INDEPEDENT NATIONAL USER GROUP

FOR THE BBC MICROCOMPUTER

BEEBUG NEWSLETTER

NUMBER 2 VOLUME 1 MAY 1982

CONTENTS

Editorial	2
Book Review	3
Addresses and Video	5
Graphics	6
Spirals and Chords	10
Assembler Guide	11
Hints and Tips	14
User Group Index	16
Merging Programs	16
Games Writing Part 1	17
Letters	21
Members' Discounts	22
Bomber	23
Careers	24
Software Competition	27
If You Write To Us	28

R BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG
ER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBU
TER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEB
TTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEE
ETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BE
LETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER BEEBUG NEWSLETTER B

EDITORIAL

We were hoping to have a written reply to the formal questions that we sent to Acorn over six weeks ago, but in their written response the most important question was left unanswered. This is what we had asked them:

> We are particularly concerned with the condition of the BBC Micros so far delivered. A number of machines have been delivered in a condition caused by bad packaging or posting, such as key-tops adrift. Many members have reported other electrical faults. Would you give us an assurance that both of these issues will be taken up?

From discussions with Acorn at the Computer Fair we have learned that improved packaging is to be used on batches despatched after 29th April 82. They also reproted that the percentage of machines returned is declining, although we are still getting letters from members whose machines do not work.

On the question of delivery dates (and apart from "What is this hole for?" - referring to the plug-in cartridge hole on early models, this was the most asked question as the "Computer Fair" at Earls Court), the answer is still 10 weeks for a model B, but is now around 4 weeks for a model A according to Herman Hauser of Acorn. Towards the end of April BL Marketing were despatching model Bs ordered in November (though due to a mix-up, a number of people who ordered Bs in January, it would appear, have also received their machines). So you can roughly work out your predicted delivery date from this data. Of course, if you ordered a disc inteface or an Econet, your machine will be delayed until those are ready. Several members we talked to at the Computer Fair reported that their machines ran extremely hot. 65 deg C was one quoted temperature, and two members had actually fitted fans inside their machines. If your machine does get very hot it is probably because it has the earlier 'Linear' power supply (black exterior) rather than the later, and more efficient 'Switched' power supply (lighter - possibly silver exterior at the rear). If you have a heat problem we have been told that Acorn will replace your linear supply with a switched one.

A number of members have asked about the guarantee that comes (or doesn't come) with the BBC Machine. The official line on this is that there is a six months guarantee on parts and labour, and that a document stating this fact will be sent at some point to all those who have taken delivery of a machine.

Moving on to BEEBUG itself, our membership has continued to grow at an extremely rapid rate - and has got so large that our disc-based micro could not cope with the volume of data. We therefore decided to hand over the job to a subscription fulfillment agency - hence the new Wandsworth address; and because of an initial backlog here, the first issue of the magazine went out later than we had hoped. Things are now under control, and this issue should have arrived a little earlier in the month than the last. We also apologise for the poor print quality of the program listings in the April issue. We have now 'invested' in a new ribbon for our matrix printer, and this should solve the problem. For those frustrated 3D Noughts and Crosses players, we reproduce in this issue, the program lines which were the most difficult to read.

In this issue, as in the last, we have tried to keep a balance between information appropriate to both beginners and more advanced users, though there is obviously more for the former than the latter. Because of a varied readership, there will naturally be some items that are not appropriate to your particular interests - either

articles that are too basic, or things that you just cannot follow if you are a newcomer to computing. We hope that you will bear with us on this. But please write to us if you think that we have the balance all wrong.

Finally, we have been invited by Herman Hauser (himself) to visit Acorn and we will report on this in next month's issue.

BOOK REVIEW

"Basic Programming on the BBC Microcomputer"
by Neil Cryer and Pat Cryer
Prentice Hall International
ISBN 13-066407-3

This is a very good book for newcomers to programming. It is clearly written, and has plenty of 'activities' for you to try. However, It is not a book for those who want to learn about computers - how they work, what they can and cannot do etc. Granted the title implies that this is a book for learning Basic programming, nevertheless I would like to have seen some guidance on other aspects.

It is one of the cheaper computer books and represents good value for money, almost certainly this is because it does not originate from the USA. The fact that it is written here means that you do not have to put up with spelling such as 'color' and words like 'gotten'. I am impressed by the lack of errors in the book, and only found four minor ones, and one serious one. This shows pretty careful proof reading.

It is a shame that the authors decided to continue the trend of numbering their first chapter - Chapter 0, no-one starts counting from zero, even in programming starting loops, arrays etc from zero can be very confusing.

The book is designed for the absolute beginner, and is well structured. It has chapters devoted to all the major topics of programming, and builds to what the authors consider an apt climax - programmable characters and sound. Each chapter has sections headed:
'Points to think about'
'Discussion on the points to think about'
'Activities'
'Discussion of Activities'
These four headings are better than a simple 'problems' type ending to a chapter.

I have noted some oversights/errors which I would like to see corrected in a second edition. I have listed the main points below, together with the page numbers for reference:

In discussing the IF statement on p43, there is no mention of the fact that you can group AND and OR within the same statement. The IF statement is such an important part of any language, that far more emphasis and examples should be given. The precedence of Boolean operators (AND, OR, EOR, NOT) should also be given. Loops are also very important, and should have more space devoted to them. The authors should also specify the dangers to be found in executing non-integer loops. No mention is made of the fact (p. 50) that a loop is always executed ONCE even if the final value is less than the starting value. This is a fault of BBC/Acorn who should have ideally made the loop operate correctly.

In page 57 (And at many other places) delays are built in using a FOR..NEXT loop, rather than using the clock as follows:

```
TIME=0:REPEAT: UNTIL TIME>100
```

One argument against using a FOR loop is that when the ROM version of the operating system is ready it will run faster than previous versions, hence reducing the time delay. The speed of operation of FOR loops will also depend on how many interrupts are active at the time.

5.10 c asks what value K will have when the loop has terminated. The answer given is 5.2 whereas, due to rounding errors it will not be 5.2 EXACTLY, try this simple test:

```
IF 0.2+0.2+0.2+0.2+0.2+0.2+0.2=1.4 THEN PRINT "CORRECT" ELSE PRINT"WOW"
```

A statement on page 75 says "Amounts of memory are measured in a unit called 1k, which we shall not define", I would have thought that an understanding of the term '1k' was essential to guage how to measure the capacity required for a particular application. What a strange decision just to ignore it.

MID$ is not defined fully on page 126, they omit to say that MID$ can be used with only two parameters. VAL is not properly described, which is important as it is a function that is very commonly used. (pi3i)

The whole chapter on procedures is somewhat poor, personally I disagree with placing all procedures at the beginning of a program and placing GOTOs around them. It appears from statements such as: "You may declare variables 'local' to a procedure. This means that values of the variables on entry to the procedure are restored on exit." that the authors are unaware of what is really happening, or that they are glossing over it for the sake of clarity. However, the real answer is just as easy to understand, so why not use it rather than stating a wrong fact. - Variables declared as 'local' are copied into another area of memory, it is these values that are used, leaving the original values unaltered. (I tend to agree that procedure parameters called by 'name', 'reference', and 'value' are too advanced for the book and BBC Basic does not support them all).
Also in this chapter, they make the statement (p.146) "most computers do not allow procedures"; this is a VERY misleading statement. It is the LANGUAGE, not the computer that determines whether procedures are allowed. Also, virtually every high-level language except Basic allows procedures. Another incorrect statement is to be found on page 142 - it says "Only numbers can be placed in brackets in the PROC statement, strings cannot". This is WRONG, and the example they quote would be better written as PROCinput("Do you wish to continue?"). A procedure that would not accept string parameters would be very poor; as it is on the BBC machine a function cannot return a string value, only a numeric one, this is rather a shame.

Chapter 13 - "File Handling" has been grossly understated. Files are one of the most important aspects of data processing, and the topic is not one that can be skirted over in 8 sides. The authors have written 9 sides on sound, which, although being quite fun for games, is hardly in the same league as files. As the authors say "To give full justice to the SOUND and ENVELOPE statements would require a complete book", so why don't they admit that the same is true of files?

Finally there are several complete omissions. These are as follows: the use of joysticks, digital to analogue, analogue to digital, CNTL-N, CNTL-O, CNTL-L, LISTO, the new O.S.(Operating System) due out later, *FX, *TV, *EXEC, OSFOOL calls, disabling the cursor, and others.

