INDEPENDENT NATIONAL USER GROUP

FOR THE BBC MICROCOMPUTER

BEEBUG NEWSLETTER

VOLUME 1 NUMBER 3 JUNE 1982

CONTENTS

2 Editorial
4 Polygon (16k)
5 Structuring in BBC Basic
7 Hints and Tips
8 User Port on A and B
(including upgrading part 2)
10 Screenplay (32k)
10 110 Baud fix
13 Video and TV review
16 Blunders
17 Games writing part 2
19 Disc news
20 Assembler part 2
21 Mini word processor (32k)
22 BEEBUG discounts
23 Read a character from screen
23 User group index & BEEBUG hams
24 Maze Trap (32k)
25 Verify
26 Cursor delete
26 Procedure anomaly
27 Software competition
28 If you write to us

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

Firstly we would like to thank all of those members who have written to us saying nice things about the newsletter. We have so far received literally scores of such letters, and more arrive daily. Since BEEBUG is obviously a totally new venture for us both, it is extremely encouraging to find that we are more or less doing it right; though of course we see plenty of room for improvement. We have also received some useful suggestions about the newsletter, and are trying to put these into practice.

The size of BEEBUG's membership continues to grow, we have now reached the 5000 mark, and as we said last month,a consequence of this is that we have had to hand over subscription and magazine circulation to a subscription agency in order to leave us time to work on the magazine. Therefore if you have comments to make regarding circulation etc, please contact the subscription agency (rather than our editorial address) in the first instance.

We are also getting a little overwhelmed with technical queries. If you do have a query, could you write it on a postcard along with your membership number and place it in an envelope with an SAE - mark this 'QUERY' and send it to the editorial address. This will make things easier for us to handle - but please be patient for a reply, and please include your membership number. This is important, since we cannot answer non-members queries.

Software Competition

The software competition closes on the 30th June, and we hope to squeeze the winners list into the July newsletter. If your entry is too late for that date, or you have other software that you would like to enter, we are running a second competition which closes on 30th September. From amongst the prizewinners of each competition, the best entrants will be offered a commission to write further software for BEEBUG. Some competition winning programs will be appearing in the newsletter from time to time, and others will go into our software library.

Condition of machines

Thank you to all those who helped with our survey on the condition of machines at arrival; we have now completed this, and do not need any further responses. If you are hot under the collar about a faulty machine, it would be best now to write directly to Acorn, although if it is a despatching or administration matter, write to - John Radcliffe, Executive Producer, Broadcasting House, London, WC2. The findings of our survey were generally that people were delighted with the machine, though unhappy with the provisional user guide; and many were unhappy about packaging and the condition of machines at delivery. The most common fault was loose keytops, but running a close second was keys which did not function because of postal damage - and similarly LED indicators (bottom left) had been forced up through the case, making the latter difficult to remove. There were also some machines that functioned for a relatively short time before crashing forever. When members received a faulty machine it tended to take around 3 or 4 weeks before it was repaired. Members also found contacts with BL Marketing to be less than satisfactory. It is, of course, almost impossible to phone them, and there appear to have been countless errors in the administration of machine allocation.

We have brought all of these matters to the attention of the directors of Acorn, and they have generally said that things have been put into effect to improve matters. We were told by Hermann Hauser that a new design of packaging was to be used to protect the keyboard, taking effect from 29th April. We are not sure whether this has been implemented, though we note with concern that the indistinct FRAGILE notices on the package have not yet been replaced.

Chris Curry of Acorn also told us that the reason for the chaos at BL (a subsidiary of Weetabix who used to mail out handbags!) was changes of orders, and that vast numbers of people who ordered a BBC machine changed their order at least once. In view of all this we were told that there will be some considerable reorganisation at BL. This will take effect by around the end of June. Let us hope that this resolves the problem.

GUARANTEES

By the time this magazine comes out, all beeb owners (I wish they had called it Proton or something) should have received a mail package from Acorn (via BL!). Amongst other things it will contain a guarantee card giving 6 months guarantee from date of purchase, covering parts and labour. The mail-shot will also contain a list of Acorn dealers who can do upgrading, maintenance etc.

ACORN VISIT

We said in last month's newsletter that we had been invited to Acorn, and would report back on our findings. We had an interesting time (though lunch was a little disappointing - we didn't get any) and as well as "accidentally" discovering a room piled high with what could only be returned BBC machines, we had a useful talk with Chris Curry, joint managing director of Acorn, where we were able to raise some of the problems that beset beeb owners. We also spent an interesting hour with Roger Wilson and others on the technical side. Here we gleaned a few ideas and fixes - such as how to get the serial port going at 110 baud for use with surplus teleprinters - we also learned about SOME of the bugs in operating system 0.1.

There were demonstrations too, proving that the BBC machine actually DOES support disc drives, and that it will work with a second processor across the tube. The demonstration used a 6502 running at 3MHz, and it was quite obvious that in programs with equal amounts of data manipulation and input/output handling, the tandem unit could function at about twice normal speed (since the beeb handles output to the screen and other I/O processes, while the second processor performs the central programming tasks).

Here are some of the outcomes of our discussions with Chris Curry:

ULA

We reported in the April issue that some below spec video ULAs had been fitted to model As, and that when a dealer upgrades your machine he will test the ULA and replace it if necessary. We were told by John Coll of Acorn that if members upgraded themselves (!) and found that they had been given a below-spec video ULA, Acorn would not supply ULAs unless they were part of a full upgrade kit (even if a ULA blew after the guarantee period). We reported this in the April newsletter, and not surprisingly some of you wrote to us urging that we press Acorn on this "unacceptable" position.

We put all this to Chris Curry, and while assuring us that very few inferior ULAs were in fact used, he has agreed that below-spec devices will be replaced free of charge if you take your machine to a dealer. The symptoms to look for appear as degrading of one kind or another in the higher resolution modes (and particularly in mode 0). This can take the form of flicker, ragged characters, spread out cursor, or even dashes all over the screen.

ROMs and EPROMs

In the April and May newsletters we indicated some of the shortcomings of the 0.1 O.S. (operating system) that all users will currently have, either in EPROM (4 chips), or in ROM (one chip) - See April issue page 6, and May issue page 15 for

further details. One reader wrote to us in disbelief about the bugs we reported, but I'm afraid that it is all true! In fact it is, if anything, slightly worse than we thought. Here is the confirmed (but not necessarily complete) list of 0.1's shortcomings:

It has less *FX calls.
It has minor bugs in graphics plotting and sound routines, plus one or two in the cassette data file system.
It will not support disc drives.
It will not support the page mode for adding extra ROM utilities.
The serial RS423 port will send but not receive data.

Supported by a number of members letters we put it to Chris Curry that the machine thus supplied fell well below the specification issued by the BBC, and based upon which, decisions were made by members to buy their BBC machines. Will members therefore be given the 1.0 O.S. free of charge when it comes out?

After some discussion the reply that we received was that if you have 0.1 in EPROM, then when 1.0 is available in ROM (about 3 months or so) you can take your machine to your nearest dealer and he will effect the change over free of charge. If you have a 0.1 in ROM on the other hand, a nominal charge will be made. This seems to us a reasonable position to adopt, and we will not press for the very last ounce of flesh - always assuming that 'nominal' is what it says. We do however feel quite strongly that now the various errors in the 0.1 O.S. are known, further machines sold with the 0.1 O.S. should be accompanied by a list of its shortcomings, so that owners do not spend fruitless hours trying to get their serial port to work etc. - of course members of BEEBUG will know all about it anyway! Over and above this, the best advice that we can offer is for you to hang on to your 0.1 EPROMs until ROM 1.0 is available (rather than trade them in for 0.1 ROMs). Two addenda while we are on this subject. If you are purchasing the disc interface, then Acorn will supply you with 1.0 EPROMs (until 1.0 ROMs are available), and secondly it would appear that the 0.1 ROM thinks that it is still a volatile EPROM, and if you type *FX 0 it will tell you as much. This means that the only way to really know what you have is to look inside the box (Careful - see BEEBUG issue 1 page 6). Incidentally, we are told by Chris Curry that the first disc systems should be ready for despatch by the end of June, and we will be carrying a review of this as soon as we can get hold of one.

Lastly if your power supply runs red hot and burns the polish off your table, then Acorn will replace it (the power supply that is), so contact them before fitting a fan or buying an asbestos table.

POLYGON 16k

This program by A.J.Vincent of Cheshire draws a series of polygons from a triangle up to a 20-sided figure. It achieves this by calling the procedure PROCPOLYGON(NO). This procedure draws a polygon of NO sides in the centre of the screen, and may be useful in its present form, or modified so as to be able to alter the position and size/shape of the figure:

```
100REM POLYGON BY A.J. VINCENT.
110K=5:XD=640:YD=512:X=400:Y=400

120FOR NO = 3 TO 20
130MODE4
140VDU19,0,4,0,0,0,19,1,3,0,0,0
150PROCPOLYGON(NO)
160TIME=0:REPEAT:UNTIL TIME=300
170NEXT NO:END

180DEFPROCPOLYGON(NO)
190MOVEXD,YD+Y
200FOR ANGLE =0 TO 2*PI STEP(2*
PI/NO)
210PLOT K,X*SIN(ANGLE)+XD,Y*COS
(ANGLE)+YD
220NEXT ANGLE
230PLOT K,XD,YD+Y
240ENDPROC
```

STRUCTURING IN BBC BASIC

It often comes as a shock to people to discover that a professional programmer does not sit in front of a computer or terminal for most of the day.

