600 REPEAT
  610 I%=I%-1
  620 UNTIL P%(I%)<>0
  630 PRINT"N TO THE POWER ";
  640 IF E%<10 PRINT " ";E%;" = "; ELSE
PRINT;E%;" = ";
  650 E%=E%+1
  660 :
  670 FORJ%=I% TO 0 STEP -1
  680 PRINT;P%(J%);
  690 NEXT
  700 :
  710 ndigits%=I%
  720 PRINT
  730 UNTIL ndigits%>asize%-nsize%
  740 PRINT "Increase array size to rea
ch higher"
  750 PRINT "powers (i.e. change value
of 'asize%'"
  760 PRINT "near the start of the prog
ram)."
  770 VDU 26:END
  780 :
 1000 DEFPROCinputN
 1010 REM Get N digit by checked digit
 1020 REM until Return (&0D) entered.
 1030 REM Shift N left on each new entry.
 1040 REM Exit on more than 'max%' digi
 1050 :
 1060 PRINT "N = ";
 1070 nsize%=0
 1080 REPEAT
 1090 REPEAT
 1100 X%=GET-48
 1110 IF X%=&0D-48 PRINT:ENDPROC
 1120 UNTIL (X%>-1 AND X%<10)
 1130 PRINT;X%;
 1140 nsize%=nsize%+1
 1150 FORI%=max% TO 1 STEP -1
 1160 N%(I%)=N%(I%-1)
 1170 NEXT
 1180 N%(0)=X%
 1190 UNTIL nsize%=max%
 1200 ENDPROC
```

When RUN this program will await your entry of a value for N, and then list the power of N in ascending order, until you either Escape, or it runs out of array space. You may wish to remove line 80 while debugging the program after typing it in so that any errors are reported directly: when the program is running you should be able to answer the question about 7 to the power 50, easily.

```
                Powers of N list

 N = 1234

 N TO THE POWER  1 = 1234
 N TO THE POWER  2 = 1522756
 N TO THE POWER  3 = 1879080904
 N TO THE POWER  4 = 2318785835536
 N TO THE POWER  5 = 2861381721051424
 N TO THE POWER  6 = 3530945043777457216
 N TO THE POWER  7 = 4357186184021382204544
 N TO THE POWER  8 = 5376767751082385640407296
 N TO THE POWER  9 = 6634931404835663880262603264
```

DIVISION

In contrast to the long multiplication used above it is quite easy to improve the accuracy of division operations to any number of decimal places, at least when integers are involved (and decimal numbers can always be converted to integers by multiplying by the appropriate power of 10 if necessary). The following short program demonstrates the use of the DIV and MOD operators to provide an accurate answer for any integer division sum (except division by zero, of course).

```
  10 MODE 7
  20 PRINTTAB(3,1);CHR$131;"Exact valu
e of X% devided by Y%"
  30 INPUT "What is X% ? "X%
  40 INPUT "What is Y% ? "Y%
  50 PRINT
  60 Q%=X% DIV Y%
  70 PRINT Q%;".";
  80 REPEAT
  90 X%=(X% MOD Y%)*10
 100 Q%=X% DIV Y%
 110 PRINT;Q%;
 120 UNTIL FALSE
```

Given some initial values for X% and Y% (whose values are subject to the integer range restrictions mentioned in the introduction) this program will output the quotient X%/Y% to any number of decimal places, until halted by Escape or Break. You might like to evaluate some reciprocals (e.g. 1/7), and compare them with the computer's answer (PRINT 1/7). Incidentally, the fraction 355/113 gives a good approximation to PI (better than 22/7).

Tested on Basic I & II and O.S. 1.2

MACHINE CODE GRAPHICS (Part 4)

by Peter Clease

This month we will look at how to produce simple animation by means of machine code graphics, with particular reference to Mode 2.

If you have seen a cartoon, and then tried to make one yourself, you will know that the simulation of movement is produced by drawing a shape, leaving it on the screen for a while, then rubbing it out and drawing it in a slightly different position. If this is done many times, and fast enough, the eye accepts the series of pictures as a moving image. A similar method can be used on the micro to show, for example, a 'galaxian' moving across the screen. First, the 'galaxion' is plotted in an initial position, then left there for a short length of time, and then replotted in a slightly different position on the screen, after deleting the first image.

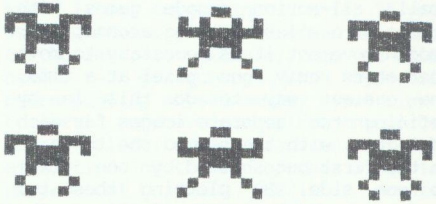
If you look at the earlier programs in this series you will see that they only plot characters at actual screen character positions, and not at places in between these. The positioning that this gives is far too coarse to be of much use for games, or most other applications. Moving a character across the screen is, however, relatively simple: assuming that &70 and &71 hold the address of a screen location (as in our earlier examples), then all we need to do is plot the required shape, wait, delete the shape, add 8 to the contents of &70 and &71, and plot the shape again. The reason for adding 8, which is equivalent to moving one byte of screen memory, is explained below.

The following program (Program 9) will move a space-invader across the screen. Notice that it plots the invader with its arms up, deletes it, plots the same space-invader slightly to the right, deletes, it and then plots the invader with its arms down twice. This gives a much smoother movement than by plotting the two characters alternately, because the two different forms of the space invader stay on the screen for longer, and are thus easier for the eye to follow.

```
  10 REM PROGRAM 9
  20 REM VERSION B1.1
  30 REM BEEBUG APRIL 1984
  40 REM AUTHOR PETER CLEASE
  50 REM PROGRAM SUBJECT TO COPYRIGHT.
  60 :
 100 PROCread
 110 PROCassemble
 120 MODE 2
 130 VDU 23,1,0;0;0;0;
 140 CALL start
 150 END
 160 :
1000 DEF PROCassemble
1010 DIM code 300
1020 FOR PASS=0 TO 3 STEP 3
1030 P%=code
1040 [
1050 OPT PASS
1060 .start
1070 LDA #0
1080 STA &70
1090 STA &72
1100 LDA #&30
1110 STA &71
1120 .loop
1130 BIT &FF
1140 BMI end
1150 JSR action1
1160 JSR action1
1170 JSR action2
1180 JSR action2
1190 LDA &72
1200 CMP #1
1210 BNE loop
1220 RTS
1230 .action1
1240 JSR plot1
1250 JSR wait
1260 JSR unplot
1270 JSR inc
1280 RTS
1290 .action2
1300 JSR plot2
1310 JSR wait
1320 JSR unplot
1330 JSR inc
1340 RTS
1350 .plot1
1360 LDY #0
1370 .loop1
1380 LDA base1,Y
1390 STA (&70),Y
1400 INY
1410 CPY #32
1420 BNE loop1
1430 RTS
1440 .plot2
1450 LDY #0
1460 .loop2
1470 LDA base2,Y
1480 STA (&70),Y
1490 INY
1500 CPY #32
1510 BNE loop2
1520 RTS
1530 .unplot
1540 LDY #0
1550 TYA
1560 .loop3
1570 STA (&70),Y
1580 INY
1590 CPY #32
1600 BNE loop3
1610 RTS
1620 .wait
1630 LDX #100
1640 .loop4
```

```
1650 LDY #255
1660 .loop5
1670 DEY
1680 BNE loop5
1690 DEX
1700 BNE loop4
1710 RTS
1720 .inc
1730 CLC
1740 LDA &70
1750 ADC #8
1760 STA &70
1770 LDA &71
1780 ADC #0
1790 STA &71
1800 BMI end
1810 RTS
1820 .end
1830 INC &72
1840 RTS
1850 ]
1860 NEXT
1870 ENDPROC
1880 :
2000 DEF PROCread
2010 DIM base1 31
2020 DIM base2 31
2030 FOR I%=0 TO 31
2040 READ I%?base1
2050 NEXT
2060 FOR I%=0 TO 31
2070 READ I%?base2
2080 NEXT
2090 ENDPROC
```