SUMMARY

The book builds up to a climax in describing the sound and graphics features of the BBC machine. It is as if the authors are saying that the machine is best suited for games. The BBC Micro is suitable for far more than just games and I hope that any sequel to this book will redress this imbalance. All in all, a good book that does require a sequel for those who wish to go further. BEEBUG can recommend it's purchase.

S.W.

We have negotiated a special price for this book of £5.90 post free from 'Mine of Information'. The address is given elsewhere in the discounts section, but please note that the 5% discount does NOT apply to this book, as it is already reduced.

SCREEN ADDRESSES

Acorn have kindly supplied us with the following screen addresses for a 32k machine. ~~For the 16k machine you presumably subtract &4000 from all numbers.~~

MODE	
0	&3000 to &7FFF
1	&3000 to &7FFF
2	&3000 to &7FFF
3	&4000 to &7E7F
4	&5800 to &7FFF
5	&5800 to &7FFF
6	&6000 to &7F3F
7	&7D00 to &7FE8

I/O ADDRESSES

Brian Carroll of Aldershot has sent the following list of base addresses for I/O devices that he obtained from Acorn. We have only been able to verify the VIA addresses at present:

FE00	6845 CRT controller
FE08	Serial port
FE10	Serial processor ULA
FE20	Video processor ULA
FE40	1st VIA
FE60	2nd VIA (Port A for printer, Port B user port)
FE80	Disc controller 8271
FEC0	7002 A/D converter
FEE0	"Tube"

COMPOSITE VIDEO OUTPUT

Brian Carroll has also sent us the following information on a colour video fix:

"The BNC connector provides only Black & White output. However, Acorn told me that a 470pF ceramic capacitor from the emitter of Q7 (between IC45 and UHF modulator - BC239) to the base of Q7 (next to the empty IC97 - BC309) will provide colour. It does; and a mini-spst switch in series with the capacitor provides a choice. Incidentally, S26 near the video ULA provides inverted video."

GRAPHICS

THE SCREEN

Text can be displayed in any one of the following ways:- 40x25, 20x32, 40x32, 80x32. The origin for text is the upper-left corner of the screen. So PRINT TAB(10,15);"X" will place an X, 10 characters across and 15 down.

Graphics, which is what we are discussing here, have the origin in the lower-left corner. It is possible to join both the text and graphics cursors so that text can be displayed relative to the graphics origin. Use VDU 5 to join the cursors.

Graphics always conform to the same size screen, unlike the variety of options in the size of the text screen. This size is 1280 horizontal (x) positions, and 1024 vertical (y) positions. A little thought would reveal that a standard TV set has only 625 vertical lines (in Britain), and therefore it is not possible to display 1024 vertical dots. The graphics modes are:-

MODE	SIZE
0	640 x 256
1	320 x 256
2	160 x 256
3	text only
4	320 x 256
5	160 x 256
6	text only
7	text only

Note: Modes 0-3 are only available if you have 32k, as they use too much memory for the Model A.

It does not matter which mode you are in, when plotting a dot it appears in the same position in all modes. This is of profound importance, and is a useful feature allowing you simply to change modes for any reason, eg if you aquire memory expansion at a later date.
It is also possible to change the coordinates of the bottom-left corner which is usually (0,0) to other values, for example VDU 29,500;200; (Note the weird syntax) will change the coordinates of the origin to 500,200.

MOVE & DRAW

These are the two simplest commands, and perform exactly as they say. MOVE 1279,1023 will move to the top-right corner, nothing will appear because you said MOVE and not DRAW. DRAW 0,0 will draw a straight line to 0,0 from the last point (even if it was invisible). Try this out in MODE 4. It is necessary to specify MODE 4 because when the machine is turned on it starts in MODE 7, which has only limited teletext graphics capability, so enter;

```
MODE 4:MOVE 1279,1023:DRAW 0,0
```

a straight line should appear on the screen from top-right to bottom-left. We can go further and plot a rectangle whose bottom-left coordinates are X1,Y1 and whose top-right coordinates are X2,Y2 using the following PROCedure:

```
1000 DEF PROCRECT(X1,Y1,X2,Y2)
1010   MOVE X1,Y1:DRAW X2,Y1
1020   DRAW X2,Y2:DRAW X1,Y2:DRAW X1,Y1
1030 ENDPROC
```

Here is a short program that draws boxes smaller and smaller starting at the maximum screen size. It uses the procedure above:

```
  10 MODE 4
  20 FOR GAP=0 TO 1000 STEP 20
  30   PROCRECT(GAP,GAP,1279-GAP,1023-GAP)
  40 NEXT GAP
  50 STOP
1000 DEF PROCRECT(X1,Y1,X2,Y2)
1010   MOVE X1,Y1:DRAW X2,Y1
1020   DRAW X2,Y2:DRAW X1,Y2:DRAW X1,Y1
1030 ENDPROC
```

You can try changing line 10 to modes 0,1,2, and 5 and you will notice that there is no difference other than the lines being thicker. The reason for the change in thickness is because of the change in resolution.
Let me explain:
If the screen is 1280 points across and you are working in MODE 4 then there are only 320 accessible points across that relate to those 1280. This means that each of the 320 points refers to a block of 4 (1280/320) actual points on the screen. The relationship between the number of points actually lit up when a single point is plotted is given below:

Graphics Resolution	Area Covered
640x256	2x4
320x256	4x4
160x256	8x4

So in MODE 5 the vertical lines will be twice as thick as in MODE 4.

PLOT

There are some things that cannot be done using MOVE and DRAW, and so the PLOT command is available for a variety of other effects, for example 'filling triangles'.

The syntax is PLOT K,X,Y where X and Y are the coordinates of a point and K is defined on pages 136,137 of the P.U.G. If K has the value 4 then the PLOT command is identical to the MOVE command (eg MOVE 100,200 is identical to PLOT 4,100,200) If K has the value 5 then the PLOT command is identical to the DRAW command.

Points can be chosen and accessed either "absolute" (origin is bottom-left corner), or "relative" (origin is where you are now).

You can move, draw a point, draw a line, or fill a triangle.

You can draw in either the current foreground colour, the current background colour, or the logical inverse colour.

You can draw single points, whole lines or dotted lines.

The whole lot is controlled by the value of K. K can range from 0 to 255, but at present only values 0-31, 64-87 are in operation. In order to identify which value of K goes with which operation refer to pages 136,137 of the PUG or you may find that the table below gives you more help:
Add up the numbers next to the operations that you require, check all the operations to make sure that you have not missed any:

0	move to a point but don't draw
0	relative plotting
0	draw a line
1	foreground colour
2	logical inverse colour
3	background colour
4	absolute (origin bottom-left corner)
8	last point in line omitted
16	dotted line
64	point
80	triangle

The total formed in this way is the value of K to use with the PLOT command.
Examples:
1. Draw a dotted line, absolute, in current foreground colour, is 16+4+1=21
2. Fill a triangle, in the background colour, relative to last point, is 80+3+0=83 (Note that there is no value to add for relative plotting).

When filling triangles the triangle to be filled is always between the point mentioned and the last TWO points visited. Hence a procedure to fill a box IN CURRENT FOREGROUND COLOUR with bottom-left corner X1,Y1 and top-right corner X2,Y2 is:

```
2000 DEF PROCBOX(X1,Y1,X2,Y2)
2010 MOVE X2,Y1:MOVE X1,Y1
2020 PLOT 85,X2,Y2:PLOT 85,X1,Y2
2030 ENDPROC
```

The above discussion should give you enough background to experiment. The quickest way to learn is by practice.

Here are some exercises to try:
1. Fill the screen with a grid.
2. Draw a house.
3. Draw a flag with a cross in the centre.
4. Start at a the centre of the screen and move in random directions leaving a trail.
5. Starting at the centre and using the cursor keys, move a dot around the screen leaving a trail.

DEFINING YOUR OWN CHARACTERS AND SHAPES

Another way to produce graphics is to print combinations of graphics characters; and BBC Basic has the facility for you to define for yourself any new shapes or characters you wish.

All presently defined characters are made up of an array (matrix) 8 dots wide by eight dots high. If you have a monitor set (not a TV) then if you look closely at the screen you will be able to see the individual dots that make up each character.