A programmer is not usually involved in the design of a system that is to be programmed, but instead codes the program from a precisely written specification. The programmer, therefore has very little say over the three major parts of all programs:

The input data to the program.
The method used for processing the data.
The screen or printed output design.

I realise that you will probably have to do all three items above but do not sit in front of your computer expecting to develop a wondrous program. Far better to sit in another room and work there designing how it is to be written. For some reason, if the machine is in the room with you it compels you to enter things via its keys.

If you take a typical project that is to be computerised, whether it be a game or a data processing task, a sizeable proportion of the time should be allocated to designing the program and its operation.

So, consider the following points before ever approaching the computer:

1) Write down PRECISELY what it is that you intend to do. Give it to someone else to read, if they cannot follow your instructions, then you have little hope of turning it into a program. This stage is deciding on the 'algorithm'. That means the rules and method governing the solution to the problem.
2) Design your screen layouts using grid paper (or coding sheets) EXACTLY as you want them.
3) Decide on what data your program requires, which order to accept it, and what validation checks you will make on each data item.
4) Decide on names for all your variables. Choose meaningful names. Variable names are not restricted to letters and digits, they may also contain the underline character (just above the 'return' key). Using this makes your programs even more readable, for example you can have a variable called His_name or Overdraft_limit.
5) If your program requires files then make sure that you have file designs that include - file type, method of access, record description, etc.
6) Divide your program into modules (ie separate entities that can be tested independently).

It is while writing the modules that a structured approach pays dividends. With a structured approach you should find that you can go back to your program days or even weeks later, have at your side your notes such as variable names, procedures, and flowcharts, and be able to continue within minutes of where you left off.

It should be possible to write all programs without GOTO statements, for example:

UNSTRUCTURED

```
10 INPUT"How old are you";age$
20   age=val(age$)
30 IF age<=0 OR age>100 THEN 10
```

STRUCTURED

```
10 REPEAT
20   INPUT"How old are you",age$
30   age=val(age$)
40 UNTIL age>0 AND age<100
```

The unstructured program with the GOTO is shorter, but GOTO statements require your eye to follow where they are destined.

Programs developed without thought, while sitting in front of the computer tend to end up with a statement such as GOTO 100 and statement 100 GOTO 720, and

statement 720 GOTO 950, all because some lines were missed out and the programmer redirected them somewhere else. I have seen this happen DOZENS of times. Program flow tends to be erratic when GOTO statements are involved. All this can be avoided using a structured approach.

Basic on the BBC machine is not really structured in the same way as say ALGOL, Pascal, or Comal but it does have some useful features that help considerably - long variable names, REPEAT...UNTIL, PROCedures. If you can imagine what Basic would be without FOR loops then the above features each add to Basic as much sophistication again.

We will look at structuring programs in BBC Basic under a number of heading:

1. Procedures
2. IF..THEN..ELSE
3. FOR loops, REPEAT loops

PROCEDURES

These have been provided in other high-level language since their inception, however they are a newcomer to Basic and need some explaining.

Procedures give us a way of performing a set of statements using just ONE statement, very similar to GOSUB but more powerful. It might be best to forget that GOSUB exists at all as it is just a very inferior procedure call.

A good program structure to aim for would be a "master" section of the program that just contained procedure calls surrounded by FOR..NEXT and REPEAT..UNTIL statements. For example:

```
100 PROCedure1                    150 REPEAT
110 FOR tune%=1 TO 10             160   PROCedure4
120   PROCedure2                  170   PROCedure5
130   PROCedure3                  180 UNTIL tired
140 NEXT tune%                    190 END
```

Suppose that you wanted to write the following program section:

```
IF condition THEN
    s1      (where s1, s2, s3, and t1, t2
    s2       are each Basic statements)
    s3
    ELSE
    t1
    t2
rest of program
```

In standard Basic this would have to be done as:

```
100 IF condition THEN s1:s2:s3:GOTO 130
110 t1
120 t2
130 rest of program
```

Note how your eye needs to scan to find line 130. In BBC Basic we could write:

```
100 IF condition THEN PROCs ELSE PROCt
110 rest of program
```

In the above, s and t are simply the (user defined) names of two procedures PROCs and PROCt. We would then place the corresponding procedures that execute s1,s2,s3 and t1,t2 at the end of the program somewhere, for example:

```
1000 DEF PROCs                    2000 DEF PROCt
1010 s1:s2:s3                     2010 t1:t2
1020 ENDPROC                      2020 ENDPROC
```

See the Provisional user guide p147 & p74 for details of DEF PROC and ENDPROC. If careful thought has been given to the design of the program then many routines could be written in this way, and tested before they are joined into one.

Once you want to make a procedure general, then you will need to place parameters in brackets after their name. For example here is a procedure that will print text centred on line Y%.

```
1000 DEF PROCcentre(Q$,Y%)
1010 LOCAL T%
1020  T%=(40-LEN(Q$)) DIV 2
1030   PRINT TAB( T%,Y%);Q$
1040 ENDPROC
```

To print 'Hello' centred on line 10 we would call the procedure by:

```
10 PROCcentre("Hello",10)
```

There are two 'parameters' to PROCcentre, but there may be any number of parameters (including none). They may be string or numeric. Values are transferred via these parameters to the procedure. This article is not meant to show you how to use procedures, but when and why to use them to maintain a structured program, so refer to the provisional guide for further details.

You should do your utmost to write programs without GOTO statements, we ourselves have not adhered to this in our published programs as much as we ought. Our feeble excuse is that some of the programs have been converted from other machines - but we will try harder next time.

Next month we will look at the REPEAT...UNTIL structure, and how it can help with the readability of programs, and begin a procedure column in the magazine giving useful or interesting procedures that may be of general use. S.W.

HINTS & TIPS

*MOTOR 1 or just *M.1 turns the cassette motor on.
*MOTOR 0 or just *M. turns the cassette motor off.

Terry Bromilow has discovered how to get the background screen colour effect used in the BBC's "Computer Programme" try: MODE 6:VDU 19;4;0;
Try changing the 4 to other values, for instance 1 gives RED, 2 GREEN, 3 YELLOW etc.

Nicholas Clifton of Chislehurst, Kent supplied us with the following:

1. To disable the 'Escape' and 'Break' keys: This makes the program very difficult to list once it is running.

```
   10 ON ERROR GOTO 10000
   20 *KEY 10 "OLD|M RUN|M"
   30 .
      . your program
      .
 9999 END
10000 IF ERR=17 THEN GOTO ERL
10010 PRINT"Error in line ";ERL
10020 END
```

Amongst other things, this shows the fact that the 'Break' key is recognised as user defined key 10 by the operating system, and can be programmed just like the other 10 user defined keys.

2. The word THEN in an IF statement is optional except when a line number follows.

More bugs

There are at least two bugs in the cassette data filing system. One occurs because the BPUT routine doesn't save the accumulator on the stack (so you can arrange a PHA....PLA fix for it).

The other has no simple fix, but causes occasional blocks to be unrecognised on reading. Acorn say the thing to do is to save each file twice to get round this one!

FUNBUGS For several people's favourite error message enter: AUTO 1000,1000
For a colourful bug, switch on the machine from cold, type OLD return, 255 return.

USER PORT ON A & B

(including upgrading part 2)

We continue the treatment of upgrading the model A by installing the user port, and give details here (to be continued next month) of how to program it on both upgraded A, and standard model B machines. But first a note on what the user port can do. A port is a channel through which data may be input to the computer from the outside world, or through which data may be output to the outside world. Thus influencing its surroundings. The user port installed in the BBC machine is what is called a parallel port - this means (in this case) that the data is transferred in parallel across 8 data lines simultaneously. On input, the port can be used (in the simplest case) to detect the state of push buttons and games paddles, while on output, the port allows relay control of external devices to give one example. In practice the port can be used to interface a wide range of more complex devices, though for the moment we shall restrict ourselves to simple interfacing applications.

Installing the port

Only one integrated circuit (I.C.) is required to make the user port on model A functional. This is the 6522 VIA (IC 69), and we described how to insert this in the April issue for the printer upgrade (since the user port and printer share this I.C.). Once the 6522 has been installed, it is only necessary to solder in a socket (PL 10) to the board. The plug required is a 2 x 10 way right angle I.D.C. pcb mounted connector (male) and may be obtained from a number of suppliers (we obtained ours from Watford Electronics).

Refer to the instructions and precautions given in the April issue regarding guarantee invalidation and how to remove the case etc, and solder in the plug to PL 10. Note that there are 11 (not 10) positions on the pcb. You should place the connector so as to leave the left-most pair of holes empty, as suggested by the white overlay line on the pcb. When you have done this, put the machine back together again, taking care not to push the LED indicators through the circuit board (as explained in issue 1). You now only need to connect up a suitable female plug. The plugs are insulation displacing (hence I.D. connector), which means that this requires no soldering (you use a vice and a piece of 20-way ribbon cable), but you can also buy these plugs ready wired.