```
2100 DATA 0,0,10,15,0,0,5,10,0,5,10,15
,15,10,0,0,0,0,10,5,15,15,5,0,0,0,0,5,15,
0,0,10,5
2110 DATA 0,0,0,15,10,0,5,0,0,5,10,15,
15,10,0,10,0,10,5,15,15,5,0,5,0,0,0,15,
5,0,10,0
```



TECHNICAL NOTES ON PROGRAM 9

Again, the structure of this example program is based around two main procedures. The first one (PROCread) allocates space for the two sets of data, and reads them into the area given. The second one (PROCassemble) assembles the machine code into another area. Listed below are some details on the machine code section of Program 9.

LINES — MAIN FUNCTION

1060-1110
These lines set up the screen pointer, and the exit flag.

1130-1140
Tests the Escape flag directly, and branches to the exit routine if the top bit is set. (The top bit will be set if there is an 'Escape' condition.) By testing this flag, we can exit early if the user presses 'Escape'.

1150-1180
These lines call the routines to display the invader in the two different positions. Note the way that each routine is called twice; this causes the invader to be displayed on the screen for long enough in each position for the eye to register the two different shapes.

1190-1220
These few lines test for the end of the run, and exit if it has occurred.

1230-1280
The first level of the 'plot space invader' routine in one position.

1290-1340
The other space invader first level routine.

1350-1430
The first routine that actually takes the data, and puts it into screen memory.

1440-1520
The second routine to put data into screen memory.

1530-1610
This routine puts a zero into each byte of the appropriate screen memory to unplot the character. (By altering the value in the accumulator, you can alter the 'background' colour that the invader leaves.)

1620-1710
This section of code causes a delay by doing a lot of counting. Altering the value in the LDX # statement at line 1630 will alter the timing of the delay loop. (You could also alter the LDY statement, but this will not have such a notable effect.)

1720-1810
This subroutine updates the pointer in screen memory by eight.

1820-1840
The exit routine.

As was mentioned in last month's article, each byte in Mode 2 holds two horizontal pixels. This means that if you define a character and move it one byte to the left or right, then the resulting change will be a movement of two pixels either left or right. While this may not be too inconvenient on small all-action arcade games with hordes of nasties circling around, for smooth movement it is necessary to move characters only one pixel at a time. The easiest way to do this is by defining two separate images for each character, with the second one the same as the first but shifted by one pixel to one side. By plotting these two characters at the same position, the shape will appear to move very slightly (by one pixel, actually) left or right. Below is a one-pixel movement program (Program 10) that sends an arrow across the screen in a similar way to the previous space 'invader program'.

```
  10 REM PROGRAM 10
  20 REM VERSION B1.0
  30 REM AUTHOR PETER CLEASE
  40 REM BEEBUG APRIL 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT.
  60 :
 100 PROCread
 110 PROCassemble
 120 MODE 2
 130 VDU 23,1,0;0;0;0;
 140 CALL start
 150 END
 160 :
1000 DEF PROCread
1010 DIM base 79
1020 FOR I%=0 TO 79
1030 READ I%?base
1040 NEXT
1050 ENDPROC
1060 :
1070 DEF PROCassemble
1080 DIM code 300
1090 FOR PASS=0 TO 3 STEP 3
1100 P%=code
1110 [
1120 OPT PASS
1130 .start
1140 LDA #0
1150 STA &70
1160 LDA #&30
1170 STA &71
1180 .loop1
1190 LDY #0
1200 .loop2
1210 LDA base,Y
1220 STA (&70),Y
1230 INY
1240 CPY #40
1250 BNE loop2
1260 JSR wait
1270 LDY #0
1280 .loop3
1290 LDA base+&28,Y
1300 STA (&70),Y
1310 INY
1320 CPY #40
1330 BNE loop3
1340 JSR wait
1350 CLC
1360 LDA &70
1370 ADC #8
1380 STA &70
1390 LDA &71
1400 ADC #0
1410 STA &71
1420 BMI end
1430 BIT &FF
1440 BPL loop1
1450 .end
1460 RTS
1470 .wait
1480 LDX #20
1490 .loop4
1500 LDY #255
1510 .loop5
1520 DEY
1530 BNE loop5
1540 DEX
1550 BNE loop4
1560 RTS
1570 ]
1580 NEXT
1590 ENDPROC
1600 :
2000 DATA 0,0,0,4,4,0,0,0,0,0,0,12,12,
0,0,0,4,4,4,12,12,4,4,4,0,8,12,12,12,12
,8,0,0,0,0,8,8,0,0,0
2010 DATA 0,0,0,0,0,0,0,0,0,0,0,12,12,
0,0,0,0,0,0,12,12,0,0,0,8,12,12,12,12,1
2,12,8,0,0,8,12,12,8,0,0
```

To produce the data necessary for the arrow in the two alternate positions is very simple.

The coding of the pixels into bytes, suitable for direct insertion into screen memory, is done in the same way as the examples were last month. If you want, you can try coding them yourself, but we will list them here so that you can see clearly the correct coding.

For the arrow in the left of the two positions, the data is as follows:

```
Column 1   0,0,0,4,4,0,0,0,
Column 2   0,0,0,12,12,0,0,0
Column 3   4,4,4,12,12,4,4,4,
Column 4   0,8,12,12,12,12,8,0
Column 5   0,0,0,8,8,0,0,0
```

For the arrow in the right of the two positions, the following data applies:

```
Column 1  0,0,0,0,0,0,0,0
Column 2  0,0,0,12,12,0,0,0
Column 3  0,0,0,12,12,0,0,0
Column 4  8,12,12,12,12,12,12,8,
Column 5  0,0,8,12,12,8,0,0
```

Incidentally, by careful design of the two characters, there is no need for any 'blanking' routine, as each character plotted includes a strip of background to the left of it, and this

will progressively erase the previously plotted character, provided the movement is always to the right. The operation of Program 10, apart from the different method of erasing the shape, is very similar to that of Program 9, with a number of the routines being virtually identical.

MOVEMENT IN OTHER MODES:

Moving our characters horizontally in Mode 2 was easy because each byte only held the data for two pixels. This allowed us to store two different images of the character, and to place these onto the screen to give the impression of movement for each pixel within a byte's width. However, all the other graphics modes store the data for either four or eight pixels in each byte. To use the method above then, we would need to store either four or eight separate images of each character. This is extremely wasteful of memory even for small characters, and large characters could require more memory than there was available! If this has not put you off using one of the other modes (say because you need the extra resolution of Mode 1, or the memory economy of Mode 5), then you will need to use some other technique.

The method that I suggest requires more programming than for the 'Mode 2' method, and is slower in operation; however, it will still be quicker than using any operating system routines, so there are still some advantages. Although slower, you should find that if you write your game fully in machine code, and structure it well, then you may need delay loops, even for some of the 'slower' modes! What we have to do is to go back to basics. The technique that I am suggesting is this. If you want to move a given screen character to the left, for example, then you will need to go through each byte of corresponding screen memory (from right to left), and move each pixel one place to the left. This requires careful extraction of the appropriate bits via usage of the 6502's rotate instructions. Because of the way in which the bits for each pixel are interleaved (see article 2 & 3 in this series), rather than just following in sequence, this technique is much more complex than the rest of the techniques, and goes beyond the scope of this introductory series.

Next month, in the final article in this series, we will deal with how to move a character vertically, and present a general routine that will place a character at any point on the screen. We then will use this to produce a simple animation.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

RELOCATING CHARACTER DEFINITIONS - S.Smith

Redefining characters using *FX20,n to explode the soft character RAM allocation (see the User Guide p.427) can leave very little room for programs on a disc-based machine. It can also present problems if you try to relocate the program to reclaim some of the disc operating space.

One answer is to persuade the O.S. to put the definitions in a different place. Memory locations &368 to 36E contain the high-byte addresses of the pages used for the soft character explosion. It is possible to use pages &9 (sound and speech buffer) and &A (cassette or RS423 buffer) in which to store definitions, and even in page &B if the soft-keys are not required. For example, to redefine up to 64 characters (sufficient for an alternative character set) in the range &80 to &BF, try the following header program (with your own character definitions set up in the Basic program called 'chars':)