In order to define your own shapes you will need a sheet of graph paper and on it mark off a grid 8 squares by 8 squares. In this grid draw the shape that you want by filling in some of the 64 boxes within this grid. For example, if we wished to define a character as a matchstick man we could design:

```
                1                       These are the values that must
                2 6 3 1                 be added, eg left-hand column
                8 4 2 6 8 4 2 1         has value 128, next column 64 etc.
                ---------------
--***-----      0 1 1 1 0 0 0 0 ------> 64+32+16=112
--***-----      0 1 1 1 0 0 0 0 ------> 64+32+16=112
---*------      0 0 1 0 0 0 0 0 ------> 32
*****-----      1 1 1 1 1 1 0 0 ->128+64+32+16+8=248
---*------      0 0 1 0 0 0 0 0 ------> 32
---*------      0 0 1 0 0 0 0 0 ------> 32
--*-*-----      0 1 0 1 0 0 0 0 ------ >64+16=80
*-----*---      1 0 0 0 1 0 0 0 ------ >128+8=136
```

As you can see, it doesn't look very realistic, however, it will look very much better when you actually display it on the screen, because it will be shrunk considerably.

To tell the computer about the shape that you have decided on is simply a matter of giving each row of dots a value. The value corresponds to the binary equivalent of the positions within the rows used. The diagram above has allocated values to each row. The values of the rows are thus: 112,112,32,248,32,32,80,136.

All the characters presently used on the machine have ASCII codes (American Standard Code for Information Interchange) between 32 and 127, for instance an A has code 65. It would be silly to redefine character 65 to something else (but you may have reasons) so instead space has been allocated in the BBC machine from ASCII code 224 to 255 for you to define up to 32 characters for yourself. It is possible to change other characters but this involves a slightly more complex procedure. Using the range 224-255 does not waste any memory. Let us allocate our man to code 224, so that when we PRINT CHR$ 224 the little man appears. We do this as follows:

```
VDU 23,224,112,112,32,248,32,32,80,136
```

VDU 23 tells the computer we are going to define a character, the 224 is the ASCII code that is being used to store the character, and the remaining eight values are the rows that make up the character. This command can be incorporated into a program.

We can make the man walk across the screen by using the TAB facility (see "Games Writing" in this issue)

DEFINING LARGER SHAPES (will be continue next month).

S.W.

SPIRAL & CHORDS

by Dave Leedham

Here are two interesting short programs submitted by Dave Leedham of Enfield, Middx.

"CHORDS" produces some sweet sounding music that is generated at random, but at the same time it gives the effect of being carefully composed. The three sound channels are running at different speeds and the notes played are selected (at random) from an array containing all possible notes which make up the particular chord being played. The program keeps running until ESCAPE is pressed. (Look at the April newsletter for the actual musical values for the notes)

"SPIRAL" gives some nice visual effects - though to get the full effect you need colour, and 32k RAM.
Two parameters have to be entered. Firstly, the "Sector Angle", which is the angle in degrees of each sector drawn, and secondly the "Radius Change", which determines the rate at which the sectors spiral in towards the centre. When the shape has been drawn, a press on the space bar will start the rotation. A further press will return you to the start. Press ESCAPE to leave the program. To give an idea of the range of effects, try sector angles of 25, 88, 125, 190 with radius changes of between 4 and 16.

```
  10 REM*** MOVING SPIRAL ***
  20 REM*** BY DAVE LEEDHAM ***
  30 DIM COL%(7)
  40 DATA 7,1,2,3,4,5,6
  50 FORI%=0TO6
  60   READCOL%(I%)
  70   NEXT
  80 MODE2:FORI=1TO7:COLOURI:PRI
NTTAB(5,10+I);"SPIRAL":NEXT
  90 PRINTTAB(0,20)"ENTER SECTOR
 ANGLE."
 100 INPUTSA
 110 PRINT"ENTER RADIUS CHANGE,
";INPUTDR%
 120 X%=640:Y%=512:R%=500
 130 AN=0
 140 SA=RAD(SA)
 150 MODE2
 160 C%=1
 170 MOVEX%+R%,Y%
 180 REPEAT
 190   PROCSETCOL
 200   PROCPLOT
 210   UNTILR%<=DR%
 220 A$=GET$
 230 FOREX%=1TO7
 240   FORCX%=1TO7
 250     D%=CX%-EX%:IFD%<1 THEN D%
=D%+7
 260     VDU19,C%,D%,0,0,0
 270     NEXT
 280   FORP=1TO100:NEXT
 290   NEXT
 300 IFINKEY(0)=32THENGO
 310 GOTO230
 320 DEFPROCPLOT
 330 R%=R%-DR%
 340 AN=AN+SA
 350 MOVEX%,Y%
 360 PLOT85,R%*COS(AN)+X%,R%*SIN
(AN)+Y%
 370 ENDPROC
```

```
 390 DEFPROCSETCOL
 400 GCOL0,C%
 410 C%=C%+1:IFC%=8THENC%=1
 420 ENDPROC
```

```
  10 REM*** CHORDS ***
  20 REM*** BY DAVE LEEDHAM ***
  30 MODE5
  40 VDU23;8202;0;0;0;
  50 ON ERROR GOTO270
  60 CLS
  70  ENVELOPE 1,1,0,0,0,0,0,0,1
27,-1,-10,-10,100,20
  80 DIMA(13)
  90 FORI=1TO13:READA(I):NEXT
 100 REPEAT
 110   FORI=1TO10
 120     SOUND 1,1,A(RND(13)),8
 130     FORJ=1TO2
 140       SOUND 2,1,A(RND(13)),
4
 150       FORK=1TO2
 160         SOUND 3,1,A(RND(13)
),2
 170           DRAWRND(1279),RND(1
023)
 180           NEXTK:NEXTJ:NEXTI
 190   RX=(RND(9)-5)*4
 200   FORH%=1TO13:A(H%)=A(H%)+R
X:NEXT
 210   IFA(1)>48 THENRX=-24:GOTO
200
 220   IFA(1)<4 THENRX=24:GOTO20
0
 230   VDU19,1,RND(8)-1,0,0,0,19
,2,RND(8)-1,0,0,0,19,3,RND(8)-1,0
,0,0
 240   GCOL0,RND(4)-1
 250   UNTIL FALSE
 260 DATA4,20,32,52,68,80,100,11
6,128,148,164,176,196
 270 MODE7
```

What the Assembler does

The BBC computer comes equipped with one high-level language - BBC BASIC, though other languages, such as Pascal, are promised. High-level languages are generally easy to use and they have debugging aids such as error reporting etc. The 6502 processor chip which forms the heart of the BBC machine does not understand BASIC. It must be programmed in 6502 machine code, so there are machine code programs stored in the machine's firmware that interpret and execute your BASIC programs line by line. The BBC machine, like most other microcomputers, can be programmed by the user directly in machine code. The chief reason for doing this is speed of operation. By using machine code it is possible to achieve speeds up to 1000 times faster than in BASIC (an average speed improvement is more likely to be between 50 and 100 times) - this is essential in say a chess program where a deep search strategy is required, or in certain high speed graphics applications to take two examples.

Programming directly in machine code is difficult and tedious, because it requires you to enter long lists of fairly incomprehensible code. The use of an 'Assembler' however, transforms the task. What an assembler does is to allow you to enter mnemonics for the program code that the microprocessor uses. So that, for example, instead of having to enter the hex bytes A9FE to get the processor to load its accumulator with the value 254, the assembler can be given the instruction LDA #254. The assembler will also calculate relative jumps, and allow the use of labels, and do much more to make life easier. In the BBC machine it is entered very much like a BASIC program, except that when you type RUN, the code that your mnemonics represent is not executed, what happens instead is that the mnemonics are assembled (translated) into machine code by the assembler, and are stored at a location (which you specify) somewhere in the machines memory.

To actually execute the code you can use the CALL or USR commands in BASIC. So there are two distinct operations - assembly and execution. If we look at a few examples, that should clarify things a bit. The following is an incomplete assembly listing for a very short program

```
100 [
110 LDA #50
120 STA  5000
130 RTS
140 ]
```

The square brackets (these come out as arrows in MODE 7) indicate the beginning and end of the assembly program. Lines 110 to 130 contain three instructions. The first means load the accumulator with the decimal value 50 (the # means that the 50 is data rather than an address). Line 120 means store the contents of the accumulator (ie the data 50) at memory location 5000 decimal. Line 130 means return from the subroutine. We need this last line to effect a return to BASIC when we actually execute the machine code routine.

(NOTE: The accumulator is a storage register through which most operations are carried out on the 6502 processor, its function can be likened to that of a pocket calculator. When we say 'add 6' we mean 'add 6 to the accumulator'.)