The provisional user guide (p 220) gives you the connections to the IDC socket on the pcb (PB7 is pin 20), and to complement this, we give a plan for the connections as they emerge from the ribbon cable (fig 1). As you can see there are 8 data lines labelled PB7 to PB0 interspersed with earth lines. PB7 stands for port B line number 7.

The 6522 is a fairly complex input/output device containing two 8-bit I/O ports, and a serial port as well as housing two 16-bit timers. On the BBC mchine all of these facilities except for one of the two I/O ports (port A is used for the printer) is at the disposal of the user. Because of the complexity of the 6522 we would advise those interested to obtain data sheets on the device, or consult an appropriate book. Leo Scanlon's "6502 Software Design" which we mentioned last month in connection with using the assembler has a useful chapter on the 6522.

Even without a manual, there is quite a bit that we can cover in the newsletter. Briefly speaking the 6522 is a 16 register device - this means that there are 16 separately addressable sections to it. It has been connected up in the Beeb in such a way that the 16 registers behave like ordinary memory locations within the memory map of the machine. Acorn have given it a base address of &FE60 (that is 65120 decimal). This means that register zero of the device is located at 65120, register 1 at 65121, and so on up to register 15 (the sixteenth) at 65135. To see what data resides in any of these registers, you just need to PEEK the

appropriate addresses (PEEK is explained in the May issue p 14). Thus the command PRINT ?65120 (or PRINT ?&FE60) will yield the contents of register zero, and so on. Similarly you can change the data in any of these registers with a POKE (again see May p 14). Thus the command ?65122=255 will put the value 255 into register 2. You can check that it has done so by PEEKing it. It should be possible to put into this particular register (register 2) any integer value between 0 and 255. Note however, that you cannot alter all the registers in this way, and that changing some of them may hang up your machine (until you press 'break'). Incidentally there are better ways to access the port on the 1.0 operating system, but until this becomes available we will have to use PEEK and POKE - or we could use a short routine in assembler of the form: LDX # register: LDA # value: STA &FE60,X

This loads the X-register of the 6502 (not the 6522) with the 6522 register number, the accumulator with the data (value), then stores the data in the appropriate 6522 register using an indexed command in the final line (which provides an offset dependent on the values in the X register of the 6502).

6522 handling program

At the end of this article there is a listing for a 6522 handler. This fairly short program displays the contents of all registers of the 6522 on the BBC machine, and allows each to be altered. The screen printout gives both register number and initials indicating the function of each register, as well as its data content. The program is so arranged that entering a register number >15 causes all registers to be read but not altered. You will find when you run it that some registers read a different value each time they are read. These are the timer registers 4,5,8 and 9 (unless the timers happen to have been stopped for some reason). Again it should be emphasised that that you cannot put any value of data into any register that you choose, and random use of the registers may lock up your machine (until you press 'break').

The Parallel Ports

The two parallel ports are operated from registers 0 to 3. And of these, only registers 0 and 2 affect port B (the user port). So for the moment we will concentrate on these. Their use is, in fact, extremely straightforward. Register 2 is the data direction register for port B (ie DDB). If it is assigned the value zero then data can be input through the port to the computer on all 8 data lines, and if it is at 255 then data can be output through the port from the computer to the outside world. In fact, values between 0 and 255 will allow input on some of the 8 data lines, and output on the remainder. The I/O status of the port is thus programmable via register 2. For the moment we will just consider the two extreme cases of all lines set to input, or all lines set to output.

Input port

If you enter the value zero into register 2 (ie DBB) you should also see the data in register 0 - the data register for port B (DRB) - change to 255. The 255 now represents a value read in through port B - ie through the data register. If you now short line PB7 to earth and at the same time read the registers of the 6522, you will get the result 127 in register zero, indicating that the short has been detected at the port, and that we have successfully passed data into the machine. In fact each of the 8 data lines is assigned a value according to the table opposite. This is done in such a way that if a given data bit or data line is 'high' it will contribute the corresponding figure to the data read at the port. If it is in a 'low' state it will contribute zero to it. 'High' is usually somewhere between 3 and 5 volts, and 'low' is usually less than 1v. The nature of the 6522 circuitry is such that the data lines in the input mode take on high values when left floating. This is why the port initially read 255. If you

PB7	128
PB6	64
PB5	32
PB4	16
PB3	8
PB2	4
PB1	2
PB0	1

add the numbers 128,64,32,16,8,4,2, and 1 it comes to 255. Each line was 'high' and so contributed its value to the total. When we shorted PB7, we brought it low, so that its contribution of 128 was no longer made. This is why the value read in was 127 (ie 255-128). It is obvious that we can use this to good effect for reading the states of switches and pushbuttons.

Next month we will cover hardware and software to achieve this, and will also have a look at data output through the port.

D.E.G.

```
   80  REM 6522 HANDLER
   90  REM BEEBUG MAY 1982
  100 A%=&FE60
  110 DIMA$(16)
  120 MODE7
  150 PROCREGS
  200 PROCSCREEN
  220 PROCPORT
  300 PRINTTAB(15,20)SPC(20);TAB(15
,20);
  310 INPUT R%
  320 IF R%>16 OR R%<0 THEN 220
  330 PRINTTAB(15,21)SPC(20);TAB(15
,21);
  340 INPUT D%
  350 IF D%>255 OR D%<0 THEN 330
  360 ?(A%+R%)=D%
  370 GOTO 220
 1000 DEF PROCREGS
 1010 RESTORE
 1020 FORI%=0TO15
 1030   READ A$(I%)
 1040   NEXT I%
 1050 DATADRB,DRA,DDB,DDA,T1L,T1H,T
1L,T1H,T2L,T2H,SR ,ACR,PCR,IFR,IER,
DRA
 1080 ENDPROC
 1200 DEF PROCSCREEN
 1210 CLS
 1220 FORI%=0 TO 15
 1230   PRINTTAB(0,I%+3)I%;TAB(16,I
%+3)A$(I%)
 1235   NEXTI%
 1250 PRINTTAB(4,20)"REGISTER"
 1260 PRINTTAB(4,21)"DATA"
 1280 PRINTTAB(10,1)"6522 VIA AT &F
E60"
 1290 PRINTTAB(4,23)"(Register valu
e > 16 causes a read)"
 1300 ENDPROC
 1400 DEF PROCPORT
 1410 FORI%=0 TO 15
 1420   PRINTTAB(20,I%+3)?(A%+I%);"
  "
 1425   NEXT I%
 1430 ENDPROC
```

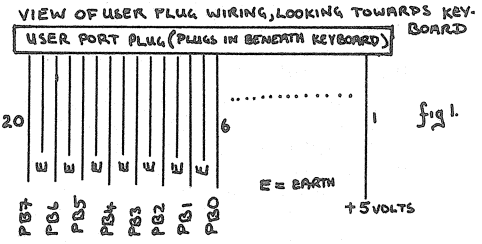


fig 1.

SCREEN PLAY

Ian Bell of Hatfield, Herts has supplied a program to show the speed of the BBC Micro's graphics and the ease and speed of the assembler, we could not get it to work on a model A despite his suggestions so it is given here as a model B program:

```
 20 MODE 2:NC=16
 30 P%=&1100
 40 [OPT0
 50 .ST LDX #0:STX &70:LDX #&30:STX &71
    :LDX #0
 60 .HERE LDA &70:STA (&70,X):INC &70
    :BNE HERE:INC &71:LDY &71
    :CPY #&80:BNE HERE :RTS
 70 ]
 80 CALL ST
 85 REPEAT
 90  VDU 19,RND(NC)-1,RND(8)-1,0,0,0
100  REPEAT UNTIL GET=32
110 UNTIL FALSE
```

Pressing the space-bar will change the pattern.
Try changing the first &70 of line 60 to &FE48. Also try deleting line 100.

110 BAUD FIX

It can be done! Swap link S28 (ie cut the link, and solder another from the free pin to the centre pin). Then use *FX 8,1 to set 75 baud. The resultant output stream through the RS423 is 108.333 baud, which is near enough. This will also make the cassette port run too fast, so a single-pole, double-throw switch on link S28 would be a smart idea.

GRAPHICS PART 2

Last issue we defined the shape of a man; and I said that you could make him walk across the screen. I expect you noticed the error on the third line of the man's shape table. It should have been 32 not 64. This means that the VDU statement was also slightly wrong, the correct statement is given in line 10 below. Making him walk can be achieved with the following:

```
  3 MODE 4
  5 REM Define the shape as ch 224
 10 VDU 23,224,112,112,32,248,32,32,80,136
100 REM Now make him move across screen
110 FOR X%=1 TO 39
120   PRINT TAB(X%,10);CHR$ 224:REM Draw man
130   TIME=0:REPEAT:UNTIL TIME>50
200 NEXT X%
```

Notice how he leaves a trail behind him. In order to erase the trail we must plot a space where he was previously, before plotting him in the new position. We can do this by inserting line:

```
140 PRINT TAB(X%,10);SPC 1
```

[NOTE:You must enter a graphics mode (eg MODE 4) before you can define a graphics character.]