```
10 *FX20,1
20 ?&36C=&9
30 PAGE=&1900
40 CHAIN "chars"
```

This will place your 64 definitions in the locations &C00 to &CFF (as it usually does) and &900 to &9FF (instead of &1900 to &19FF). Change the value of PAGE in line 30 to &E00 if you are using cassettes.

Tested on Basic I & II and O.S. 1.2

TESTING OUT YOUR MICRO (Part 3) – RANDOM ACCESS MEMORY

by Hugh D. Brown-Smith

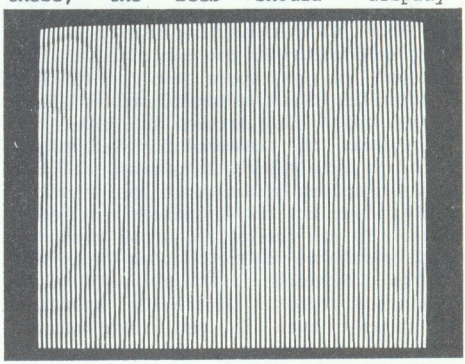
Does your micro suffer from Amnesia? This month's article on testing out your micro concentrates on checking the Beeb's Random Access Memory (RAM) to see if there is any loss of memory.

In order to test the micro's memory (RAM), two checks must be performed, firstly that each bit of memory can be set (to 1), and secondly that each bit can be cleared (to Ø). Writing a program to test memory is not as easy as it sounds, since the program must occupy part of the memory to be tested, and that part of the memory at least must be working correctly for the program to run. Additionally, the O.S. makes continual use of memory page zero, adding a further complication. This can be overcome by dispensing with the O.S. and keeping the test program as short as possible. In this program, therefore, all interrupts have been turned off and a suitable routine to detect a key press included since the normal keyboard interrupt is no longer effective.

Before trying the program, it is important to save it to cassette or disc after you have first typed it in, as running the program leaves it in a corrupted state and the micro partially disabled. This is because the program tests the whole of memory, including zero page and the machine work space. For this reason, after the program has been run, the computer will need to be turned off in order to reset it.

In the test program, memory is tested in two sections, each of 20K in size. This value was chosen so that the memory being checked could be displayed as a complete Mode 2 screen. Rather than checking and reporting on every bit of memory, the program represents memory visually on the screen in a way which you can then check for yourself. Because of this, program length can be kept to a minimum. The Mode 2 screen has the lowest resolution of the 20K modes, and therefore the displays are easy to see, even on a very low resolution television or monitor.

When run the program asks whether you want to check the upper or lower sections of RAM. On selecting one of these, the Beeb should display



parallel, vertical black and white lines. The white lines show locations set to 1, the black lines those set to Ø, while pressing any key will reverse the colours of the black and white stripes, thus ensuring that each memory bit can be tested for both 'set' and 'clear'. To test the other half of memory, perform a cold start, re-run the program and select the other section of memory.

The pattern of bit setting and clearing has been chosen to give the simple black and white striped display referred to above. This has the advantage that any bit that is incorrectly set will appear either as a ridge in one of the lines, or as a spot of a different colour. Checking memory in this way ensures that every bit is tested for both 'set' and 'clear' (set to a '1' state, clear to a 'Ø' state), and that the screen display provides a ready check of memory status.

Because of the quite complex way in which the Beeb's memory is related to the physical memory chips, it is not possible with this program to identify specific memory chips in the event of some error being apparent, only to identify memory, as a whole, as being at fault.

```
  10 REM Program RAM TEST
  20 REM Version B1.4
  30 REM Author Hugh D.Brown-Smith
  40 REM BEEBUG May 1984
  50 REM Program subject to Copyright
  60 :
  70 MODE7
  80 PRINT"HI OR LO MEMORY BLOCK (H OR
L)"
  90 *FX15,1
 100 REPEAT A$=GET$:UNTIL A$="H" OR A$
="L"
 110 MODE2
 120 VDU19,15,7,0;0
 130 FORZ=1TO14
 140 VDU19,Z,0;0;
 150 NEXT
 160 IFA$="H":M%=&2E00:MEM%=&3000:GOTO
180
 170 IFA$="L":M%=&7E00:MEM%=0:VDU23;12
;0;0;0;
```

```
180 VDU23,1,0;0;0;0;
190 FORX=0TO1
200 P%=M%
210 [OPT0
220 SEI
230 LDA #&00
240 .clearmem0
250 LDX #&50
260 .clearmem1
270 LDY #&00
280 .clearmem2
290 STA MEM%,Y
300 DEY
310 BNE clearmem2
320 INC clearmem2+2
330 DEX
340 BNE clearmem1
350 .crosshatch
360 LDA #&55
370 LDX #&50
380 .crsh0
390 LDY #&00
400 .crsh1
410 STA MEM%,Y
420 DEY
430 BNE crsh1
440 INC crsh1+2
450 DEX
460 BNE crsh0
470 .keyscan
480 LDX #15
490 STX &FE41
500 LDX #1
510 STX &FE4D
520 LDX #9
530 STX &FE41
540 LDA #1
550 BIT &FE4D
560 BEQ keyscan
570 .swaplines
580 LDA #&AA
590 LDX #&50
600 .swap1
610 LDY #&00
620 .swap2
630 STA MEM%,Y
640 DEY
650 BNE swap2
660 INC swap2+2
670 DEX
680 BNE swap1
690 .keyscan2
700 LDX #15
710 STX &FE41
720 LDA #1
730 STX &FE4D
740 LDX #15
750 STX &FE41
760 LDA #1
770 BIT &FE4D
780 BEQ keyscan2
790 LDA #MEM% DIV 256
800 STA crsh1+2
810 STA swap2+2
820 JMP crosshatch
830 ]
840 NEXT
850 CALL M%
```

POINTS ARISING

TESTING OUT YOUR MICRO - SIDEWAYS ROMS

The program called 'ROM' listed in BEEBUG Vol.2 No.9 fails to read socket zero. The program can be easily amended to do this by changing line 1200 to read:

```
1200 BPL mainloop
```

instead of the BNE instruction listed at this line number.

BACH'S CANTATA NO.147

A program which played part of this music was also included in BEEBUG Vol.2 No.9. Laurie Crawford has sugested that a slight musical improvement can be achieved by changing line 1040 to read as follows:

```
1040 DATA &101,137,4,&102,101,12,1,145,4,1,129,4,&101,125,4,&102,89,12,1,129,4,1,
137,4
```

You may like to try this variation and see what you think.

MULTIPLE DISC CATALOGUES

Despite extensive testing, two errors occurred in the program CAT2 printed in BEEBUG Vol.2 No.9. This version is intended for use with 40 track drives and the number 79 in lines 230 and 240 should be changed to 39 in each case. This is a fairly obvious change, but it is also an important one.

SPECIAL SYMBOLS FROM WORDWISE

This Wordwise hint for a 'scissors' symbol, in BEEBUG Vol.2 No.9, was inaccurate as given. The correct sequence of control codes should read:

```
OC27,75,8,0,108,170,108,40,16,40,68,130
```

and not as printed.

VASM – A 6502 DISC-BASED ASSEMBLER

Reviewed by David A. Fell

VASM is a powerful 6502 assembler in EPROM now available from Vida Rebus. David Fell looks at this new product and explains how it works in this in-depth review.

When you purchase the BBC micro, you receive a 6502 assembler built into the Basic language ROM. This can cope with the assembly requirements of most beginners, but when developing lengthy and complex machine code programs, the Basic assembler can often prove to be a limiting factor. Vida Rebus have now produced an assembler in EPROM that resides in one of the sideways ROM sockets, and assembles directly from disc, to provide a powerful development system for any disc-based BBC micro.

The Vida Rebus VASM assembler comes in the form of an 8K EPROM, and it is specially designed for use with a disc system. It is accompanied by an attractive A5-sized manual that details the syntax and usage of all of the features of VASM. The provision of a number of very powerful features makes this assembler an essential tool for the serious machine code programmer on the BBC micro.

One of the most striking features of VASM (which is common to most powerful assemblers), is the fact that it takes its source code directly from a text file on disc, and not from memory. The BBC Basic assembler takes its source code directly from memory, embedded within a Basic program. It is very difficult to assemble large amounts of source code with the BBC assembler as memory must be shared by the source and assembled code. Because VASM takes its source code from disc, and sends its output directly back to disc, VASM is limited only by the size of your disc drives, and not by the size of memory. This is a major advantage over the Basic assembler.

Because VASM needs its source code in the form of a text file, you will also need to have your own text editor to create and edit source code files. VASM ignores any control characters (e.g. Tab) that it does not recognise, and so any standard text editor may be used. Both View and Wordwise have been found to work, but you could also write your own simple editor if you wanted to.

Once you have created your source code, as a text file on disc, you can use VASM to assemble the source code, producing an intermediate text file in so called 'INTEL hex' format. This consists of a series of ASCII characters which represent the hexadecimal equivalent of the machine code assembled, plus some other information. 'INTEL hex' is one of the common ways in which assemblers produce their output, and is widely recognised as being an industry standard.

Once the intermediate file has been generated, it can be loaded into any memory area using the *LOCATE command provided by VASM. The resulting machine code program in memory can be saved (using *SAVE) on disc, and run at any time as a machine code program. This method also allows programs to be assembled in one area of memory, but eventually run in another. This facility is very useful when developing sideways ROMs, or other machine code programs that reside in 'dangerous' areas of memory.

Intermediate code files in 'INTEL hex' format also incorporate automatic error detection through the use of checksums, to provide an additional level of security. A further advantage of this standard is that it opens up the possibility of using the Beeb as a development system for any 6502 based micro, with the 'INTEL hex' file being transmitted through the RS423 port to the target machine.

Although there are some superficial differences between VASM and the Beeb's Basic assembler, these are few and in many cases VASM will accept the

conventions of both the Basic assembler and those of the more common industry standards. Indeed, anyone who has used such a 'standard' 6502 assembler, and moved to the Beeb, may find VASM more familiar than the built in assembler. All the requirements of assembly are controlled through VASM, and there is no need to embed the source code in a Basic program or use any of the other conventions of Basic imposed by the Beeb's built-in assembler.

One of the major criticisms of the Beeb's assembler is the difficulty which is encountered when attempting to introduce data into the program. To this end, VASM provides a comprehensive set of directives that allow most data types to be readily incorporated into the final machine code program.

Large programs can be divided into several modules to be combined at the time of assembly, and the provision of local labels is useful here. VASM does not, however, support full macro assembly features. Conditional assembly of parts of a program is also possible. By careful use of these features it is possible to write a large source file, containing all possible routines for a given application, but to assemble only specific portions in response to prompts given at assembly time.

To conclude, VASM is a very comprehensive assembler, and has few missing features (perhaps the only one is the inability to specify a binary constant). I would thus recommend it to anyone considering serious machine code development on the Beeb, along with one of the numerous machine code monitors available on the market (for example, BEEBUGSOFT's EXMON).

Vida Rebus (which means 'The shape of things to come') is a software house specialising in systems control software, and they are currently preparing cross assemblers for 6809, Z-80 and 68000 processors, all to the same common format, with the output being presented in an intermediate form of INTEL hex files. For the duration of May, a special offer available only to members of BEEBUG, you can obtain a copy of VASM at the discounted price of £35.00 (inc.) by writing to Vida Rebus, and quoting your membership number. The normal price is £40.25 (inc.) Vida Rebus are at:

VIDA REBUS
P.O. BOX 256
WATFORD
HERTS.
WD1 8HY

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SOME WORDWISE HINTS

We have put together several short hints concerned with Wordwise from various contributors. The first comes from Mr T.Baker who has told us that it is possible to have 15 function keys available with Wordwise by using the five cursor keys (up, down, left, right and Copy). They are programmed in exactly the same manner using the command *KEYxx where xx is a number from 11 to 15, and accessed by pressing them in conjunction with Control and Shift. The function key numbers associated with the cursor keys are as follows:

11	Copy	14	Down
12	Left	15	Up
13	Right		

Secondly, Mr.J.Brett has pointed out an inaccuracy with the word counter, in that it fails to update the word count at the top of the screen if the word is followed by a Return. If an accurate value of the number of words in a piece of text is required, then the solution it to put a space in after the word, and before the Return.

Finally, Mr.M.Robson has sent in the following very useful key definition which finds and removes markers in a piece of text. It is accessed by holding the Control and Shift keys down together, and pressing function key 0.