The assembly program above is incomplete because so far we have not allocated a space in memory where the machine code instructions

that the assembler will produce are to be stored. One way to do this is to add the following two lines:

```
80 DIM CODE 100
90 P%=CODE
```

Line 80 uses a special convention to assign a block of memory to a variable - we have called that variable CODE. This looks similar to dimensioning an array (as in DIM CODE(100)), but without the brackets around the number, the effect is different. Once memory has been allocated in this way, the variable CODE simply takes on the value of the first free memory location of the reserved block. Line 90 sets the location at which the assembler will start to store the machine code program. You must always use P% for this assignment, but what follows the equals sign may be any address, or any variable used to hold an address (as does the variable CODE).

Having entered lines 80 to 140, if you type RUN, you should see the following

```
0E4E
0E4E A9 32    LDA #50
0E50 8D 88 13 STA 5000
0E53 60       RTS
```

The left-hand column is the address (in hex) where the code is stored, next is the code itself, one byte at a time (again in hex), and to the far right is the corresponding assembler statement. As well as giving this listing to the screen, the assembler also assembles the code into memory. So to check this you could type PRINT ?CODE. This should return the value 169 which is the decimal equivalent of the op code A9 (=LDA). To check this you can type PRINT &A9. The answer will be 169, since the operator '&' signifies that a hexadecimal number follows it. So far the assembler has loaded the code into memory, but he code itself has not been executed. To do this enter CALL CODE. This has the effect of calling the machine code subroutine located at the address CODE. Once you have done this, you can check that it has performed its task correctly by reading the contents of location 5000 to see if it contains the value 50 - ie PRINT ?5000 should give 50.

Storing variables at randomly chosen memory locations (such as 5000) is ill advised, since it may have unexpected effects. It would be better to store them in the block that has been set aside by the command in line 80. We could change line 120 to read 120 STA CODE+50. This will cause the value 50 to be stored 50 bytes up from the start of the machine code program at the address CODE.

The above gives the barest bones of a guide to using the assembler on the BBC machine. If you are interested in programming in assembler/machine code, then we would recommend that you get a book on the subject which will explain the use of the complete 6502 instruction set. In this connection you may find one of the following three books useful:

R. Zaks Programming the 6502
L. Scanlon 6502 Software Design
L. Leventhal Assembly Language Programming

Of these, Scanlon's book is arguably the best for the absolute beginner.

If you do begin to use the assembler in the 'beeb', there are one or two further features that you will need to make use of:

OPT

If your programs involve forward branching, and you are using labels, then when you run the assembler you will get error indications because the assembler does not know where to branch to. A special command OPT (see P.U.G. p131) has been incorporated to cope with this. What you do in such a case is to get the assembler to be passed through twice, and by the second pass the machine knows where all the labels apply. To make the assembler operate like a traditional "two pass" assembler, you need to use a FOR loop (which is a little tedious). The following format works well and is used in the as yet unpublished full user guide (as well as P.U.G. p132).

```
100 DIM PROG 100
110 FOR PASS=0 TO 1
120   P%=PROG
130   [
140   OPT PASS*3
  .     .
  .     . assembler program
  .     .
500   ]
510 NEXT PASS
```

Lines 100,120,130 and 500 use the same format as the program discussed above, but a two-pass FOR loop is inserted at line 110. The OPT statement is used in line 140, and this is arranged so that during the first pass it calls OPT 0 - this suppresses both assembler error display, and assembler listing. During the second pass OPT 3 is called which gives both an assembler listing, and reports assembler errors. NB one member has suggested that a third pass may be necessary under certain circumstances.

Labels - these must be preceded by a full stop - thus:

```
100 .Label LDA  40
```

Multi-statements - you can put more than one assembler line on a single BASIC line using the normal BASIC separator, ie:

```
100 .HERE LDA  20:RTS
```

We hope to have space for more on the assembler in forthcoming issues.

[HEX is hexadecimal coding. Computers work in binary, and hex is a way of representing this in a more useable manner. (See Scanlon or Leventhal mentioned above) for a fuller explanation.]

D.E.G.

BEEBUG HAMS

Many members are using their micros in connection with their hobbies. One of the many hobbies where computers can be of major help is amateur radio. We intend to publish regular lists of members who are also 'Hams'. If you would like to be included in the list please put your name, call sign, and address on a POSTCARD, and send it to our editorial address given on the back page. We will not publish addresses unless specifically requested to do so.

HAM LIST

P. Knight, G3HGR, Sevenoaks, Kent.
John Yale, G3ZTY, Corfe Mullen, Dorset.

NOTE: The Norwich and District User Group have plans for an Autumn meeting entitled "Computing and Amateur Radio", please write to them for further details. (See User Group Index for their address). Please say you saw it in the BEEBUG MAG.

HINTS & TIPS

FX calls in code

```
&FX P,Q,R
can be replaced by
LDA  P
LDX  Q
LDY  R
JSR &FFF4
```

PEEK and POKE equivalents

Indirection operators ! and ?

PEEK

The symbol ? can be used to read, or PEEK, a byte of machine memory, thus PRINT ?65518 will yield 108 (if you have O.S. EPROM 0.1), this is the data at memory location 65518 (&FFEE hex)

The operator ! can be used to read and write four bytes at a time. Thus the command PRINT !65518 (or PRINT ! &FFEE) yields a number made up of four bytes of data at the locations 65518,19,20, and 21. The four bytes are not added, but combined according to:

w+x*256+y*256*256+z*256*256*256

where w,x,y, and z are the contents of the four consecutive locations.

The AND operator can be used to extract any combination of the 4 bytes, or any of the bytes singly.

Thus PRINT ! 65518 AND 255 will yield the value 108, which is the lowest byte. For the rest, things become easier in hex, so to extract the second byte PRINT ! &FFEE AND &FF00, ANDing with &FF0000 will extract the third byte etc.

POKE

To POKE data to the addresses, you can use the following - ?A=B or !A=B where A=the address (or lowest address in the case of !), and B=the data (all 4 bytes in the case of !).

WARNING: Using these operators to write code will not damage your machine, but there are better ways using FX calls, and assembler routines; better in the sense that they are not dependent on the given ROM/EPROM operating system that your machine has, and also they will work across the 'tube'. Some of these calls are however not available on O.S. 0.1 which the majority of members will have at the time of writing. (To find out which you have type *FX 0).

GETTING RID OF THE CURSOR

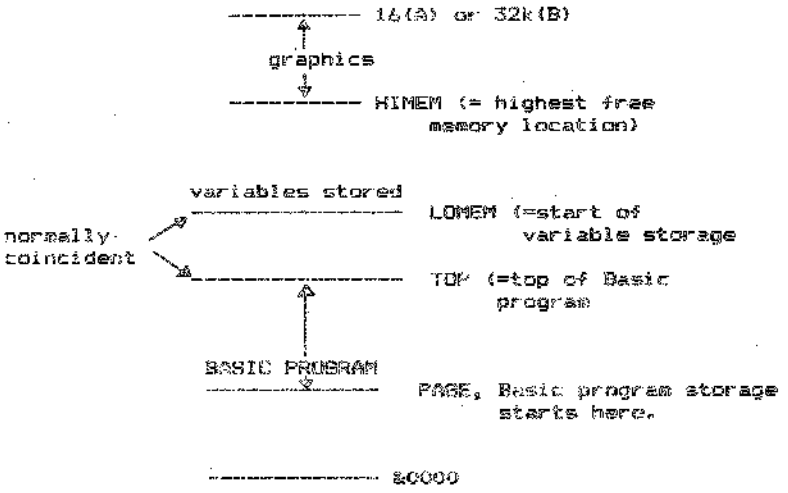
For many applications it is useful to be able to extinguish the blinking cursor. This can be done in a number of ways. Acorn have given us details of two of them:

```
VDU 23;8202;0,0,0
VDU 23,0,11,0,0,0
```

The first leaves a reverse character cursor for editing purposes, while the second removes all trace of either cursor. But both appear to have unwanted side effects. The first seems to corrupt the POINT statement, and both can interfere with character definitions. To avoid this it is necessary to define all required characters using VDU 23... before disabling the cursor. One of Acorn's firmware writers has told us that a trouble-free cursor delete command is being incorporated into the 1.0 O.S.