If you want very fast graphics you will get a better display if you clear the previous character just before you plot it in the new position. It is better on the eye for the character to be displayed for as long as possible. For example in the program above it would look better when run at speed if you were to use:

```
115 PRINT TAB(X%-1,10);SPC 1
```

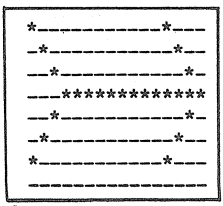
instead of using line 140.

A slightly more readable program would be to define the string at 20 MAN$=CHR$ 32 + CHR$ 224 and then 120 PRINT TAB(X%,10);MAN$ Delete lines 115 and 140 if you have left either of them in.

DEFINING LARGER SHAPES

It often occurs that the shape that we want to display just won't fit into a dot array 8x8. In which case you will have to use a larger array, for example 8 across by 16 down, or 16 across by 16 down.

Consider a large arrow made using 16 columns and 8 rows as shown below:



Each half must be defined separately, so the left hand half is

```
VDU 23,224,128,64,32,31,32,64,128,0
```

and the right hand half is:

```
VDU 23,225,8,4,2,255,2,4,8,0
```

These two characters can be joined into a single string variable using:

```
ARROW$=CHR$ 224 + CHR$ 225
```

Now whenever you want it to appear all you need do is say

```
PRINT TAB(10,10);ARROW$
```

It is slightly more complex if you need a shape that is two vertical characters. A vertical arrow of size 8 across by 16 down for instance would be defined as follows:

```
-----*--- = 8
----***-- =28
---*-*-*- =42
--*--*--* =73    Top half - VDU 23,224,8,28,42,73,8,8,8
-----*--- = 8
-----*--- = 8
-----*--- = 8
-----*--- = 8.................................................
-----*--- = 8
-----*--- = 8
-----*--- = 8
-----*--- = 8  Bottom half - VDU 23,225,8,8,8,8,8,20,34,65
-----*--- = 8
----*-*-- =20
---*---*- =34
--*-----* =65
```

However we cannot just join them into one character as before because we would then get a character looking like this: ↑A. A little thought will tell us that we need to define it as: 'top half of arrow' 'cursor left' 'cursor down' 'bottom half of arrow'. This can be done provided we know how to move the cursor left. CHR$ 8 is cursor left, CHR$ 10 is cursor down. So here we go:

```
ARROW$=CHR$ 224 + CHR$ 8 + CHR$ 10 +CHR$ 225
       top-half   csr-lft  csr-dwn  top-half
```

Now we can display it as a single character

```
PRINT TAB(10,10);ARROW$
```

as before.

POKING to the screen

Although this can be done in any mode, it is easiest to demonstrate in mode 7 (Teletext mode). In issue 2 page 5 we printed the screen addresses for all the graphics modes. The screen is 'memory mapped' this means that there is a location in memory corresponding to each cell of the screen display. The memory address of the top left-hand corner in mode 7 is &7C00 and the bottom right corner is &7FE8 (though unexpectedly this changes when the screen is scrolled). So if we place the code for an 'A' (&41) in location &7C00 then an 'A' should appear in the top left hand corner. We do this using the instructions given in issue 2 page 14 as follows:

```
MODE 7:?&7C00=&41
```

Here is an interesting little program to display random characters all over the screen:

```
10 MODE 7
20 CH=RND(255)
30 P=RND(&3E8)+7BFF
40 ?P=CH
50 GOTO 20
```

I have only shown this as a matter of interest. If you do want to place characters on the screen the best way is to PRINT them there with TAB. The POKEing method above will not work across the tube, and makes for non-portable programs. This is clearly a disadvantage, and should be avoided wherever possible.

S.W.

LIMITATIONS OF MICROCOMPUTER GRAPHICS

Microcomputers (including the BBC Micro) suffer from being very slow (in computer terms). Even in machine code this lack of speed starts to show if we try really complicated routines. One example is filling a triangle, which is a built-in feature of BBC Basic. This is done at machine code speeds, and yet note how slow it is, it is definitely not instantaneous.

The sophisticated displays that you see on computer programs on TV where a 3D object is rotated slowly, or you simulate a flight through the solar system, is done either with an incredibly fast mainframe computer, or it is done by cheating. By cheating I mean that each frame is calculated and displayed by the computer and is then photographed, then the next frame is displayed etc. Another method of cheating is to avoid the calculation phase, and have all the succeeding coordinates stored in memory. However provided that you only have to change a small portion of the screen each time then it is still possible to get quite fast moving graphics. It is only when vast chunks need updating that the speed really deteriorates.

Once you are in Basic the speed gets worse by a factor of about 50. Note the difference in speed between drawing a line in machine code and the same line in Basic:

```
Machine code: MOVE 0,0:DRAW 1000,1000
       Basic: FOR X%=0 TO 1000:PLOT 69,X%,X%:NEXT
```

Of course it could be argued that this is not a totally fair test, but it does make the point I hope.

There will be more on BBC graphics in future issues. Next month we take a look at the teletext mode.

S.W.

VIDEO & TV REVIEW

One of the most frequently asked questions from members has been "Should I buy a monitor set, will it give a better picture than my television?". There have been other questions in a similar vein such as "I am considering buying a colour TV, should I alter my choice now my BBC micro has arrived?"

This article is aimed at describing the difference between the various types of set on the market, and will give you guidance, hopefully enabling YOU to make the final choice. We won't recommend any particular set, and bear in mind that there are hundreds on the market that we haven't reviewed, and these could represent better value for money than the ones we review here.

We have reviewed 8 sets: 2 colour TVs
1 colour TV/with video input
2 colour monitors
1 monochrome TV
2 monochrome monitors

The points we looked for were colour separation, clarity of 80 column text, TV reception quality, weight/size, and crispness of the image at the edges of the screen. We only had some sets for a period of about three/four weeks so we cannot comment on their reliability I am afraid.

What is the difference between a monitor and a TV?

Television signals are coded (modulated) to enable them to be transmitted. The TV set then needs to decode (demodulate) this signal before feeding it to the rest of the television circuitry. Your computer produces a pure video signal, but in order to use an ordinary TV set this signal has to be modulated before feeding it

into the aerial socket, all this just so the TV can demodulate it again! This modulating and demodulating degrades the signal, and hence degrades the picture considerably. A better solution is not to modulate the computer's video output but to feed it into a TV that has no demodulating and R.F. circuitry. Such a TV is called a 'monitor' (in some contexts 'monitor' also implies a higher quality of display). Monitor sets should be cheaper than TV sets because there is no UHF circuitry, but due to the limited quantities made, their price is still higher than TVs (a lot higher if the picture quality is superior). If you have a circuit diagram for your TV it would be possible to place a video input on your TV, but this can be highly dangerous if not performed correctly and should only be tackled by a very competent electronics engineer.

RGB

The BBC computer has 3 alternative picture signal outputs: the modulated UHF signal, a video signal and a so called RGB output. RGB means Red, Green, and Blue. Colour TVs have three guns, one controlling each of red, green and blue. By altering the intensity of each, all the colours required in a TV picture can be obtained. In a colour computer these three colours are controlled by the computer, and they are mixed if fed to the TV lead or the monitor lead. This signal is split up again inside the TV set. The RGB socket has enough pins to feed each of the red, green and blue colours separately without mixing. These are fed directly to the TV guns, and lead to the best quality colour picture.

Not many TV makers feel it is worth wiring in a video or RGB input, probably for much the same reason that cars do not have starting handles any more. (It is a very minor selling point and not worth the extra production costs).

Actual review

The sets reviewed ranged from a second-hand 1973 GEC set bought for £40 to the Salora colour monitor at £465. The details are given below:

1. GEC	22"	colour	television	£40 (second hand)
2. NEC 12P63	14"	mono	television	£59
3. ZENITH	13"	mono	monitor	£82
4. APF TVM-10	9"	mono	monitor	£95
5. MICROVITEK LCCD	20"	colour	RGB	£287 (from the BBC)
6. LUXOR 3711	14"	colour	TV/monitor/RGB	£299
7. SONY KV1612UB	16"	colour	TV	£365
8. SALORA	24"	colour	monitor/RGB	£465

We are much obliged to Microage Electronics who loaned us the Zenith monitor, and to Portatel who loaned us the Luxor for review. Although most sets can be obtained in many places the suppliers for sets numbered 3, 5, 6 and 8 are given at the end of this article. We have negotiated a small discount on sets 3 and 6 for members from the suppliers listed.

The best colour pictures obtained when receiving broadcasts (ie using the set to watch 'The Computer Programme') came from the Sony, with the Luxor in close contention. The only B&W TV set we tested gave a reasonable picture, though it was not as crisp as more expensive black and white portables.

Results using the BBC Micro

The best picture using the UHF (aerial) input came from the Sony, this was followed closely by the Luxor, both these sets gave pictures with the colours running into one another, as if you were using water colours on blotting paper; but if you had never seen a monitor's picture then you would probably be quite pleased with the Sony, and accept the Luxor. The old 1972 GEC set didn't fair at all well,

it gave a very blurred picture, with none of the colour edges being distinct, and a noticeable instability in the picture. None of the colour sets gave very good results using the 80 column mode (MODE 0). The Sony television was unusable. The LUXOR using RGB gave a poor but just readable picture, which would undoubtedly cause eyestrain after a few minutes use. The Salora was a bit better. The Microvitek was a lot better and many would say acceptable. The B&W TV gave a poor image in 80 column mode. It would seem advisable that if you want to use 80 column mode for things like word processing, then you must buy a monochrome MONITOR such as the Zenith or APF, unless perhaps you were only doing an hour's word processing at a time. [Note that the Zenith has a green screen, so is green and white not black and white. Personally I prefer good old black and white]

The sets with RGB input gave by far the best pictures. The picture on the Microvitek being superior to the others. The Salora and the Luxor were blurred (out of focus) at the corners although this was not too bad. (It was staggeringly good compared with the ordinary TV pictures). The quality of the image shows most on 80 column text display, though if you study the screen carefully you can notice a difference between the quality of the displays of the different sets.