```
*KEY0~|!|O~|!|O|!$|!#|A|!|O|!$|!#|A|!|O|A|!$~|A
```

It can be entered in immediate mode or used in a program which initialises the function keys.

Tested on Basic I & II and O.S. 1·2

DOMINOES (32K)

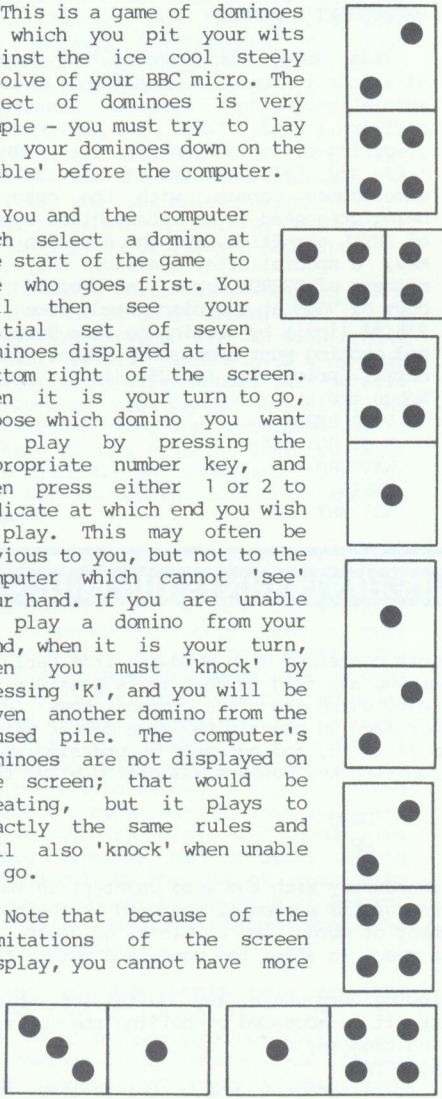
by C. C. Radcliffe

The program DOMINO provides an excellent computer version of the popular table game of dominoes for you to play against your BBC micro. The graphics used in this program are a real delight and show how well traditional games like dominoes can be represented on the screen.

This is a game of dominoes in which you pit your wits against the ice cool steely resolve of your BBC micro. The object of dominoes is very simple - you must try to lay all your dominoes down on the 'table' before the computer.

You and the computer each select a domino at the start of the game to see who goes first. You will then see your initial set of seven dominoes displayed at the bottom right of the screen. When it is your turn to go, choose which domino you want to play by pressing the appropriate number key, and then press either 1 or 2 to indicate at which end you wish to play. This may often be obvious to you, but not to the computer which cannot 'see' your hand. If you are unable to play a domino from your hand, when it is your turn, then you must 'knock' by pressing 'K', and you will be given another domino from the unused pile. The computer's dominoes are not displayed on the screen; that would be cheating, but it plays to exactly the same rules and will also 'knock' when unable to go.

Note that because of the limitations of the screen display, you cannot have more than 9 dominoes in your hand at any one time. At the end of the game, the overall score is displayed on the screen, together with any dominoes remaining in either hand.



This is a very nicely presented game using Mode 1 graphics which is really pleasing to play.

NOTES FOR DISC USERS

Because this program runs in Mode 1, there is insufficient room on a normal disc system (with PAGE set to &1900) for both program and data. Either set PAGE to &1200 before loading the program or use a suitable move-down routine such as the following:

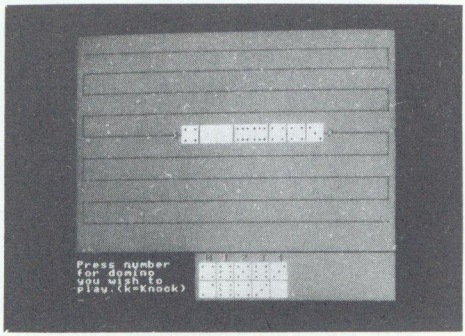
```
10MODE7
20*L.DOMINO 2000
30*T.
40*K.0 F.A%=0TO&2000S.4:A%!&E00=A%!&
2000:N.|M PAGE=&E00|MOLD|MRUN|M
50*FX138,0,128
60END
```

This can be saved as B.DOMINO. To run the Domino program type

```
CHAIN "B.DOMINO"
```

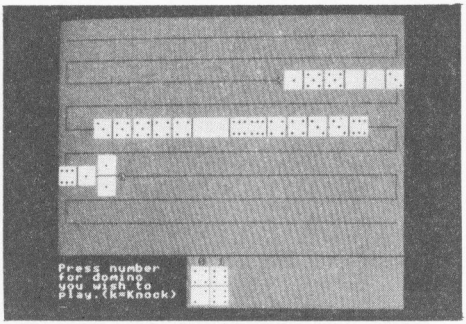
which will automatically load and run the Domino program provided this is SAVEd as 'DOMINO'.

```
 10 REM PROGRAM DOMINOES
 20  REM AUTHOR  C.C.Radcliffe
 30  REM VERSION B0.1
 40 REM BEEBUG  MAY 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 ON ERROR GOTO 430
110 :
120 CLEAR
130 *FX11,0
140 MODE1
```