HIMEM LOMEM TOP & PAGE

Four pseudo-variables for use by the user - details are given in PUG. But you may find the following useful:



Normally LOMEM is set equal to TOP by the machine operating system, but space can be created by raising TOP - eg 100 LOMEM=TOP+200 creates a 200 byte gap for machine code programs etc. Note, however, that the two are reset when new program lines are entered.

```
PRINT TOP-PAGE      Prints how big a program is.

PRINT HIMEM-TOP     Prints how much space is left
                    (ignoring variable storage).
PRINT &8000-HIMEM   Gives space taken up by graphics on model B.
PRINT &4000-HIMEM   Same but for model A
```

The value of PAGE can be reset to allow multiple program storage.

PRINTERS

A number of members have said that they would like to pick up a cheap printer, and that they have been offered the old "Teletype 33", or equivalent printer. This will not work on the BBC Micro because the serial interface (RS423) will not support the 110 baud required by these printers. However, it is probably possible to find a software or hardware fix for this. If anyone has done this, please let us know.

Before accepting a very cheap serial printer, check what baud rate it requires.

ROMs and EPROMs

Version 0.1 of the operating system is in EPROM in most machines. ROM 0.1s are just appearing. Type *FX 0 to find out which you have. ROM versions run at twice the speed, but 0.1 does not support disc, or paged ROMs, and has a few bugs, and less operating system calls.

1.0 is shortly to be ROMmed but this takes about 3 months - 1.0 is required for disc - so, we hear, Acorn will be supplying this in EPROM to those who need it.

USER GROUP INDEX

BEEBUG is happy to act as an information point for local groups. If you would like to be in our index, just drop a line to us and mark the envelope "Local user groups".

Aldershot/Farnham

B. Carroll The Cottage
42, Manor Road
Aldershot GU11 3DB

Bedfordshire

D. L. Evans, 23 Hitchin Road
Henlow Camp
Bedfordshire.

Brighton Area

Jim Price Bedford House
27/28 St Georges Rd
BRIGHTON
Sussex.

Norwich & District

Paul Beverley Room 812a
Norwich City College
Ipswich Road
NORWICH NR2 2LJ

West Midlands Area

Roger Luff Knutsford 288721

OVERSEAS

Norway
N. Christiansen & K.H.Howells
Narienlystveien 2 - Stavne
N-7000 Trondheim
Norway.

MERGING PROGRAMS

Adding two programs together can be useful since it allows you to build up a library of procedures, and then piece them together in programs that you are writing. It will also allow you to use the RENUMBER command on individual parts of programs. A number of members including Mark Page and Graham Taylor realised the great potential here, and Graham Taylor has actually produced a short program to do this. But there is an "official" way to perform this feat, and we bring it to you courtesy of Acorn/BBC.

It involves producing a program file using the *SPOOL command. To create a file on tape of any program called NAME do the following:

```
*SPOOL press return
 set recorder to record, and press return
LIST   press return
*SPOOL press return
       stop the recorder
```

Then load in the program that you want to add this to. Then do the following:

```
*EXEC NAME
 press "play" on recorder
 press return
 ignore error messages
```

As you will see if you try this on a file of any length, things go more smoothly with motor control; though there is no problem without.

Incidentally, the first program can be numbered above or below the second one, though line numbers that coincide will be overwritten by the file program.

This operation is slightly messy, and we hope that it will be cleaned up by O.S. 1.0. Ideally there should just be a single command to save the file rather than 3. And it would be nice if the spurious error messages did not appear.

GAMES WRITING PART 1

Here David Graham puts together a simple animated game, explaining each step of the way.

One good way to pick up tips about programming a new machine is to look at someone else's programs, and borrow some of the techniques used. Here we hope to go one better, and put together an animated game step by step so that you can steal some of the techniques, improve upon them, and apply them to programs you are writing yourself.

This is meant to be a relatively introductory foray into games writing, so we will pick a fairly simple game to work on. The game chosen is called "Bomber". The object is to bomb a number of tower blocks so that your plane can land on the rubble. The plane starts at the top of the screen and makes a number of sweeps, descending lower each sweep. The only control that you have is the bomb release, and the problem is to release your bombs at exactly the right moment. That's the game - it may not, in fact sound it, but it is reasonably gripping to play.

We will now go through it stage by stage, and if you enter the lines as instructed as the article proceeds, you will gradually build up the program on your machine, testing each part as you go.

The first job from a programming point of view is to identify the various parts of the game:

1. A tower landscape must be created.
2. Aeroplane and bomb shapes must be generated.
3. The aeroplane must be moved across the screen descending over lower.
4. As it descends, a watch must be kept to see if it hits a tower.
5. The keyboard must be checked to see if a bomb is to be released.
6. If a bomb is released, a bomb shape must be displayed and made to descend.
7. Towers must be obliterated if the bomb is on target.
8. A safe landing must be detected to end the game.
9. Sound effects are to be incorporated at various stages.

We will take these one at a time, then piece them together, though I will leave the generation of the towers until we have created a viable aeroplane.

GENERATION OF AN AEROPLANE SHAPE

Even before defining character shapes, you need to decide what graphics mode to choose. To keep things simple we will use MODE 4 here. This gives sufficient resolution, and at the same time can be used on model A or B machines.

The best way to get an aeroplane shape is to use the character definition command VDU 23,A,B,C,D,E,F,G,H,I. How the character is built up with the variables B to I is described under "Graphics" elsewhere in this issue.

If you experiment a little with this command, and try to build up a few shapes, you will find that using a single character for an object such as an aeroplane, which is the centrepiece of the display, results in a rather small plane. It would probably be large enough, (though not so well detailed) in MODE 5. After a little experimenting I decided to use the following two characters side by side:

VDU 23,224,128,192,224,255,224,192,128,0
and VDU 23,225, 1,129,192,255, 0, 1, 1,0

The first defines the front half of the plane, the second the back half. To see the effect of this try the following:

```
   5 MODE 4
2010 VDU 23,224,128,192,224,255,224,192,128,0
2020 VDU 23,225,  1,129,192,255,  0,  1,  1,0
2030 PRINT TAB(10,10);CHR$ 225
2040 PRINT TAB(15,10);CHR$ 224
2050 PRINT TAB(10,15);CHR$ 225;CHR$ 224
```

This puts the machine in MODE 4, defines the two characters, and displays them both separately, and together. The last display should look like a reasonably viable plane. You can try modifications of it; but delete lines 2030-2050 before proceeding further with the article.

STRUCTURING

BBC Basic allows greater flexibility in structuring programs than is usual on most micros, and particularly it allows the definition of procedures which may be grouped at the end of the program. We will use a number of procedures in this games program to keep things neat and clear. Our first procedure will be one for defining graphics characters to be used during the program.

We will start the procedure definition at line 2000, and call it PROCCHR. (See p147 of the Provisional User Guide for details of PROC). Lines 2010 and 2020 are the same as above and do not need to be re-input. Here is the procedure definition:

```
2000 DEF PROCCHR
2010 VDU 23,224,128,192,224,255,224,192,128,0
2020 VDU 23,225,  1,129,192,255,  0,  1,  1,0
2050 ENDPROC
```

ENDPROC has been put at line 2050 to allow further characters to be defined within this procedure. If you want to try out this procedure add the following:

```
 5 MODE 4
30 PROCCHR
40 PRINT TAB(20,10);CHR$ 255;CHR$ 224
50 STOP
```

Line 30 calls the PROCedure at line 2000, and then line 40 prints the pair of characters. The STOP at line 50 is necessary, because otherwise the machine will enter the procedure definition as the program runs, and have nowhere to exit when it encounters ENDPROC.

MOVING THE PLANE

The plane must be moved both across the screen, and then down the screen one line after each crossing. This is easily accomplished using the TAB command (syntax TAB(X,Y) see P.U.G. p171), and changing X and Y with a pair of FOR...NEXT loops.

Try the following in conjunction with lines 2000 to 2050 above:

```
  5 MODE 4
 30 PROCCHR
170 FOR BX=0 TO 39
200   PRINT TAB(BX,10);CHR$ 225;CHR$ 224
600 NEXT BX
```

There are a few things wrong with this! Firstly the plane moves

too fast, so we will insert a slowing down statement:

220 TIME=0:REPEAT:UNTIL TIME>=3

(See P.U.G. for details of TIME, REPEAT..UNTIL). This slows things down to a reasonable pace.
Secondly we must erase the old plane's images from the screen as we go, or else we'll end up with a screen full of them. To do this modify line 200 to:

200 PRINT TAB(B%,10);" ";CHR$ 225;CHR$ 224

This does the trick, but we have pushed the nose of the plane over too far ahead, and it appears on the next line. This overlapping will prove inconvenient later. So reduce the FOR loop at line 170 to FOR B%=0 TO 37

Next we want the plane to get lower by one line after each sweep of the screen. To achieve this we will introduce a FOR loop on the Y parameter of the TAB command. Enter:

150 FOR A%=1 TO 30
650 NEXT A%

and change 200 TAB(B%,A%);" ";CHR$ 225;CHR$ 224

If you run this you will see that the plane descends as it should, but a single plane character gets left on the right hand side of each line:
620 PRINT TAB(38,A%);" " (double space)
has the effect of printing a double blank space in the correct position at the end of each line after the completion of each A% loop.