The two monochrome sets (APF and Zenith) were very good indeed. They would both be ideal for word processing. There was absolutely no noticeable distortion at the edges on either set. I have seen many other monochrome monitors working, but not on the BBC Micro, I should think that virtually any monitor would give an equally good display so look around at alternatives. If you have never seen a monitor display, then I suggest that you call into your nearest computer shop and see one in action.

Lastly, it should be said that the Luxor is one of the only sets that provides an ordinary (portable, it has its own aerial) colour TV with monitor and RGB facilities; and although not brilliant in any aspects it is good in most. I expect that if you haven't already got a colour TV and you were thinking of getting one, then this is certainly worth considering.

STOP PRESS: We have just been informed that Kingsley TV Services 40 Shields Rd, Newcastle upon Tyne (0632 650653) offer a range of monitors with screen sizes 14" to 26". These are Grundig television sets converted to be switchable between standard TV and monitor. It has separate RGB inputs for analogue and TTL. 1v video input and output is available. There is an interesting remote control facility between the computer and the monitor. Inputs are provided for both the BBC micro and the RML 380Z/480Z. Price for a 16" monitor is £248, teletext is available for an extra £50. BEEBUG has not seen or tested this range of monitors so cannot comment on them. However Kingsley TV Services has agreed to give BEEBUG members a special price of £238.50 (+VAT) provided they quote their membership numbers.

S.W.

Suppliers

SALORA
Gadney Electronics, 179 Torridon Rd
Catford SE6 (01-697 0079)

LUXOR
Portatel, Sunbury Cross Centre
Staines Road West, Sunbury-on-Thames
Middx. TW16 7BB (Sunbury-on-Thames 88972)

MICROVITEK
Direct from the BBC Micro suppliers
or Microvitek PO Box 188, Bolling Rd
Bradford BD4 7TU

ZENITH
Microage Electronics, 135 Hale Lane
Edgware, Middx. HA8 9QP (01-959 7119)

TV CONTROL

The operating system on the BBC micro supports an *TV command. This allows you control over certain video functions. When you look at a clear monitor picture it may appear to jiggle up and down very slightly. You can alter the interlacing with *TV 0,1 followed by a MODE change. This should then make the picture rock steady.

Another problem may be that your picture is too close to the top of the screen. This can be cured by *TV 255 or *TV 254 to move the picture down, or *TV 1 and *TV 2 to move it up. Any *TV command must be followed by a MODE change before the effect is noticeable. You may like to try *TV 240 and notice the effect when you scroll a large program. *TV 4 gives a software vertical hold!

To summarise - *TV a,b alters the TV image as follows:-

a=0 interlacing off
1 interlacing on (move display by one 625 line)

b=0 picture in standard position
1,2 picture down one or two lines
255,254 picture up one or more lines

S. W.

BEEBUGBUGSBEEBUGBUGSBEEBUGBUGSBEE **BLUNDERS** BUGBUGSBUNNYBEEBUGBUGSBEEBUGBUGSBEEBU

We have made some oversights and errors in issues 1 and 2. Many of these have been corrected in the reprinted version of issues 1 and 2.

Page 4 issue 2:

We said that a function cannot return a string value, only a numeric one. We have now discovered a way to do this. You assign the function as on line 30 without the dollar sign.

```
10 DIM A$(3)
20   REPEAT
30     NAME$=FNRAND
40     PRINT NAME$
50   UNTIL FALSE
60 END
```

```
1000 DEF FNRAND
1010   A$(0)="TOM"
1020   A$(1)="MARY"
1030   A$(2)="CYNTHIA"
1040   A$(3)="ERMINTRUDE"
1050 =A$(RND(4)-1)
```

Page 5 issue 2

Screen Addresses - We said "For the 16k machine you presumably subtract &4000 from all numbers". In fact there is redundancy in the addressing so that you need not make the subtraction.

Page 16 issue 2 and issue 1

We have published Roger Luff's telephone number incorrectly for two issues, it should be Kingswinford not Knutsford. Also he would like to be listed under the 'Dudley Area' rather than the 'West Midlands' area.

Page 14 issue 2 and p 15 issue 1

We gave commands to disable the cursor. Several people have written in saying that we have the syntax wrong. See "Cursor Control" elsewhere in this issue.

Page 14 issue 2

To combine the 4 bytes in the PEEK command it should read:

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

Corrections to the article "Merging Programs"

Our apologies for the mistakes in this article. Below is a reprint of the key passage:

FIRST	SECOND	
Create a file on tape of any program called "name", using:	Now load the program that you want to add to this. Then do the following:	
*SPOOL"name" ret	*EXEC"name"	
set recorder to record	press play on recorder	[ret means
press return	press return	press return]
LIST ret	ignore error messages.	
*SPOOL ret		
stop the recorder		

EEBUGBUGSBEEBUGBUGSBEEBUGBUGSBEEBUGBUGSBEEBUGBUGSBEEBUGBUGSBEEBUGBUGSB

GAMES WRITING PART 2

Last month, as well as giving a full listing of the games program "Bomber" we started an article which looked at the way the program was written, breaking the program down into its functional components (modules) and looking at these individually. This was done in such a way that you could enter the particular program lines under discussion, and build up the game element by element.

Here we complete the exercise, giving the rationale behind the crash detection (continued from last month), the bomb handling section, and very briefly the sound effects and instruction sections. Before we proceed further, there are two omissions from last months treatment, (corrected in the reprint of issue 2) one of which at least, I hope you were able to spot by checking against the full program listing. At the bottom of page 19 it should have said that the number 15 in line 3090 should be replaced by "K%". In case of further typographical errors, I should also add that capital letters have been used for all variables. There was also an omission that I fear may not have been spotted. There appears to be a bug in the POINT command, and it seems not to function correctly unless you have actually drawn something in the graphics mode before using POINT. It is for this reason that line 40 is included in the program. It draws a base line, and so somehow enables the POINT command to function. We apologise for not putting this into the first article (though it was in the complete listing). The reason why it did not go in was that we mistakenly thought that the bug only occurred when the cursor disable command was used. So before going any further, enter the following line:

```
40 PLOT 4,0,0: PLOT 5,1280,0
```

This should cause the plane to crash where appropriate.

We left the program last month with the insertion of line 210 which used the function FNCRASH to stop the program if the plane hit a tower. In practice we want to do much more than just stop everything: it would be useful to print some message or other, and ask if a new game is to be played. In order to do this we must exit the double FOR loop (initiated in lines 150 and 170) in a seemly manner. It is not good programming practice to just jump out with a GOTO... because this will make the machine think that it is still inside the loops, and it will leave return values on the stack, which if this is repeated enough times will cause trouble.

So to exit the loops in a hurried, but reasonably acceptable manner you can set the two loop variables (A% and B%) to values higher than the maximum allowed in the loop and wait for nature to take its course. Thus line 210 can be replaced with:

```
210 IF FNCRASH>0 THEN A%=50:B%=50
```

Another way of getting out of FOR loops is not to use them at all, and to replace them with REPEAT..UNTIL loops. This is a more elegant solution to the problem, but execution of the latter is much slower in the BBC machine, so we will stick with FOR loops.

Speed test to compare REPEAT-UNTIL with FOR loops:

```
 10 TIME=0
 20 A=1
 30 REPEAT: A=A+1
 40 UNTIL   A=1000
 50 PRINT TIME
 60 REM Using FOR loop
110 TIME=0
120 FOR A=1 TO 1000
140 NEXT
150 PRINT TIME
```

Returning to the "Bomber" program; with line 210 setting A% and B% to 50, the program now provides two possible routes to line 650. Either by a crash, or by a safe landing (after fully completing both loops). We can test for these two conditions by adding the following:

```
700 IF A%>40 THEN PROCSND ELSE PROCLANDING
```

since if the FOR loop has been exited naturally then A% will be less than 40. PROCSND and PROCLANDING are the names of two appropriate procedures to deal with the crash (producing sound and visual effects) and the safe landing. We will treat the definition of these later.

5. Keyboard, 6. Bomb dropping, 7. Tower obliterating.

The keyboard part of the program is relatively easy. All we have to do is to introduce a keycheck at one point in the centre loop. The INKEY$ is a suitable function to use, and this can take the form:

```
215 IF INKEY$(0)="B" THEN (drop a bomb)
```

The 0 in brackets means that the program will not actually wait for the key press but will just check for it at a given moment. So the effect of the line is just to drop a bomb when the 'B' key is pressed. In this game we only allow one bomb to be in the air at a given time, and to achieve this we will need to keep track of the state of bombs in play. This can be done by means of what is called a flag. We will use the variable F% for this, in such a way that if F%=0 it means that there are no bombs in play, and if F%=1 then one is being dropped.