```
  150 VDU23,1,0;0;0;0;
  160 PROCinit
  170 PROCshuffle
  180 PROCstart
  190 REPEAT
  200 WT=FALSE
  210 PROCshuffle:S%=0
  220 GCOL0,129:CLG
  230 VDU28,0,31,14,26,12,18,0,0,25,4,48
0;188;25,5,1279;188;
  240 FOR I%=1 TO 9
  250 MOVE29,1023-I%*80:PLOT25,1221,0
  260 IF (I%MOD2 AND I%<9) PLOT25,0,-80
 ELSE IF I%<9 PLOT0,-1221,0:PLOT25,0,-80
  270 NEXT
  280 FORI%=0TO1
  290 PROCpick(I%,7):end%(I%)=-1:endl%(I
%)=3
  300 NEXT
  310 WT=TRUE
  320 REPEAT
  330 IFBEEB%PROCBEEB ELSEPROCdisphand(1
,0):PROCplayer
  340 UNTILtot%(0)=0 ORtot%(1)=0 OR(bk%A
NDpk%ANDS%>27)
  350 I%=0:W%=0:REPEAT
  360 IFtot%(I%)=0 PROCwin(I%):W%=1
  370 I%=I%+1
  380 UNTIL W% OR I%=2
  390 IFW%=0 W%=FNwin:PROCwin(W%)
  400 UNTILFALSE
  410 END
  420 :
  430 ON ERROR OFF:MODE 7
  440 *FX12
  450 IF ERR<>17 REPORT:PRINT" at line "
;ERL
  460 END
  470 :
 1000 DEFPROCinit
 1010 DIMdom$(27),hand$(1,9),tot%(1),end
%(1),endl%(1),endx%(1),win%(2),th%(1)
 1020 A%=RND(TIME)
 1030 FORI%=0TO1
 1040 tot%(I%)=0
 1050 FORK%=0TO9
 1060 hand$(I%,K%)="  "
 1070 NEXT:NEXT
 1080 FOR I%=0 TO2:win%(I%)=0:NEXT
 1090 VDU23,255,0;0;32,112,32,0
 1100 VDU19,1,2,0,0,0:VDU19,2,1,0,0,0
 1110 bk%=0:pk%=0
 1120 ENDPROC
 1130 :
 1140 DEFPROCshuffle
 1150 RESTORE1280
 1160 LOCALN%,I%
 1170 FORN%=0TO27
 1180 dom$(N%)="  "
 1190 NEXT
 1200 FORN%=0TO27
 1210 REPEAT
 1220 I%=RND(28)-1
 1230 UNTILdom$(I%)="  "
 1240 READdom$(I%)
 1250 NEXT
 1260 ENDPROC
 1270 :
 1280 DATA00,01,02,03,04,05,06,11,12,13
,14,15,16,22,23,24,25,26,33,34,35,36,44
,45,46,55,56,66
 1290 :
 1300 DEFPROCstart
 1310 LOCALB%,bd%,P%,N%,pd%,s$,t$
 1320 CLS
 1330 PRINTTAB(16,1)"DOMINOES"
 1340 PRINT'''"Welcome to BEEBDOM."'''"Pl
ease type in your name:"'
 1350 REPEAT:INPUTTAB(0,8)n$:L%=LEN(n$)
:IFL%>10PRINTTAB(0,8)SPC(L%)
 1360 UNTILL%<11
 1370 PRINT'''"Hello ";n$;"."'
 1380 PRINT"Please enter a number (1-28
) to select"'''"a domino, the BEEB will
also choose -"'''"the highest number of
dots starts!"
 1390 REPEAT:INPUTTAB(0,19)P%:P%=P%-1:U
NTILP%<28ANDP%>=0
 1400 pd%=FNtotdot(P%)
 1410 REPEAT
 1420 B%=RND(28)-1
 1430 bd%=FNtotdot(B%):s$="BEEB"
 1440 UNTILC%<>B%ANDpd%<>bd%
 1450 PROCdomino(320,320,TRUE,FNldot(do
m$(P%)),FNrdot(dom$(P%))):PROCdomino(96
0,320,TRUE,FNldot(dom$(B%)),FNrdot(dom$
(B%)))
 1460 PRINTTAB(0,26)SPC(10-LEN(n$)DIV2)
;n$;SPC(17-LEN(n$)DIV2);s$
 1470 IFbd%>pd%BEEB%=TRUEELSEBEEB%=FALS
E:s$=n$
 1480 PRINT's$;" starts!"'''"Press any k
ey to continue."
 1490 P%=GET
```

```
 1500 ENDPROC
 1510 :
 1520 DEFFNtotdot(N%)=FNldot(dom$(N%))+
FNrdot(dom$(N%))
 1530 :
 1540 DEFFNldot(D$)=VAL(LEFT$(D$,1))
 1550 :
 1560 DEFFNrdot(D$)=VAL(RIGHT$(D$,1))
 1570 :
 1580 DEFPROCdomino(X%,Y%,O%,L%,R%)
 1590 LOCALN%,D%
 1600 GCOL0,2:IFO%MOVEX%-30,Y%-6:PLOT0,
0,12:PLOT81,60,-12:PLOT81,0,12ELSEMOVEX
%-6,Y%-30:PLOT0,12,0:PLOT81,-12,60:PLOT
81,12,0
 1610 FORN%=-1TO1STEP2
 1620 GCOL0,3:IFO%MOVEX%+N%*30,Y%+N%*66
ELSEMOVEX%+N%*66,Y%+N%*30
 1630 PLOT0,-60*N%,0:PLOT81,60*N%,-60*N
%:PLOT81,-60*N%,0
 1640 NEXT
 1650 O%=ABS(O%)
 1660 FORN%=-1TO1STEP2
 1670 H%=O%*(X%-8)-(O%-1)*((X%-8)+N%*36)
 1680 V%=O%*((Y%+18)+N%*36)-(O%-1)*(Y%+
18)
 1690 IFN%=-1D%=L%ELSED%=R%
 1700 VDU5:PROCdot(H%,V%,D%)
 1710 NEXT
 1720 VDU4
 1730 ENDPROC
 1740 :
 1750 DEFPROCdot(X%,Y%,T%)
 1760 LOCALN%,M%,D%,A%,H%,V%
 1770 M%=T%MOD2:D%=T%DIV2
 1780 GCOL4,0:MOVEX%,Y%
 1790 IFM%PRINTCHR$255;
 1800 IFD%=0ENDPROC
 1810 IFO%RESTORE1940ELSERESTORE1950
 1820 N%=0
 1830 REPEAT
 1840 READA%:N%=N%+1
 1850 UNTILN%=D%
 1860 RESTORE1960
 1870 FORN%=1TO8
 1880 READH%,V%,Q%
 1890 MOVEX%,Y%:PLOT0,H%,V%
 1900 IFQ%ANDA%PRINTCHR$255
 1910 NEXT
 1920 ENDPROC
 1930 :
 1940 DATA&44,&55,&DD
 1950 DATA &11,&55,&77
 1960 DATA -20,20,1,0,20,2,20,20,4,20,0
,8,20,-20,&10,0,-20,&20,-20,-20,&40,-20
,0,&80
 1970 :
 1980 DEFPROCpick(P%,D%)
 1990 LOCALN%
 2000 FORN%=1TOD%
```




```
 2010 IFS%>27D%=N%ELSEhand$(P%,tot%(P%)
)=dom$(S%):S%=S%+1:PROCsound
 2020 NEXT
 2030 ENDPROC
 2040 :
 2050 DEFPROCdisphand(P%,T%)
 2060 VDU24,500;0+T%*184;1279;184+T%*18
4;16
 2070 LOCALN%,X%
 2080 IF tot%(P%)=0 VDU26:ENDPROC
 2090 FORN%=0 TOtot%(P%)-1
 2100 X%=532+(N%)*72
 2110 PROCdomino(X%,80+T%*184,TRUE,FNld
ot(hand$(P%,N%)),FNrdot(hand$(P%,N%)))
 2120 MOVEX%-10,180+T%*184
 2130 VDU5
 2140 IF T%=0 PRINT;N%
 2150 NEXT:IF T%=0 VDU4,24,0;196;1279;1
023; ELSE VDU4,26
 2160 ENDPROC
 2170 :
 2180 DEFPROCBEEB
 2190 LOCALE%,N%,D%,O%,L%,R%,P%:BEEB%=F
ALSE:bk%=0
 2200 IFend%(0)=-1X%=640:Y%=623:D%=FNls
t:L%=FNldot(hand$(0,D%)):R%=FNrdot(hand
$(0,D%)):end%(0)=L%:end%(1)=R%:O%=(L%=R
%):P%=1:FORE%=0 TO1:endx%(E%)=640+FNcen
tre:PROCendno(E%):NEXTELSEPROCplay:PROC
endno(E%)
 2210 IFP%PROCdomino(X%,Y%,O%,L%,R%):PR
OCsort(0,D%)ELSEPROCpick(0,1):bk%=1
 2220 ENDPROC
 2230 :
 2240 DEFFNvert(D$)
 2250 IFFNldot(D$)=FNrdot(D$)=TRUEELSE=
FALSE
 2260 :
 2270 DEFPROCplayer
 2280 LOCALE%,B%,F%,N%,P%,D%,X%,Y%,O%,I
%,J%,L%,R%:BEEB%=TRUE:pk%=0
 2290 PRINT'"Press number"'"for domino"
'"you wish to"'"play.(k=Knock)"
 2300 REPEAT:D%=GET-48:UNTIL(D%>=0 ANDD
%<tot%(1))ORD%=59 ORD%=27:IFD%>tot%(1)P
ROCpick(1,1):pk%=1:ENDPROC
```