BUILDING THE TOWERS

The towers can be built either by using the PLOT command to fill pairs of adjacent triangles, so making rectangles, or by creating a single block character, and building this up using the TAB command. We will use the latter. The character can be defined by entering the following VDU command into the definition of PROCHR:

2030 VDU 23,255,255,255,255,255,255,255,255,255,255

This designates character 255 as a solid block. We will now write a procedure to build this into a tower. Add the following:

3000 DEF PROCTOWER
3080 FOR I%=30 TO 10 STEP -1
3090 PRINT TAB(15,I%);CHR$ 255
3100 NEXT I%
3200 ENDPROC

Add line 60 PROCTOWER, and run the whole program. You will see a single tower produced at position 15 (on the x-axis). It is 20 units high. For this game, we need several of these, and of different heights. Let the number of towers required be NX, and for the moment we will assign NX the value 15 in line 10. Enter:

10 NX=15

Next we will use the RND function to produce a single tower of random height up to 20 units. Add the line 3060 JX=RND(20), and change line 3080 to 3080 FOR I%=30 TO 30-JX STEP -1
Here the height (JX) is assigned a random value between 0 and 20, and this is then used in the FOR loop. Note that this is a decreasing (STEP -1) loop.

Each time you run this, you should get a single random height tower up to a maximum of 20 units (or picture cells 'pixels'). To get NX (ie 15) towers in random positions, add the following:

3050 FOR L%=1 TO NX
3070 KX=RND(20)+10
3110 NEXT L%

The variables I%,J%,K% and L% are each only used within PROCTOWER. We can therefore define them as LOCAL so enter 3020 LOCAL I%,J%,KX,L%

(See P.U.G. p113 for explanation of LOCAL)

CHECKING FOR A CRASH

At present when you run the program, the plane will glide through the towers, oblivious to the obstacles they are meant to represent. To incorporate the possibility of a crash, we must write in a routine which checks the state of the 'pixel' (character square) immediately before the plane enters it. We can do this using the POINT statement. This checks the colour of a point at the centre of the 'pixel' to be entered next by the plane. POINT simply returns a code for the colour of any given point on the screen (See PUG p139). The value is -1 if the point is off the screen, and 0 if it is black. Any number higher than zero indicates a colour at the given point. The syntax is POINT(X,Y), where X and Y are the graphics coordinates of the point to be tested (ie ranging from 0-1279 and 0-1023 respectively). The PRINT TAB statement that we are using to drive the plane employs character (text) coordinates in the range 0-39 and 0-31 respectively. But as well as this difference, the Y coordinates are reversed - that is to say that position 0,0 is top-left for the PRINT statement, but bottom-left for PLOT and POINT. We have employed a conversion factor, and used this not in a procedure this time but in a FUNCTION call. The following should be entered to define the function:

```
2500 DEF FNCRASH
2550 X%=1280*((B%+3.5)/40)
2570 Y%=1023*((31.5-A%)/32)
2600 =POINT(X%,Y%)
```

You will see that like a procedure this is entered outside the main body of the program.

Lines 2550 and 2570 convert the character parameters to parameters suitable for use with POINT. The last line closes the Function definition, and assigns to the function the value POINT(X%,Y%). Now every time you call up FNCRASH, this will give a value depending on the colour at the centre of the pixel at coordinates (B%,A%).

To test this out enter the line:

```
210 IF FNCRASH>0 THEN STOP
```

Now if you run the program, everything should stop as soon as the nose of the plane touches a tower.

This is all we have space for in this issue, we will pick up the threads next month, and complete the program. You may like to experiment with what we have done so far. But if you can't wait for part two, the whole program listing is given in this issue.

D.E.G.

3D NOUGHTS and CROSSES PROBLEMS

We must apologise for the poor quality of reproduction of the 3D Noughts and Crosses program in last month's newsletter. Given below are those lines that were most difficult to read:

```
880  FOR C=0 TO 8
890 D=Q(N(M(A,C))):T=T+W(N+D)+P(E(M(A,C)
    DIV 4))*B(C+C)

2140 PRINTTAB(0,6);1'TAB(0,12);2
    'TAB(0,19);3'TAB(0,26);4

1810  PROCMOVE(X,YY(R)-R*T,3)
1820  PROCDRAW(XX,YY(R)-R*T)

2330  VDU 29,0;0;1;
2340  RETURN
2350   DY=49-(3-D)*4
2360   DX=20*(E+2)-25*E
2370   XD=20*(E+3)-25*(E+1)
2380   VDU 29,80;Y(D)*3;

2420      PROCMOVE(Y*DX/DY+25*E+2,Y,G(F))
2430   PROCDRAW(Y*XD/DY+25*E+23,Y)

2520  DEF PROCMOVE(XX,YX,C%)
```

Dear BEEBUG,
Firstly thank you for producing such an excellent Newsletter, it is certainly superb value for money!
I have also had great difficulty in saveing programs onto cassette. I have tried three auto-level recorders, and although they record the 'blips', it is impossible to load back into the computer. I suspect the recording level is either not high enough or distorted.....
Is it possible to load part of a program onto cassette, however incomplete or incorrect?

Alan Mothersole, Middx.

A number of people have had difficulties loading and saving programs. Some of these (though not, I think you), have simply been using wrongly connected leads. The information that comes with the machine is certainly very limited, and one or two people have referred to the diagram on p. 223 of P.U.G. and connected their input between pins 3 and 5 (rather than 3 and 2) -2 is common to both output and input.
In your case part of the trouble may well be caused by the automatic record level in your machine. One way to get some indication as to what is going wrong is to save a program, and then play it back through the speaker of the tape recorder, and compare it to the Welcome tape (which seems to load from any machine). The levels of the two recordings should be similar, and the noise level(hiss) on your own recorded tape should also not be too high. It is likely that your auto-recorded tape will be at a lower level than the Welcome tape; and it might be worth tinkering with the innards of the recorder to rectify this.
Another solution would be to save and load at 300 rather than 1200 baud. The command *TAPE 3 will set the beeb to 300, and *TAPE will return it to 1200.

D.E.G.

Dear BEEBUG,

You explained in the April issue of the magazine about the below-spec ULAs, perhaps you could tell me how we would know whether we have one or not - what are the symptoms?

Could you get hold of and publish a circuit diagram of the machine?

J.G.Stephenson, Bryanston School,
Blandford, Dorset.

We have been told that below-spec video ULAs will produce flicker in graphics modes 0 and 1, though our upgraded model A does not do this. If anyone has a model B with this symptom, it would probably be worth writing to Acorn. As to the circuit diagram, Acorn have not yet made it public (for fear of pirating we hear), though we have had a privileged glance at it. Even when it is off the secret list, it may well be subject to strict copyright limitations.

D.E.G.

DISCOUNTS

BEEBUG have arranged discounts for members at a number of retail outlets who supply computer books, software and hardware. We are at present negotiating further discounts including one with a service agent for the BBC Micro.

HAPPY MEMORIES Gladestry Kington Herefordshire HR5 3NY (054 422) 618	Computer hardware 10% off all 'one-off' prices. Quantity prices may be negotiable.
MINE of INFORMATION Ltd (Mail order) 1 Francis Avenue St Albans Hertfordshire St Albans 52801	Computer Books 5% discount.
TECHNOMATIC Ltd 15 Burnley Road London NW10 01-452 1500	Hardware, software and books 5% discount.

Members should simply quote their membership number with their order, though members taking discount will not necessarily be given credit card facilities, (you must check on this). We are not acting for these companies, nor receiving payment from them, and cannot be held responsible for their services.