We can extend line 215 as follows to incorporate this, and set up the bomb coordinates:

```
215 IF INKEY$(0)="B" AND F%=0 THEN V%=A%+2:W%=B%+1:F%=1
```

The first part checks to see if the flag is at zero, then the two coordinates V% and W% are set up for the bomb, (in relation to the position of the aeroplane) then the flag is set to 1, so that the program will know that a bomb has been released. V% is the initial Y-coordinate of the bomb, and is made two lines lower than the plane; and W% is the x-coordinate; this is made to coincide with the back half of the plane. The variables F%, V% and w% are initialised at line 35:

```
35 F%=0:V%=35:W%=0
```

We must now create a bomb character, and then cause it to be printed ever lower to the screen until it reaches the bottom.

The character can be defined by adding the following line to the PROCCHR procedure:

```
2040 VDU 23,226,0,24,24,24,60,60,24,0
```

This looks more or less like a bomb.
To drop it add the line :

```
250 IF F%=1 THEN P.TAB(W%,V%-1);SPC 1;TAB(W%,V%);CHR$226
270 V%=V%+1
```

This prints a space above the new bomb position (to erase the previous bomb image), and then prints the bomb at coordinates W%, V%. Then V% is updated. The program should now produce a dropping bomb when the 'B' key is pressed. However, as yet there is nothing to stop it falling below ground level. So we must detect when each bomb reaches ground level, stop them falling further, restore the flag to zero (to arm the next bomb) and erase the last printed bomb in the same way that we had to erase the planes printed on the far right of the screen. Line 270 can be lengthened to achieve this:

```
270 V%=V%+1:IF V%=31 THEN F%=0:PRINT TAB(W%,V%-1);SPC 1
```

The towers should now be automatically demolished whenever a bomb is on target.

8. Safe Landing

Line 700 already detects a safe landing, and points to a procedure called PROCLANDING that we have not yet written. So we just need to make it print some form of congratulation, and then ask if another game is required. To make things more interesting we could introduce skill levels into the game, and use these to produce a variable number of towers, and to change the rate of descent of the plane. We can call the skill level LE% and allow levels 1 to 5, these can be input at the start of the game, with a set of instructions. See the complete listing of the game for the definition of the procedure PROCSTART which gives instructions (lines 4500-4635) and PROCLEVEL which sets up the parameters for the skill level (lines 4640-4800). These procedures are called in line 20, and lines 20, 23 and 25 should now be entered. We can now delete line 15, which assigned a provisional value of 15 for the number of towers, and line 150 should be amended to step by S%, so as to allow the plane to descend by one or two lines depending on the skill level.

The definition of the safe landing procedure PROCLANDING is given in the full printout at line 4000, and includes both a printout of the skill level, and also of the game score (SC%). This is calculated as the number of clear passes of the plane after the last bomb has been dropped. SC% is initialised in line 35 to zero, so change 35 to:

```
35 F%=0:V%=35:W%=0:SC%=0
```

and is increased with each descent of the plane by adding line 630 SC%=SC%+1. SC% must also be set to zero in line 215 (see full listing), which is entered each time a new bomb is initiated.

Sound and other odds and ends

A simple sound has been added in lines 215 and 270 for the release and landing of the bomb. See the full listing for the commands, and the April issue of the mag for the full syntax of the SOUND command. When the plane crashes there is a crashing sound and a visual effect. These are produced in the procedure PROCSND which is already called in line 700, and which should be defined by adding lines 3500 to 3600.

Lines 800 to 1000 should now be entered. These offer the possibility of a new game after either a crash or a safe landing, and allow the skill level to be changed by calling up the procedure PROCLEVEL (defined earlier). Finally add line 50 to neaten up the display, and line 32 to delete the cursor. It is a good idea to insert the delete cursor command last of all since it can have unexpected side effects.

Conclusion

If you have followed the game through as presented here, you should find that you now have the full game on your machine. But it would be wise to check your final listing against the full listing given in the May issue. (This has been fully debugged, and printed out from the computer). In its present form, the program is a relatively short one, and there is much scope for building upon it. You could easily include a skill level which reduces the time wasting loop at line 220 for example. More interesting sound effects could also be created, and different visual effects too. Colour commands could be added, and the towers could even be made to fire back, since there is plenty of time for this in the program as soon as the waiting loop at line 220 is removed.

Finally some self criticism: There are two further ways in which this program should have been improved. 1. The main loop should have been structured as a procedure. 2. I should have made use of the 'long variable names' facility to keep the program legible. When doing this it is a good idea to use lower case, because then you do not need to worry about reserved words (eg the variable SINGLEBYTE will give a 'syntax error' because the interpreter thinks it is supposed to be SIN something. In lower case there is no problem). D.E.G.

D I S C N E W S

We have been informed by two separate companies that they are about to sell disc drives for the BBC Microcomputer. One company is about to offer a 3.5" drive that will cost around £100 to £150. The other company is to sell either a single 5.25" disc drive for about £175, or twin disc drives for £295.

I am afraid that we can tell you no more than that at present, except that we will review them at the earliest possible opportunity, and report our findings in the BEEBUG magazine. In the meantime, please do NOT write to us for details.

ASSEMBLER PART 2

Some of the material presented here is given in the full user guide - but we cover it here because we cannot be sure that everyone will get the full guide by the time our June issue comes out, and anyway a slightly different approach may prove useful.

P%

Last month we said that P% should be assigned a start address for assembling a program before entering the assembler proper - ie before the square brackets start. The value of P% is updated as assembly proceeds (and is printed in the left-hand column in hex unless you OPT out). If you wish to reserve a block of memory in the middle of a machine code program, you can do it by temporarily exiting the assembler, re-assigning the value of P%, and then re-entering the assembler. Thus:

```
100 DIM SPACE 100
110 P%=SPACE
120 [
130 first part of
      assembly program
200 ]
210 P%=P%+30
220 [
230 second part of
      program
400 ]
```

Data entry

It is just as important to be able to enter data into assembly language programs as it is using high-level languages, and the BBC Assembler provides two ways of doing this - both use a particular convention, and both require a temporary exit from the assembler. To enter the data values 10 and 14 as individual bytes at a given point in a program, exit the assembler by closing square brackets, then make the next line P%?=10:P%?1=14. These are effectively two variants on the POKE command, POKEing the value 10 to the address given by variable P%, and the value 14 to the address given by (P%+1). The following line should read P%=P%+2 so that the assembler program counter is moved on by 2 to avoid overwriting the data just entered. The assembler can then be re-entered using square brackets.

If, on the other hand, the data is in ASCII form, say the word "HULLO", then after exiting the assembler, you enter a line giving the following $P%="HULLO" and follow this with P%=P%+6 - since the assembler automatically inserts a carriage return character (0D hex), making 6 rather than 5 characters in all. The carriage return character may be deleted by only adding 5 to P% rather than 6, or it may be replaced with any other character using the modified POKE convention above. Thus the following line would replace the carriage return character (ASCII 13) with ASCII 0 - P%?5=0

This may sound a bit involved, so I will look at another example. Suppose we wish to have a data block containing the ASCII word "ERROR" followed by three 255s. The following would achieve this:

```
first part of assembler program
500 ]
510 $P%="ERROR"
520 P%?5=255:P%?6=255
530 P%?7=255
540 P%=P%+8
550 [
second part of assembler program
```

Here is one further note on assembler syntax that may prove useful. The assembler in the BBC machine uses standard symbols except that & (rather than $) is used to denote hexadecimal data, and a backslash \ is used to enter comments - note here that a colon will "re-activate" a line. Thus in the line:

```
10 LDA #5 \ First data loaded: LDX #10 \ second data
```

both LDA and LDX will be assembled. This is therefore different to the effect of REM in Basic which ignores the remainder of the line whatever it contains. D.E.G.

MINI WORD PROCESSOR 32k

It looks as if the BBC machine will make a rather nice word processor unit for handling letters and documents of all kinds. It has a high quality keyboard and an 80-column text mode - though you really do need a good quality black and white monitor to use it without too much eyestrain. As yet there is no commercially available word processing package for it, though one or two have been advertised. The best kind of word processor package will probably be one that resides in the machine in one of the paged ROM sockets (rather than having to be loaded each time from tape or disc). As we have said however, the 0.1 operating system does not support paged ROMs (though it might be possible to install an internal motherboard to get around this). Moreover if your operating system is in EPROM there are no free sockets anyway. We are told that the 1.0 O.S. will not be available in ROM for another 3 months or so, which means that there does not appear to be much point in buying a word processing package before then; unless it is disc or tape based.

As a temporary stop-gap I have written a very primitive word processor substitute. The program, as you can see is very short, but it does in fact work quite well. Essentially the text is stored in numbered lines of Basic, entered using the AUTO command, and the cursor editing features can be used to edit the text. It makes use of a number of the user-defined keys to neaten up operations. Note that in the program listing lines 950 to 1000 each contain just a few spaces. They are there to put a space at the top of the text.