```
 2310 I%=FNldot(hand$(1,D%)):J%=FNrdot(
hand$(1,D%)):O%=FNvert(hand$(1,D%))
 2320 IFend%(0)=-1X%=640:Y%=623:L%=I%:R
%=J%:end%(0)=L%:end%(1)=R%:P%=TRUE:FORE
%=0 TO1:endx%(E%)=640+FNcentre:PROCendn
o(E%):NEXT:GOTO2370
 2330 IFend%(0)<>-1CLS:PRINT'"Press num
ber"'"for end"'"you wish to"'"play.":RE
PEAT:E%=GET-49:UNTILE%=0ORE%=1
 2340 IFI%=end%(E%)B%=0:F%=1
 2350 IFJ%=end%(E%)B%=1:F%=1
 2360 IFF%PROCendno(E%):PROCarrange(B%)
:PROCsetup(E%):PROCendno(E%)
 2370 IFP%PROCdomino(X%,Y%,O%,L%,R%):PR
OCsort(1,D%)ELSEPROCpick(1,1):pk%=1
 2380 IF WT SOUND0,-15,100,6
 2390 IF WT G=INKEY(100)
 2400 ENDPROC
 2410 :
 2420 DEFPROCplay
 2430 LOCALI%,i%,J%,j,K%,B%,T%,M%:E%=-1
 2440 FORM%=0TOtot%(0)-1
 2450 i%=FNldot(hand$(0,M%)):j%=FNrdot(
hand$(0,M%))
 2460 FORK%=0TO1
 2470 IFi%=end%(K%)ANDT%<=i%+j%PROCsave
:B%=0
 2480 IFj%=end%(K%)ANDT%<=i%+j%PROCsave
:B%=1
 2490 NEXT:NEXT
 2500 O%=FNvert(hand$(0,D%))
 2510 IFE%>-1 PROCendno(E%):PROCarrange
(B%):PROCsetup(E%)
 2520 ENDPROC
 2530 :
 2540 DEFPROCsave:D%=M%:T%=i%+j%:I%=i%:
J%=j%:E%=K%:ENDPROC
 2550 :
 2560 DEFPROCsetup(E%)
 2570 LOCALC%,W%
 2580 P%=1:C%=FNcentre:X%=endx%(E%)+C%:
W%=endx%(E%)+2*C%+8*SGN(C%)
 2590 IFW%<70RW%>1271endl%(E%)=endl%(E%
)+SGN(C%):X%=C%-1279*(E%=0):endx%(E%)=W
%-endx%(E%)-1279*(E%=0)ELSEendx%(E%)=W%
 2600 Y%=1103-endl%(E%)*160
 2610 ENDPROC
 2620 :
 2630 DEFFNcentre=-((O%=0)*70+(O%=TRUE)
*34)*((E%=0)-(E%=1))
 2640 :
 2650 DEFPROCsort(P%,D%)
 2660 LOCALN%,T%
 2670 IFtot%(P%)<10T%=tot%(P%)-1ELSET%=
8:hand$(P%,9)="  "
 2680 FORN%=D%TOT%
 2690 hand$(P%,N%)=hand$(P%,N%+1)
 2700 NEXT
 2710 tot%(P%)=tot%(P%)-1
 2720 ENDPROC
 2730 :
 2740 DEFPROCarrange(F%)
 2750 IFF%=0end%(E%)=J%ELSEend%(E%)=I%
 2760 IFF%=E%R%=I%:L%=J%ELSEL%=I%:R%=J%
 2770 ENDPROC
 2780 :
 2790 DEFFNlst
 2800 LOCALN%,T%,H%,I%
 2810 FORN%=1TO7
 2820 T%=FNtotdot(N%-1):IFT%>H%H%=T%:I%
=N%-1
 2830 NEXT:=I%
 2840 :
 2850 DEF PROCendno(E%)
 2860 IFE%=-1 ENDPROC ELSE VDU5:GCOL4,0
:MOVEendx%(E%)+30*(E%=0),1115-endl%(E%)
*160:PRINT;E%+1:VDU4
 2870 ENDPROC
 2880 :
 2890 DEF FNwin
 2900 LOCAL N%,I%
 2910 FOR N%=0 TO 1
 2920 th%(N%)=0
 2930 FOR I%=0 TO tot%(N%)-1
 2940 th%(N%)=th%(N%)+FNldot(hand$(N%,I
%))+FNrdot(hand$(N%,I%))
 2950 NEXT:NEXT
 2960 IF th%(0)>th%(1) =1
 2970 IF th%(1)>th%(0) =0 ELSE =2
 2980 :
 2990 DEF PROCwin(W%)
 3000 VDU26,12
 3010 LOCALN%
 3020 IF W%=0:BEEB%=TRUE:W$="BEEB"
 3030 IF W%=1:BEEB%=FALSE:W$=n$
 3040 IF W%<2 PRINT'"The WINNER is ";W$
 ELSEPRINT'"Game Drawn!"
 3050 PRINT'"State of play:"
 3060 win%(W%)=win%(W%)+1
 3070 PRINT''SPC(6);"BEEB";SPC(10-LEN(n
$)DIV2+LEN(n$)MOD2);n$;SPC(10-LEN(n$)DI
V2);"Drawn"''SPC(2);
 3080 FOR N%=0 TO 2
 3090 PRINT;SPC(7);win%(N%);
 3100 NEXT
 3110 PRINT'"Unplayed hands: "''''n$''
'''"BEEB"'''
 3120 PRINT'"Press any key to continue
."
 3130 FOR N%=1 TO2:PROCdisphand(N%-1,N%
):tot%(N%-1)=0:NEXT
 3140 k=GET
 3150 ENDPROC
 3160 :
 3170 DEFPROCsound
 3180 IF WT SOUND1,-15,100,4:SOUND1,-15
,10,6
 3190 IF WT G=INKEY(100)
 3200 IF tot%(P%)<9 tot%(P%)=tot%(P%)+1
 3210 ENDPROC
```

Tested on Basic I & II and O.S. 1.2

DETONATOR DAN

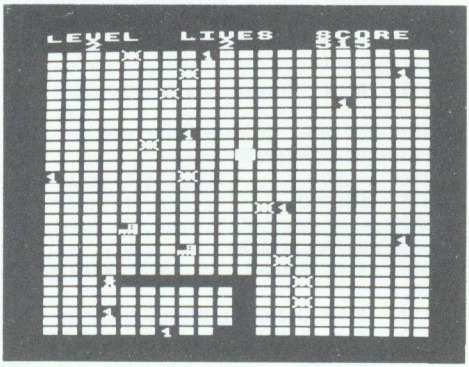
by K.Miles

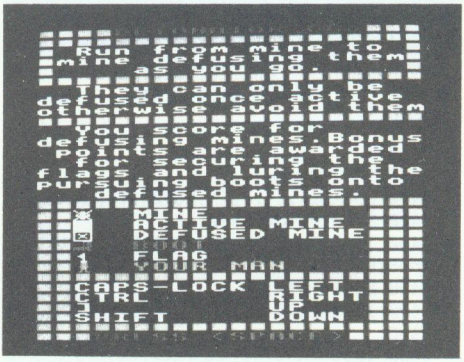
Detonator Dan is a superb fast moving action game that we have been saving just for this second anniversary issue of BEEBUG. The program is written in Basic and runs in Mode 2 to provide a very colourful display, and with excellent sound effects.

The games starts by displaying a number of unexploded mines on the screen, together with a number of flags. The object of the game is to defuse each of the mines before they explode. Once a mine becomes live it starts counting from 10 and 'ticks' down to 0. When it reaches 0, it explodes and you lose a life. You must try to reach the mine and defuse it (by moving into it) before this happens.

You must also watch out for the big army boots which will come marching after you, and will squash you if they can. As the game progresses the mines tick faster and there are more boots on the screen to avoid. With all this high speed action you certainly need a cool head and quick thinking.

You score points by defusing mines as quickly as possible, by luring the boots into defused mines to destroy them and by picking up flags. You also lose points every time you run onto a blue background square, so you need to work out your routes carefully to minimise the loss. The boots also fill in any missing 'blue' squares as they move, making life even more difficult.





You control your own movement on the screen (your position is indicated by the display of a small 'man') by the use of 'CAPS LOCK' and 'CTRL' for left and right, and ']' and 'SHIFT' for up and down. Keep an eye on your score as you play the game for if your score drops to zero you have no energy to continue and the game is lost. This is why choosing an economic path is so important. A good score, by the way, is one in excess of 50,000.

Full instructions are included in the program and there is a high score table as well. Good luck!

Users with disc systems will either need to compact the program before use (removing unnecessary spaces, REM statements etc), change the value of PAGE (to &1200) or use a move down routine (see the Dominoes game elsewhere in this issue).