BOMBER 16k

The object of the game is to bomb a series of tower blocks so that your ever-descending plane can land safely on the rubble. There is only one control, key 'B', which releases the bombs - you are only allowed one bomb in the air at a time. There are 5 skill levels, though you might also like to adjust the waiting loop in line 2i0 to make the plane move more quickly, and to add colour to the display. Instructions are included with the program, but see also the article "Games Writing" in this issue.

```
   1 REM ****BEEBUG BOMBER****
   2 REM VERSION 1.1
   5 MODE4
  20 PROCSTART:PROCLEVEL
  23 IFLEX=0 THEN MODE7: END
  25 MODE4
  30 PROCCHR
  32 VDU23,0,1,0,0,0
  35 FX=0:VX=35:WX=0:SC%=0
  40  PLOT4,0,0:PLOT5,1280,0
  50   FORX=1TO38:PRINTTAB(I,31)
CHR$255;:NEXT
  60 PROCTOWER
 100 REM BOMBER LOOP
 150 FORAX=1TO30 STEP S%
 170   FORBX=3TO37
 200     PRINTTAB(BX,AX)" ";CHR$
225;CHR$224
 210     IF FNCRASH>0 THEN AX=50
:BX=50
 215     IF INKEY$(0)="B"ANDFX=0
THEN VX=AX+2:WX=BX+1: FX=1:SOUND0
,-10,0,1:SC%=0
 220     TIME=0:REPEAT:UNTIL TIM
E>=5
 250     IF FX=1 THEN PRINTTAB(W
X,VX-1)" ";TAB(WX,VX)CHR$226
 270     VX=VX+1:IF VX=31 THEN F
X=0:PRINTTAB(WX,VX-1)" ";SOUND0,-
10,2,1
 600     NEXT BX
 620   PRINTTAB(38,AX)"  "
 630   SC%=SC%+1
 650   NEXT AX
 700  IFAX>40 THEN PROCSND ELSE
PROCLANDING
 800 PROCLEVEL
 820 IF LEX>0 AND LEX<6 THEN 25
 900 MODE7
1000 END
1999 REM******
2000 DEFPROCCHR
2010 VDU23,224,128,192,224,255,2
24,192,128,0
2020 VDU23,225,1,129,192,255,0,1
,1,0
2025 VDU23,255,255,255,255,255,2
55,255,255,255
2030 VDU23,255,255,255,255,255,2
55,255,255,255
2040 VDU23,226,0,24,24,24,60,60,
24,0
2050 ENDPROC
2499 REM******
2500 DEF FNCRASH
2550 XX=1280*((BX+3.5)/40)
2570 YX=1023*((31.5-AX)/32)
2600 =POINT(XX,YX)
2999 REM******
3000 DEFPROCTOWER
3020 LOCALIX,JX,KX,LX
3050 FORLX=1TONX
3060   JX=RND(20)
3070   KX=RND(20)+10
3080     FORIX=38 TO 30-JX STEP-1
3090     PRINTTAB(KX,IX)CHR$255
3100     NEXT IX
3110   NEXT LX
3200 ENDPROC
3499 REM******
3500 DEF PROCSND
3550 FORQX=-150 TO 0
3560   SOUND0,QX/10,6,1
3570   PLOT70,RND(100)-50+XX,RND
(100)-50+YX
3590   NEXTQX
3595 PRINTTAB(0,0);
3600 ENDPROC
3999 REM******
4000 DEF PROCLANDING
4050 PRINTTAB(5,10)"GOOD LANDING"
4060  PRINTTAB(5,12)"SCORE ";SC%*
S%;"  LEVEL ";LEX
4100 ENDPROC
4499 REM******
4500 DEFPROCSTART
4550 PRINTTAB(16,5)"BOMBER"
4570 PRINT''''"The object of the
game is to bomb the"
4580 PRINT"towers to achieve a s
afe landing of "
4590 PRINT"your plane."''
4600 PRINT"You have only one con
trol, key 'B', "
4610 PRINT"which releases the bo
mbs."''
4620 PRINT"Only one bomb is allo
wed in the air at"
4630 PRINT"a time."
4635 ENDPROC
4639 REM******
4640 DEF PROCLEVEL
4645  PRINT''''"Enter skill level
(1 to 5)"
4650 PRINT"to start game or"
4655 PRINT''"Enter zero to exit ";
4660 LEX=GET -48
4670  IFLEX>5 OR LEX<0 THEN 4640
4700 NX=15:SX=1
4710 IF LEX=2 OR LEX=4 THEN NX=20
4720 IF LEX=5 THEN NX=25
4730 IF LEX>2 THEN SX=2
4800 ENDPROC
```

CAREERS

This program requires a 32k machine, and when run will ask a number of questions. At the end it will print both alphabetical and ranked career lists either to the screen or printer. The program is presented here as the basis for a fairly serious careers analysis program. As it stands, it has a reasonable set of questions asked to the user, but its repertoire of career categories on file could usefully be extended.

Please remember that the suitability factors used for the various jobs in the program is a subjective matter, and the values were allocated by people who were NOT careers advisers. If anyone developes this program further, then we would welcome its inclusion in our software library.

As far as extending it is concerned I will show how to add a 'Systems Analyst' to the list of jobs. Once you have the idea, then the only limit to the number of jobs that can be included is the computer's memory. I, personally would not try to extend the number of questions too far, because the user will rapidly lose interest, however, a good list of jobs is essential.

To add a job you must first change the value of M in line 60, by adding one. Next you must produce a weighting factor between 0 (not important at all) and 9 (absolutely essential) for each of the questions. Place the job title in alphabetical order in the DATA statements in line 1040 to 1080, and place the row of weighting factors in the corresponding place between lines 1250 and 1300.