To use the program, first type RUN (to RUN it subsequently key-0 can be used). This puts the machine into mode 3, and sets up an unscrolling page-width guide on the bottom of the screen. If you are writing a letter requiring the address at the top, begin by pressing F2, otherwise start with key F1 (which erases your address). Both F1 and F2 initiate the AUTO line numbering, you then press F9 and begin typing. Make sure that your text does not spill over onto the next line - if it does, use the delete key. Each time you have a line of text, press F9 (rather than the RETURN key). This tabs along a number of spaces to provide a margin on the next line. Key F6 produces a six space tab each time it is pressed, and F8 a tab suitable for entering addresses at the right-hand side of a letter. Key F7 simply switches over to mode 7 for clarity of reading if this should be required. When you have entered the text required, press 'Escape' to exit from the AUTO mode, and use the LIST command and cursor edit controls to edit the text. New lines are easily inserted, though it is not easy to insert odd words if there is no space for them on a line.

When the text has been edited, it may be printed out by pressing key F5 (to get double spaced printout, precede this with *FX 6,0 - *FX 6,10 returns things to normal). The line numbers can then be cut from the left hand margin with scissors. Text is saved with the entire program with a normal SAVE command, so that when it is loaded in again, the word processor is automatically loaded with it.

This very primitive word processor works reasonably well for modest tasks, though it could be usefully improved with some machine code patches to achieve the following:

1. Remove the line numbers on printout - in which case the left-hand margin could be shifted further to the left.
2. Give an audible cue when there are only 10 spaces left on a line.
3. Right justify the text.
4. Permit words to be inserted anywhere in the text.

Note, as this stands you must avoid capital letters followed by a full-stop, because the Basic interpreter thinks that they are abbreviations for Basic commands. Thus P.O. Box becomes PRINTOLD BOX. This could be remedied in a number of ways - eg key 9 could print the word REM on the far left-hand side instead of just a number of spaces.

In fact, we now have a routine that prints out the text without printing the line numbers, and we will give this next month. D.E.G.

```
  1 REM MINI TEXT EDITOR
  2 REM BEEBUG MAY 1982
 10 GOTO25000
950
960
970
980
990
1000
1010                                                    BEEBUG
1020                                                    PO Box 50,
1030                                                    St Albans,
1040                                                    Herts.
25000 REM MINI WORD PROCESSOR
25090 MODE3
25100 *KEY 0 "RUN |M"

25110 *KEY 1 "DELETE1000,1100 |M AUTO1000,10 |M"
25120 *KEY 2 "LIST1000,1099 |M AUTO1100,10 |M"
25150 *KEY 5 "VDU2 |M LIST100,10000 |M VDU3 |M"
25160 *KEY 6 "      "
25170 *KEY 7 "MODE7 |M"
25180 *KEY 8 "                                          "
25190 *KEY 9 "|M        "
25300 PROCWINDOW
26999 END
27000 DEF PROCWINDOW
27030 PRINTTAB(8,23)"I         I         I         I         I         I
 I        I"
27050 VDU28,0,22,79,0
27060 PRINTTAB(0,22);
27200 ENDPROC
```

DISCOUNTS

BEEBUG have arranged discounts for members at a number of retail outlets who supply computer books, software, hardware and services. We are at present negotiating further discounts.

CHEAP MEMORY UPGRADE

We have been shopping around to find the best price for memory upgrades, and have come up with a good offer from Watford Electronics. They will supply a set of eight '4816' memories for £16+VAT to BEEBUG members who apply (with membership number) before 15th July 82. After that date it will be £18.

Electronics Applied 4 Dromore Road Carrickfergus Co Antrim BT38 7PJ (Tel: 65973)	Cassette leads for BBC Machine 5% discount.
Happy Memories Gladestry Kington Herefordshire HR5 3NY (Tel: 054422 618)	Computer Hardware 10% off all 'one -off' prices. Quantity prices may be negotiable.
Midwich Computer Company Hewitt House Northgate Street Bury St Edmunds Suffolk (Tel: 0284 701321)	Hardware 10% discount on all items.
Mine of Information Ltd (Mail Order) 1 Francis Avenue St Albans Herts (Tel: 0727 52801)	Computer Books 5% discount

Technomatic Ltd 15 Burnley Road LONDON NW10 (Tel: 01-452 1500)	Hardware, software and books, 5% discount	Watford Electronics 33 Cardiff Road Watford Herts. (Tel: Watford 40588)	5% off all items, except special offer mentioned elsewhere

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 this). We are not acting for these companies, nor receiving payment from them, and cannot be held responsible for their services.

READ A CHARACTER

To read the ASCII value of a character at any text position on the screen: First set the cursor (even if disabled) to the position required, then execute OSBYTE 135. The ASCII value of the character appears in the X-register of the 6502.

Explanation

The machine operating system contains a number of useful subroutines that may be called separately by the user. Many of these are available as *FX calls (See April '82 newsletter). When values need to be passed back from a call of this kind, it is probably easiest to execute it via the USR call, which does allow data to be passed back (*FX calls do not).

To execute an OSBYTE call, you load the 6502 accumulator with the call number (135 in this case), and then execute a call to the address &FFF4 (which is indirected via &020A). The following program makes use of OSBYTE 135

```
10 PRINT"QWERTYUIOP";
20 VDU 8                    (ie a backspace)
30 A%=135
40 PRINT (USR(&FFF4) AND &FF00)/&100
```

It will work in any mode.

Line 10 simply prints some text on the screen leaving the cursor on a blank space, VDU 8 brings the cursor back one position leaving it on the 'P' (though you could TAB it to anywhere on the screen).

A% is then set to 135, USR(&FFF4) is then called, which puts the contents of A% into the 6502 accumulator and then executes the OSBYTE routine. The result is found by masking out the first, third and fourth bytes (with &0000FF00) and scaling this down by dividing by 256 (&100) to extract the contents of the 6502's X-register. - see Assembler guide article in this issue for further details. The answer will be 80, which is the ASCII value of the letter 'P'. We apologise for not explaining more fully how this works, but space is at a premium this issue.

USER GROUP INDEX & BEEBUG HAMS

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". Printed here are additions and corrections to last months list:

Peterborough
Tony Goodhew
Sir Harry Smith College
Whittlesea
Peterborough

Woking Area
John Gardner
22, St John's Rd
St Johns, Woking
Surrey

EDUCATIONAL SUB-GROUP
D.H.Maddock 18, Brian Avenue
Stockton Heath
Warrington
Cheshire

Dudley Area Roger Luff Kingswinford 288721 [NOTE: Correction of previous issues]

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.

Roger Cooke	G3LDI	P. Knight,	G3HGR	John Ray	G8DZH
K. Gee	GM4LNN	David Norris	G8HUP	David Sandy	G6EKV
S.W. Edwards	G8TMI	Ted Pratt	G4MID	R. C. Sterry	G4BLT
				John Yale,	G3ZTY

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.

MAZE TRAP

The object of the game is to bounce your ball out of the maze configuration using just two controls. The 'Z' and the 'M' key. The ball that bounces around can be deflected by means of these two keys; if it was moving up the screen its direction can be reversed by pressing 'Z', similarly if it were moving down the screen its direction can be reversed using 'Z'. Practice the game using only the 'Z' key until you have mastered exactly what it does.

The 'M' key is considered by some to be the more difficult of the two to master. It reverses the ball's horizontal movement. For example - if the ball were moving to the right and you pressed 'M' then it would immediately start moving to the left, and vice versa. Again, try playing the game with the 'M' key only until you have got the hang of it.

You can alter various parameters at the start of the program, for example line 75 NBARS%=25 controls how many vertical bars are in the maze. You can also change SCRHEIGHT% and SCRWIDTH%.

On many TV sets the game scores along the top are too close to the top of the screen. In the review of TVs and monitors elsewhere in this issue you will see line 65 explained. If your picture is too close to either the top or the bottom then adjust line 65 accordingly. I have also turned the interlacing 'on' using this statement, this makes the picture steadier on our set.