```
10 REM PROGRAM Detonator Dan
20 REM AUTHOR  K.Miles
30 REM VERSION B0.5
40 REM BEEBUG  MAY 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
```

```
  60 :
 100 ON ERROR GOTO 490
 110 MODE2
 120 ENVELOPE3,129,2,4,6,28,14,7,0,0,0
,-80,80,80
 130 ENVELOPE2,1,4,-4,4,10,20,10,127,0
,0,-5,126,126
 140 ENVELOPE1,1,0,0,0,0,0,0,0,0,0,-1,
126,0
 150 VDU23,1,0;0;0;0;
 160 DIMX%(19),Y%(19),S1%(5),S2%(5),A$
(10,1)
 170 PROCcharacters
 180 PROCinit
 190 PROCtitles
 200 CLS:LV%=3:CO%=10:sc%=15:sp%=8
 210 SC%=1000:LVL%=1
 220 RA%=1:AL%=4:SP%=sp%
 230 IFAL%>18 AL%=18
 240 IFRA%>4 RA%=4
 250 IFSP%<2 SP%=2
 260 IFCO%<2 CO%=2
 270 R%=0:C%=0:E%=0:SX%=10:SY%=29:XS%=
SX%:YS%=SY%:FL%=-1:GO%=1:DL%=0:BT%=0
 280 PROCboard
 290 PROCmine
 300 PROCboot
 310 PROCflag
 320 REPEAT:PROCtrigger:PROCman:PROCbo
ot2
 330 IFSC%<=0 SC%=0:PROCdead
 340 COLOUR3:PRINTTAB(14,1);SC%;" ";:U
NTIL C%=AL%+1 OR E%=1 OR SC%<=0
 350 IFE%=1OR SC%<=0 THEN390
 360 LVL%=LVL%+1
 370 IFC%=AL%+1AND AL%=18:CO%=CO%-2:sc
%=sc%+5:sp%=sp%-2:LV%=LV%+1:GOTO220
 380 RA%=RA%+1:AL%=AL%+5:SP%=SP%-2:GOT
O230
 390 IFLV%>0THEN250
 400 PROCexplode
 410 FOR wait=1TO5000:NEXT
 420 PROChiscore
 430 COLOUR8:PRINTTAB(4,28)"DETONATOR
DAN":COLOUR3:PRINTTAB(0,30)"ANOTHER GAM
E?";
 440 *FX21,1
 450 A$=GET$:IFA$="Y"OR A$="y" GOTO190
 460 IFA$="N"OR A$="n" MODE7:GOTO 520
 470 GOTO450
 480 :
 490 ON ERROR OFF
 500 MODE 7:IF ERR=17 GOTO 520
 510 REPORT:PRINT" at line ";ERL
 520 ?&02=&20: REM RETURN TO UP/CASE
 530 END
 540 :
1000 DEFPROCboard
1010 COLOUR3:PRINTTAB(0,0)"LEVEL"TAB(7
,0)"LIVES"TAB(14,0)"SCORE"
1020 PRINTTAB(2,1);LVL%;TAB(9,1);LV%;T
AB(14,1);SC%
1030 COLOUR4:FORI%=1TO28:PRINTSTRING$(
20,CHR$224);:NEXT
1040 ENDPROC
1050 :
1060 DEFPROCcharacters
1070 VDU23,224,0,126,126,126,126,126,1
26,0,23,225,15,11,15,11,255,255,255,171
1080 VDU23,226,129,90,60,90,126,60,90,
129,23,227,28,28,28,62,28,28,20,54
1090 VDU23,228,0,8,24,56,24,8,8,28,23,
229,255,129,165,153,153,165,129,255,23,
230;146,84;198;84,146
1100 ENDPROC
1110 :
1120 DEFPROCinit
1130 FORI=1TO10:A$(I,0)=STR$(I*1000):A
$(I,1)="BEEBUG":NEXT
1140 ENDPROC
1150 :
1160 DEFPROCmine
1170 COLOUR6:FORI%=0TO AL%:X%(I%)=1+RN
D(18):Y%(I%)=1+RND(27):VDU31,X%(I%),Y%(
I%),226:NEXT
1180 ENDPROC
1190 :
1200 DEFPROCflag
1210 FORI%=0TO AL%
1220 X1%=RND(20)-1:Y1%=RND(28)+1
1230 fg%=FNPT(X1%,Y1%)
1240 IFfg%=4 COLOUR3:VDU31,X1%,Y1%,228
ELSE GOTO1220
1250 NEXT
1260 ENDPROC
1270 :
1280 DEFPROCboot
1290 FORI%=1TORA%
1300 S1%(I%)=RND(20)-1:S2%(I%)=RND(28)
+1
1310 bt%=FNPT(S1%(I%),S2%(I%))
1320 IFbt%=4 COLOUR1:VDU31,S1%(I%),S2%
(I%),225 ELSE GOTO1300
1330 NEXT
1340 ENDPROC
1350 :
1360 DEFPROCtrigger
1370 IFFL%>-1 PROCmine2:GOTO1420
1380 IFRND(1)<.95THEN1420
1390 R%=RND(AL%+1)-1
1400 IFY%(R%)=-32THEN1390
1410 FL%=10
1420 ENDPROC
1430 :
1440 DEFPROCmine2
1450 IFFL%=0 PROCexplode:E%=1:LV%=LV%-
1:GOTO1500
1460 DL%=DL%+1:IFDL%=CO% DL%=0 ELSEGOT
O1500
1470 FL%=FL%-1
```

```
 1480 COLOUR5:COLOUR135:VDU31,X%(R%),Y%
(R%),48+FL%
 1490 COLOUR128
 1500 ENDPROC
 1510 :
 1520 DEFPROCman
 1530 VDU31,SX%,SY%,32
 1540 XS%=SX%:YS%=SY%
 1550 *FX21,0
 1560 *FX21,5
 1570 IFINKEY(-2)AND SX%<19 SX%=SX%+1:G
OTO1620
 1580 IFINKEY(-65)AND SX%>0 SX%=SX%-1:G
OTO1620
 1590 IFINKEY(-89)AND SY%>2 SY%=SY%-1:G
OTO1620
 1600 IFINKEY(-1)AND SY%<29 SY%=SY%+1:G
OTO1620
 1610 GOTO1690
 1620 SOUND&11,2,50,1
 1630 mn%=FNPT(SX%,SY%):mno%=FNpt(SX%,S
Y%)
 1640 IFmn%=4 SC%=SC%-sc%:GOTO1690
 1650 IFmn%=14 SX%=XS%:SY%=YS%:GOTO1690
 1660 IFmno%=5 PROCdefuse:SX%=XS%:SY%=Y
S%:GOTO1690
 1670 IFmn%=10R mn%=6 PROCdead:GOTO1700
 1680 IFmn%=3 PROCbonus
 1690 COLOUR2:VDU31,SX%,SY%,227
 1700 ENDPROC
 1710 :
 1720 DEFPROCbonus
 1730 SOUND&13,3,50,10
 1740 SC%=SC%+100
 1750 ENDPROC
 1760 :
 1770 DEFPROCdefuse
 1780 SOUND0,1,5,1
 1790 FORJ%=0TO AL%
 1800 IFSX%<>X%(J%)ORSY%<>Y%(J%)THEN1830
 1810 SC%=SC%+(100*LVL%):C%=C%+1
 1820 Y%(J%)=-32
 1830 NEXT
 1840 COLOUR14:VDU31,SX%,SY%,229
 1850 FL%=-1
 1860 ENDPROC
 1870 :
 1880 DEFPROCdead
 1890 SOUND0,1,5,1
 1900 E%=1:LV%=LV%-1
 1910 VDU31,SX%,SY%,230
 1920 FORL=1TO500:NEXT
 1930 ENDPROC
 1940 :
 1950 DEFPROCexplode
 1960 FORI=1TO5:SOUND0,-15,6,15:FORJ=1T
O10:VDU19,0,RND(6);0;:FOR J2=1TO100:NEX
T:NEXT:NEXT
 1970 VDU20
 1980 ENDPROC
 1990 :
 2000 DEFPROCtitles
 2010 CLS:COLOUR4:FORI%=0 TO30:PRINTSTR
ING$(20,CHR$224);:NEXT
 2020 COLOUR1:PRINTTAB(4,0)"DETONATOR D
AN"
 2030 COLOUR3:PRINTTAB(1,2)" Run from m
ine to "TAB(1,3)"mine defusing them"TAB
(1,4)SPC(4)"as you go.";SPC(4)
 2040 PRINTTAB(0,6)SPC(2)"They can only
 be"SPC(2);TAB(0,7)"defused once active
 "TAB(0,8)"otherwise avoid them"
 2050 PRINTTAB(0,10)SPC(2)"You score fo
r"SPC(5)TAB(0,11)"defusing mines.Bonus"
TAB(0,12)" points are awarded "TAB(0,13
)SPC(2)"for securing the"SPC(2)
 2055 PRINTTAB(0,14)"flags and luring t
he "TAB(0,15)"pursuing boots onto "TAB(
0,16)SPC(3)"defused mines."SPC(3)
 2060 COLOUR6:PRINTTAB(2,18)CHR$226SPC(
2)"MINE"SPC(8)
 2070 COLOUR5:COLOUR135:PRINTTAB(2,19)"
1":COLOUR128:PRINTTAB(3,19)"  ACTIVE MI
NE "
 2080 COLOUR14:PRINTTAB(2,20)CHR$229"
DEFUSED MINE"
 2090 COLOUR1:PRINTTAB(2,21)CHR$225"  B
OOT"SPC(8)
 2100 COLOUR3:PRINTTAB(2,22)CHR$228"  F
LAG"SPC(8)
 2110 COLOUR2:PRINTTAB(2,23)CHR$227"  Y
OUR MAN     "
 2120 COLOUR3:PRINTTAB(2,25)"CAPS-LOCK
LEFT "TAB(2,26)"CTRL      RIGHT"TAB(2,2
7)"]         UP   "TAB(2,28)"SHIFT
DOWN "
 2130 COLOUR1:PRINTTAB(3,30)"PRESS <SPA
CE>";
 2140 I=0:REPEATI=I+1:SOUND0,-15,I,10:U
NTILI=5
 2150 IFINKEY$(100)=" "THEN2160ELSE2140
 2160 ENDPROC
 2170 :
 2180 DEFPROCboot2
 2190 IFBT%=RA%THEN2370
 2200 GO%=GO%+1:IFGO%<>SP% THEN2370
 2210 A%=A%+1
 2220 IFA%>RA% A%=1
 2230 IFS1%(A%)=-32AND BT%<>RA%THEN2210
 2240 *FX21,6
 2250 *FX21,7
 2260 COLOUR4:VDU31,S1%(A%),S2%(A%),224
 2270 Q1%=S1%(A%):Q2%=S2%(A%)
 2280 S1%(A%)=S1%(A%)+(S1%(A%)>SX%)-(S1
%(A%)<SX%):S2%(A%)=S2%(A%)+(S2%(A%)>SY%
)-(S2%(A%)<SY%)
 2290 mv%=FNPT(S1%(A%),S2%(A%)):mvo%=FN
pt(S1%(A%),S2%(A%))
```