You should check that your row of weighting factors is the same length as the other DATA statements. In the end you will have allocated 51 numbers from 0-9, eg for a 'systems analyst' I would suggest:

```
009213000000104000007178555110732797707856680388064
```

```
  10  REM       CAREER ANALYSIS
  20  REM          15-IV-82
  30  REM------------------------------
  40  VDU 3:REM Disable Printer
  50  MODE 7
  60  M=27:REM Number of job categories.
  70  N=51:REM Number of questions.
  80  M=M-1:N1=N:N=N-1
  90  DIM U(N),A(M),J$(M),R(M),C(M)
 100 VDU 12: REM Clear the screen
 110  PRINT TAB(12)"C A R E E R S"
 120  PRINT TAB(12)"-------------"
 130 PRINT '"This program will ask you questions,"
 140 PRINT"it will then match you with a list of"
 150 PRINT"careers. The first career in the final"
 160 PRINT"list will be the most suitable. Beside"
 170 PRINT"each career you will find a '
 180 PRINT"'suitability factor' ranging from"
 190 PRINT"-1000 to 1000."
 200  PRINT"Answer all questions on a 0-9 scale,"
 210  PRINT"where 9 is the most favourable response."
 215  VDU 15:REM Turn of Pacing
 220  PRINT'
 230  REPEAT
 240   INPUT"What is your name";Name$
 250    UNTIL LEN(Name$)>1
 260 PRINT '"Do you want me to print your responses "
 270  PRINT"to my questions "Name$;"? ";
 280  C$=CHR$(GET AND 223):PR=(C$="Y")
 290  IF PR THEN VDU 2: GOTO 310
 300  IF C$<>"N" THEN 280 ELSE VDU 3
```

```
310  PRINT '''"Career analysis for ";Name$
320  PRINTSTRING$(21+LEN(Name$),"-")
330  REM Read M Job catesories
340  REM-------------------------
350  FOR I=0 TO M
360    READ J$(I):B(I)=-1:C(I)=-1
370    NEXT I
380  C1=0:S1=0
390  FOR J=0 TO N
400    READ C$:C1$=C$
410   PRINT C$;TAB(27);
420    REPEAT
430      A=GET-48
440      UNTIL A>=0 AND A<=9
450    PRINT ;A
460    U(J)=A
470    C1=C1+U(J):S1=S1+U(J)*U(J)
480    NEXT J
490  C1=C1/N1
500  X=S1/N1-C1*C1
510 IF ABS(X)>1 THEN 570
530 PRINT '"Please try again; but this time very    your responses"
540  PRINT
550  RESTORE
560  GOTO 350
570  PRINT
580 PRINT TAB(4);"ALPHABETICAL CAREERS LIST"
590 PRINT TAB(4);"-------------------------"
600  PROCHEADING
610  FOR I=1 TO M
620    C2=0:S2=0:P2=0
630   READ D$
640   FOR J=0 TO N
650     D=VAL(MID$(D$,J+1,1))
660     C2=C2+D
670     S2=S2+D*D
680     P2=P2+D*U(J)
690     NEXT J
700   C2=C2/N1
710   Y=SQR(S2/N1-C2*C2)
720   Z=P2/N1-C1*C2
730   A(I)=INT(1000*Z/X/Y+.5)
740   A=LEN(STR$(A(I)))
750   PRINT TAB(11-A);A(I);TAB(17);J$(I)
760   E=INT(A(I)*0.012+11): B=B(E)
770   IF B=-1 THEN B(E)=I: GOTO 820
780   IF A(B)<A(I) THEN B(E)=I: C(I)=B: GOTO 820
790   C=C(B):IF C=-1 THEN C(B)=I: GOTO 820
800   IF A(C)<A(I) THEN C(B)=I: C(I)=C: GOTO 820
810   B=C: GOTO 790
820   NEXT I
830  J=0
832  IF PR THEN PRINT#7:GOTO 850
835  VDU 14:REM Turn on pagins
840 PRINT'"When output stops press 'SHIFT' key to  continue"'
850 PRINT TAB(4);"SORTED INTO SUITABILITY ORDER"
860 PRINT TAB(4);"-----------------------------"
870 PROCHEADING
880 FOR I=M TO 0 STEP -1
890   B=B(I)
900   IF B=-1 THEN 940
910   A=LEN(STR$(A(B))): J=J+1
920   PRINT TAB(11-A);A(B);TAB(17);J$(B)
930   B=C(B): GOTO 900
```

```
  940   NEXT I
  945  PRINT''''!VDU 3
  950 PRINT"Do you wish to review the printout (R)" or start again (s) or exit (
E) ";
  970 C$=CHR$(GET AND 223)
  980  IF C$="S" THEN RESTORE: GOTO 215
  985 IF C$="R" THEN 830
  990  IF C$<>"E" THEN 970
 1000 PRINT:VDU 3:END
 1010 REM      ---------------
 1020 REM **** Job categories ****
 1030 REM      ---------------
 1040 DATA Accountant,Advertising,Airline pilot,Architect,Armed forces
 1050 DATA Banker,Civil servant,Clergy,Computer Operator,Computer Programmer
 1055 DATA Education(Teaching),Engineer
 1060 DATA Entertainer,Estate manager,Farmer,Industrial research
 1070 DATA Managerial,Medical services,Musician,Pharmacist,Police
 1080 DATA Politician,Retailer,Social services,Solicitor,Surveyor
 1085 DATA Systems Analyst
 1090 REM      ----------------------
 1100 REM **** Subjects and questions ****
 1110 REM      ----------------------
 1120 DATA Biology,Chemistry,Computers,Maths/Statistics,Physics/Engineering
 1130 DATA Economics,Geography,History,Politics,Archaeology,Greek
 1140 DATA Latin,Art,Divinity,English Language,English literature
 1150 DATA Music,French,German,Other languages
 1160 DATA Ambitious,Artistic
 1170 DATA Cautious,Confident,Cooperative,Critical,Dependable
 1180 DATA Energetic,Extrovert,Impulsive,Intellectual,Persuasive
 1190 DATA Practical,Precise,Realistic,Sociable,Systematic
 1200 DATA Unpredictable,Like meeting people,Want high wage/salary
 1210 DATA Want indoor work,Want regular hours,Like facts and figures
 1220 DATA Want to work with machines,Like to take risks,Quiet
 1230 DATA Creative,Like organising,Like sport,Get on with people
 1240 DATA Want to exert authority
 1250 DATA0458464444000600540000207778901156496449256789108847177:REM Accountant
 1260 DATA0045454003006605404008846735477429233097567627712187:REM Advertising
 1270 DATA0045445023040054055030687497387359928138106780387691:REM Airline pilot
 1280 DATA0025445004008054050079556644947736986435757542737267:REM Architect
 1290 DATA2005555002000054050083778699334688772564112781689891:REM Armed forces
 1300 DATA0055064441060005005020777990425769758556888367253661:REM Banker
 1310 DATA0045065061800054050092080700573982332845788118180781:REM Civil servant
 1320 DATA0005005543450665400042769485634566785381452176284991:REM Clergy
 1323 DATA0092120000000040000070688371426446457425915705441661:REM C Operator
 1326 DATA0093220000001060000081788960417739829338977308991451:REM C Programmer
 1330 DATA4406555550500546545004557897875835866728286312417288:REM Education
 1340 DATA0546844004000005500034757586426189728347565736862441:REM Engineer
 1350 DATA0005005001005054600084485547972453375768100161211:REM Entertainer
 1360 DATA6505557503000054000033577675796874664444222396145?:REM Estate manage
 1370 DATA7505006006000054050021786459553477667323101627536611:REM Farmer
 1380 DATA5655545446000050050051777674538798758317688608701221:REM Indus researc
 1390 DATA0006464553050055050081788783748766677578766274192881:REM Managerial
 1400 DATA6645560001056054050031778586749789677167862240061661:REM Medical serv
 1410 DATA0000000000605055740069575858763335576483697001311261:REM Musician
 1420 DATA5605540002050055040022886592238279647555984112751531:REM Pharmacist
 1430 DATA0404404442000053050060079699736876677173011711779811:REM Police
 1440 DATA0026484577854056640806668717268236726867531351771?:REM Politician
 1450 DATA0005349022004053040054778586752633885374885108170861:REM Retailer
 1460 DATA0034442243000540400527896978358444961924212618199:REM Social servic
 1470 DATA0034456584250056054020997762506818859275873156170891:REM Solicitor
 1480 DATA4455656604045055000346587847387375784363996271504911:REM Surveyor
 1490 DATA0092130000001040000071789551107927797078866800800641:REM Systems alyst

 1499  END
 1500  DEF PROCHEADING
 1510  PRINT

 1520 PRINT TAB(4);"Suitability  Career"
 1530 PRINT TAB(4);"   factor    choice"
 1540 PRINT TAB(4);"-------------  ------"
 1550 ENDPROC
```

SOFTWARE COMPETITION

Programs of all types (eg games, educational, business etc) are eligible. They should be submitted on a cassette with explanation, instructions, and documentation on paper (typed if possible); and accompanied by the application form below (or a copy of it). Members may submit more than one program, but each entry must be sent under separate cover. If you require the tape to be returned, please enclose a suitable stamped addressed package.

Prizes range from £10 to £50. There will also be two special reserved prizes of £100 which will be awarded by the editors to programs which they judge to be of outstanding merit. Should no programs fulfil this criterion within the date limit of the competition, the date limit for these two prizes will be extended.

Programs must arrive by 30th June 1982. Judging will be carried out by the editors, and their decision is final.

Post to BEEBUG, PO BOX 50, St Albans, Herts AL1 2AR
Mark entries in top left hand corner - "COMPETITION"

BEEBUG SOFTWARE COMPETITION - ENTRY FORM

Program Title:..... Name:.....

Category:..... Membership No:.....
(Games, Educational (This is essential)
Business, other) Address:.....

Will run on Model A.... B.... .....

.....

.....

The program submitted here is my own work, and has not been submitted to another organisation.
I understand that if I am a prize winner my program may be used by BEEBUG for its program library or for publication. In either case this would be with acknowledgement, but without further payment.

Signed..... Date:.....

IF YOU WRITE TO US

Please note our new address is:

BEEBUG
P.O. BOX 50
St Albans
Herts AL1 2AR

Please do not send mail to the old address, it will be considerably delayed, and please do not call personally.

If you write requiring a reply then please send an SAE, and write the word QUERY on the top left-hand corner of your (outer) envelope.

Contributions for the newsletter are welcome - copy date is 1st of the month, although we need more notice for substantial items. Longer items should preferably be typed. Please write the word NEWSLETTER on the top left hand corner of your envelope.

PROGRAMS - if these are longer than a few lines, please send us a cassette (with appropriate SAE if you want it back). If instructions to use the program are not contained within it then please send written instructions (preferably typed).

MEMBERSHIP APPLICATIONS should be addressed to:

BEEBUG Dept. 1.,
374 Wandsworth Road,
London SW8 4TE.

From 30 June membership costs £4.90 for 5 issues, £8.90 for 10 issues. Cheques should be made payable to BEEBUG.

NEXT MONTH

Next month's newsletter is due out in mid-June. In it we will be continuing with the second parts of both "Games writing" and "Graphics", and producing an article entitled "Writing Structured Programs in BBC Basic". Plus:
Some useful procedures that you can incorporate into your programs.
More on upgrading (postponed from this issue) and on using some of the hidden features of the model B.
More programs and members' contributions.

BEEBUG NEWSLETTER is edited and produced by Dr David Graham and Sheridan Williams, and its contents are subject to copyright.