As with all the programs that we publish, we expect people to get as much satisfaction from studying and modifying them, as they get by playing them, so please experiment.

```
   10  REM      M A Z E - T R A P
   20  REM      -----------------
   30 REM             by
   40 REM          BEEBUG
   50 REM          26/V/82
   60 REM=========================
====
   65  *TV 255,1
   70  ON ERROR GOTO 8000
   75  NBARS%=25
   80  SCRWIDTH% =1270
   81  SCRHEIGHT%=950
   90    XINC%=5
   91    YINC%=5
   95  START%=5
   96  DBB%=SCRWIDTH% DIV NBARS%:
REM Distance between bars
   97  SCRWIDTH%=DBB%*NBARS%
   98  AVERAGE%=0:BEST%=9999:NGAM
ES%=0:LASTGO%=0
  100 CLS
  105  MODE 1
  110  PROCinstructions
  190  CLS
  191  VDU 23;11,0;0;0;0
  195  PROCheadings
  200  GATEWIDTH%=200
  250  PROCheadings
  300  NGAMES%=NGAMES%+1
  330  PROCdrawcourse
  400  Z%=RND(SCRHEIGHT%)
  410  PLOT 69,START%,Z%
  420  X%=START%
  430  Y%=Z%
  440  B%=XINC%:C%=YINC%
  450  TIME=0
  999  REM----------------------
-------
 1000  KEYPRESSED$=INKEY$(0)
 1020  IF KEYPRESSED$="M" THEN B%
=-B%:GOTO 1040
 1030  IF KEYPRESSED$="Z" THEN C%
=-C%
 1040  PRINT TAB(1,1);TIME DIV 10
0
 1100  D%=X%:E%=Y%
 1110  X%=X%+B%:Y%=Y%+C%
 1115  colour%=POINT(X%,Y%)
 1120  IF X%>=SCRWIDTH% THEN 1330

 1125   IF colour%=0 THEN PLOT 69
,X%,Y%:PLOT 71,D%,E%:GOTO 1000
 1130  IF colour%=2 THEN X%=D%:B%
=-B% ELSE Y%=E%:C%=-C%
 1240  SOUND 1,-15,200,1:GOTO 111
0
 1301  REM----------------------
-------
 1330  F%=1
 1380   SCORE%=TIME DIV 100
 1390  PROCendsound
 1400  IF SCORE%<BEST% THEN BEST%
=SCORE%
 1410  LASTGO%=SCORE%
```

```
 1500  AVERAGE%=((NGAMES%-1)*AVER
AGE%+SCORE%)/NGAMES%
 1510  GOTO 195
 2000  DEF PROCdrawcourse
 2005  GCOL 0,1
 2010  MOVE 0,0
 2015  PROCBOX(0,0,SCRWIDTH%,5,1)
 2017  GCOL 0,2
 2020  PROCBOX(SCRWIDTH%,0,SCRWID
TH%+5,SCRHEIGHT%,1)
 2025  GCOL 0,1
 2030  PROCBOX(SCRWIDTH%,SCRHEIGH
T%,0,SCRHEIGHT%+5,1)
 2032  GCOL 0,2
 2035  PROCBOX(0,0,5,SCRHEIGHT%,1
)
 2040  REM Put in gaps
 2050  FOR X%=DBB% TO SCRWIDTH% S
TEP DBB%
 2052    Z%=RND(SCRHEIGHT%-GATEWI
DTH%)+2
 2055    FOR I%=X% TO X%+5
 2080      MOVE I%,1:DRAW I%,Z%
 2085      MOVE I%,Z%+GATEWIDTH%:
DRAW I%,SCRHEIGHT%
 2090      NEXT
 2120    NEXT X%
 2999  ENDPROC
 3000  DEF PROCinstructions
 3010  CLS
 3020  PRINT TAB(12,10);"M A Z E
 - T R A P"
 3030  PRINT TAB(12,11);STRING$(1
7,"-")
 3040  PRINT TAB(0,13);
 3050  PRINT"Find your way from t
he left of the      screen to the
 right."
 3060  PRINT'"Use the  'Z'  and '
M'  keys to control  the bounce a
s follows:-"
 3070  PRINT
 3080  PRINT"  Z - reverses  VERT
ICAL  movement"
 3090  PRINT"  M - reverses HORIZ
ONTAL movement"
 3095  PRINT ''
 3100  PRINT'"Press 'space bar' w
hen ready ";
 3105  REPEAT
 3110    Q$=GET$
 3120    UNTIL Q$=" "
 3999  ENDPROC
 5000  DEF PROCheadings
 5010  LOCAL Y%
 5015  Y%=0
 5017  PROCBOX(0,0,SCRWIDTH%+5,SC
RHEIGHT%+5,0)
 5020  PRINT TAB(0,Y%);"TIME"
 5030  PRINT TAB(7,Y%);"No of Goe
s"
 5040  PRINT TAB(19,Y%);"AVERAGE"
 5050  PRINT TAB(27,Y%);"BEST"
 5060  PRINT TAB(33,Y%);"LAST GO"
 5100  PRINT TAB(0,Y%+1);SPC6
 5110  PRINT TAB(10,Y%+1);SPC1;NG
AMES%;SPC1
 5120  PRINT TAB(21,Y%+1);SPC1;AV
ERAGE%;SPC1
 5130  PRINT TAB(26,Y%+1);SPC2;BE
ST%;SPC3
 5140  PRINT TAB(33,Y%+1);SPC3;LA
STGO%;SPC1
 5999  ENDPROC
 7000  DEF PROCBOX(X1%,Y1%,X2%,Y2
%,F%)
 7005  IF F%=0 THEN F%=87 ELSE F%
=85
 7010  MOVE X1%,Y1%
 7020  MOVE X2%,Y1%
 7030  PLOT F%,X2%,Y2%
 7040  MOVE X1%,Y2%
 7050  PLOT F%,X1%,Y1%
 7999  ENDPROC
 8000  REM Come here if 'escape'
pressed
 8001  REM-----------------------
-------
 8010  MODE 7
 8020  END
 9000  DEF PROCendsound
 9010  FOR X%=1 TO 20
 9020    SOUND 1,-12,30,1
 9030    SOUND 1,-12,100,1
 9040    NEXT X%
 9050  ENDPROC
>,100
```

VERIFY

We have been asked by several members about verifying tapes. The provisional user guide suggests that you use the *CAT command to verify programs, since this checks that blocks and headers are correctly recorded. One problem with the *CAT command is that it never positively tells you if your recording is OK, it only tells you if it isn't. This is not totally satisfactory from a psychological point of view, but Acorn's technical staff have advised us that it provides a complete check on the tape.

However, it is obviously not a proper VERIFY in the sense that it does not compare a program byte by byte with what is in memory (since you can perform *CAT

even when there is no program in memory). And immediately after Acorn's assurance, we got graphic proof that our suspicions were not ungrounded. A long program file passed the *CAT test perfectly, but refused to load at all. All blocks and headers showed faultlessly, but the machine refused actually to recognise and load it, it simply searched through it. The only advice we can now offer is to save two copies of any program, and to do a *CAT on at least one of these.
We would strongly urge Acorn to include a full VERIFY command in their next version of the operating system.

CURSOR DELETE

We gave two VDU 23 commands to turn off the cursor in the May issue (and one in the April issue). Though these work, they can cause problems when further VDU 23 calls are made, since we got the syntax wrong.

We have since discovered that all VDU 23 calls must be followed by 9 bytes of data (as in the character redefine call), but that a semicolon signifies that two bytes of data follow on, and a semicolon at the end acts like an extra zero.
Here are two pairs of cursor commands:

Method 1

```
VDU 23;8202;0;0;0;     cursor off (all modes)
VDU 23;29194;0;0;0;    cursor on (mode 7 only)
                       (discovered by Ray Skinner
                       of Cleveland and others)
```

Method 2

```
VDU 23;11,0;0;0;0
     cursor off (all modes)
VDU 23;11,255;0;0;0
     cursor on (all modes)
```

Method 1 conveniently reinstates a block cursor when the cursor editing keys are pressed, but the reinstate cursor command that goes with it is dependent on the graphics mode chosen. The reinstate command given is for mode 7. Method 2 does not return the cursor when the editing keys are used, (making editing difficult), but its cursor reinstate command is not mode dependent. With both methods the cursor is automatically reinstated when the graphics mode is changed.

PROCEDURE ANOMALY

BBC Basic implements procedures differently to most other languages. This may be a bug that will be cured in future versions, but it is annoying and severely restricts their use.

A value passed into a procedure as a parameter, can have its value changed within the procedure as one would expect, but upon exit the original value is restored, rather than the new value being passed out for use in the main body of the program.

For example: suppose we want to find the area of a triangle given three sides and also calculate its perimeter, then the following procedure should do just that:

```
1000 DEF PROCarea(A,B,C,AREA,PERIMETER)
1010 LOCAL S
1020  PERIMETER=A+B+C
1030  S=PERIMETER/2
1040  AREA=SQR(S*(S-A)*(S-B)*(S-C))
1050 ENDPROC
```

When this procedure is called in the main program as PROCarea(3,4,5,SIZE,PERIM) the variables SIZE and PERIM should be assigned the values 6 and 12 respectively. This is a fundamental use for a procedure, and it is used often. However BBC Basic won't allow you to pass the values back to the main program.

You can of course avoid putting the last two parameters into the procedure definition or call, but then you always have to use the same variables in your main program ie AREA and PERIMETER, which you may not always want to do.

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:.....

This (and previous application forms) is also valid for the September software competition.

IF YOU WRITE TO US

Membership applications, subscriptions queries, Magazine back issues and software library.
Send to:

BEEBUG
Dept 1
374 Wandsworth Road
LONDON
SW8 4TE

If in difficulty
tel: 01-720 9314

For back copies, please give your membership number, and send an SAE plus 80p per issue required.

Membership costs £4.90 for 6 months (5 issues)
£8.90 for 1 year (10 issues)

The magazine is published each month except August and December.

Our editorial address for newsletter contributions is:

The Editor
PO Box 50
St Albans
Herts
AL1 2AR

Please do not send subscription letters to the editorial address, this will delay things somewhat.

NEXT MONTH

Next month's newsletter is due out in mid-July. We will not be publishing a newsletter during August. In our July issue we will be continuing with our articles on: "Graphics" which will include details of Mode 7 (Teletext graphics); and "The User Port" which will deal with data output. We will be reviewing software, showing how to dump the contents of the screen to an Epson printer, and have articles on 'Logic' and 'Screen manipulation'.

We shall of course, have more programs and members contributions, together with the ever-popular "Hints & Tips".

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