```
 2300 IFmv%=14 SOUND0,1,5,1:BT%=BT%+1:S
C%=SC%+100:COLOUR4:VDU31,Q1%,Q2%,224,31
,S1%(A%),S2%(A%),224:S1%(A%)=-32:GOTO23
60
 2310 IFmv%=2 PROCdead
 2320 IFmv%=3 SC%=SC%-50
 2330 IFmv%=60R mvo%=5 S1%(A%)=Q1%:S2%(
A%)=Q2%
 2340 SOUND&12,2,50,10:SOUND3,3,50,1
 2350 COLOUR1:VDU31,S1%(A%),S2%(A%),225
 2360 GO%=0
 2370 ENDPROC
 2380 :
 2390 DEFPROChiscore
 2400 CLS:*FX15
 2410 IFSC%<=VAL(A$(1,0))GOTO2540
 2420 COLOUR1:PRINT''TAB(2)"Congratulat
ions!"
 2430 COLOUR3:PRINT''TAB(3)"You are in
the"
 2440 PRINTTAB(2)"HIGH SCORE table"
 2450 SC$=STR$(SC%):PRINT''"Your Score.
..";:COLOUR2:PRINTSC$:COLOUR3
 2460 PRINT''"Enter Your Name...."'':CO
LOUR2
 2470 A$(1,0)=SC$:X%=&80:Y%=&A:A%=0:!&A
80=&A00:?&A82=10:?&A83=32:?&A84=122
 2480 CALL&FFF1:A$(1,1)=$&A00
 2490 REPEAT:SWAP=0
 2500 I%=0:REPEAT:I%=I%+1
 2510 IFVAL(A$(I%,0))>VAL(A$(I%+1,0)) B
$=A$(I%+1,0):A$(I%+1,0)=A$(I%,0):A$(I%,
0)=B$:B$=A$(I%+1,1):A$(I%+1,1)=A$(I%,1)
:A$(I%,1)=B$:SWAP=1
 2520 UNTILI%=9
 2530 UNTILSWAP=0
 2540 CLS:COLOUR1:PRINTTAB(3)"HALL OF F
AME."
 2550 COLOUR3:FORI%=10TO1STEP-1:A%=LEN(
A$(I%,0)):PRINT'TAB(6-A%);A$(I%,0);"...
";A$(I%,1):NEXT
 2560 ENDPROC
 2570 :
 2580 DEFFNPT(d%,e%)
 2590 =POINT((d%*64)+32,1008-(e%*32))
 2600 :
 2610 DEFFNpt(f%,g%)
 2620 =POINT((f%*64)+32,1023-(g%*32))
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TELETEXT DOWNLOADER CLASH

The Teletext Filing System (TFS) makes available to the user several new commands which are accessible using the '*' format. For example, *PAGE selects the specified teletext page you want to look at. All of the new commands have an equivalent shortened version (*PAGE becomes *P.). However, some of the abbreviations clash with other ROMs on the market such as Toolkit. This ROM will interpret *P. as *PACK. It is advisable, therefore, to use the unshortened versions of all commands to prevent this sort of clash as far as possible. The BBC Telesoftware down-loader program has been amended in this way.

ACCURATELY FILLING RECTANGULAR AREAS - M.Mertens

Filling a rectangular area on the screen using triangle plot options 80 to 87, can lead to the problem of unwanted lines when using the GCOL statement with first parameters greater than zero. This is caused by the desired plotting effect occurring twice at the same place. One method of filling a rectangular area which overcomes this problem is to define the area as a graphics window using VDU 24, setting the background colour and plot option to the required values and perform a CLG to fill the window. A second method involves plotting the two triangles as before, and drawing a line in the required colour and plot option where the triangles meet.(See 'Moving Chequer Board' in BEEBUG Vol.2 No.7 lines 210 to 270.)

SUPER-CONDENSED CHARACTERS ON AN EPSON PRINTER - M.Nixon

According to Epson who make the FX and RX printers, the smallest print possible is that produced when either 'subscript' or 'superscript' modes are selected. However, these are not very legible, in fact the most legible small print is 'condensed'.

If condensed mode is combined with superscript, a new supercondensed mode is formed, which is very legible. This mode is accessible from Wordwise simply by using the following embedded commands to switch this effect on:

OC27,33,20 and OC27,83,1

In immediate mode, the following command may be used:

VDU 2,1,27,1,33,1,20,1,27,1,83,1,1,3

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.

SUBSCRIPTIONS

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

MEMBERSHIP COSTS:

U.K.
£5.40 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for 1 year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

Subscriptions & Software Address	Subscriptions and Software Help Line
BEEBUG	St.Albans
PO BOX 109	(0727) 60263
High Wycombe	Manned Mon-Fri
Bucks	1pm-4pm

PROGRAMS AND ARTICLES

All programs and articles used are paid for at around £25 per page, but please give us warning of anything substantial 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 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

BEEBUG
PO Box 50
St Albans
Herts

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility, whatsoever, for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.
BEEBUG Publications Ltd (c) 1984.

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

BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the 1.2 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 machine 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.

FREE

NEW ROMS FOR OLD
EXCHANGE YOUR 1.0 FOR THE 1.2

We can now exchange your old 1.0 operating system for the new 1.2, free of charge. To take advantage of this offer, please send your 1.0 (supplied on eprom with a carrier board), in good condition to the address below.

£5 FOR YOUR OLD 1.0

If you have the 1.0 operating system and have already bought a 1.2, we will exchange the 1.0 (supplied on eprom with a carrier board) for a £5 voucher. This voucher may be used against any purchase from BEEBUGSOFT.

ADDRESS FOR 1.2 OS:-
ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

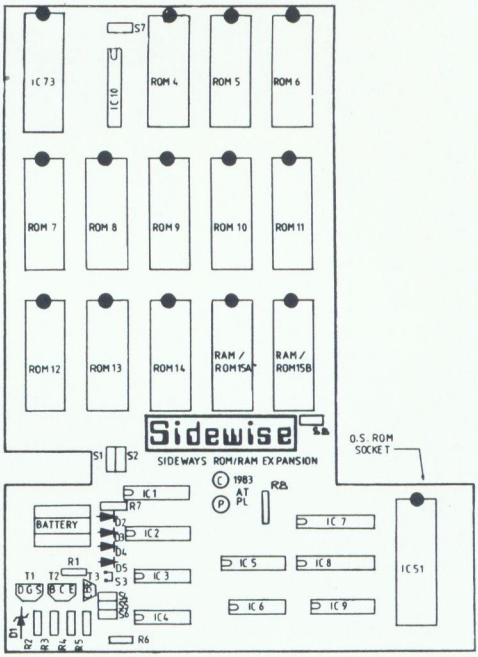
BEEBUGSOFT

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 (parts available directly from ATPL at extra cost)
* 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.

MAGAZINE CASSETTE OFFER

To save wear and tear on fingers and brain, we offer, 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.

All previous magazine cassettes (from Vol.1 No.1Ø) are available.

This month's cassette (Vol.3 No.1) includes: Detonator Dan game, Dominoes game, complete 3D Rotation program plus Spitfire data, Multi-Function Graph Plotter, BEEBUG Workshop Data Entry Routine, program to test Random Access Memory, Function Key Editior, Accurate Arithmetic (two programs), general Move-down Routine for disc users, and our new and super-fast Shapes game.



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.

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 BEEBUG magazine is available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to store the whole of one volume of the magazine as a single reference book. Individual issues may be easily added or removed, thus providing ideal storage for the current volume as well.

BINDER PRICE
U.K. £3.9Ø inc p&p, and VAT.
Europe £4.9Ø inc p&p
(no VAT payable).
Elsewhere £5.9Ø inc p&p
(no VAT payable).

Make cheques payable to BEEBUG.
Send to Binder Offer, BEEBUG, PO Box 1Ø9, High Wycombe, Bucks.
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