BETA BASIC NEWSLETTER No. 6

****

SUBSCRIPTIONS

The vast majority of Newsletter subscriptions are now due for renewal, so unless there is a slip with this Newsletter (headed "NOT YOU!") or you have paid already, please forward your payment or I will assume you do not wish any more issues. The price is the same as before - £5.00 for 6 issues in the U.K., £5.50 in Europe, and £6.00 elsewhere. I hope you will stay with us!

****

MUSINGS ON THE SPECTRUM 128 AND BETA BASIC 4.0

If you got this Newsletter when it was published, rather than as a back issue, you will have received a sheet of details on Beta Basic 4.0, telling you what it can do. However, I thought that some of the details behind the programming might be of interest.

The memory arrangement of keeping most of Beta Basic in main memory, with only 6K of code in paged RAM, was originally partly a matter of necessity, since it avoided a radical re-write of 18K of code. In hindsight, it now appears the best possible arrangement. It means that only 1K is taken from the available RAM disc space (5K of code is located in unused paged RAM) allowing individual 64K arrays to be created in the RAM disc with space left over. It is quite easy to load in extra program sections more or less instantly, too.

Once I got started on the programming I got quite interested, and the program grew to something much more ambitious than I had in mind originally. I added several features after the program was "finished", while writing the manual. (I think this is called the "wouldn't it be neat if.." syndrome, in the States.) Eventually deadlines (such as late Newsletters!) forced an end to additions. But I still have lots of unused ideas. Let's see how this version sells!

The 128K Spectrums allow another area of memory in the top 16K to be used as screen memory, and I spent some time playing with ideas to use this feature. (It is NOT possible to switch a new memory block into the normal screen memory area, which would be more useful. It would avoid the special PLOT, PRINT, DRAW etc. routines you would need in order to use the top 16K.)

I thought that by switching between the two screen areas thousands of time a second, attributes could be smoothly blended to give many new colours. Then I realised that an individual pixel is only displayed 50 times a second; the best you can do is to alternate 2 colours 25 times a second - and this gives a very poor effect. I still haven't found a use for altering the screen location; I think it must be for running CP/M. Cartoon animation is better achieved by saving successive frames on the RAM disc.

The sound command got more and more complicated - the sound chip can do lots of things, and you need a parameter for each one. Things sometimes go wildly wrong while they are being tested, of course. Interrupt-driven routines are particularly

interesting at this stage. Several times when I thought a test had finished was reading my notes, the computer interrupted with a rude chorus of buzzes, burps, chirps and hisses. I never managed anything as interesting when designing sound effects!

The division of the paged RAM into 16K blocks is inconvenient, since you are always having to check that you haven't "fallen off" the end of the block. SORT was a particular pain. (Is the highest string in this block? Does it straddle two blocks? Does the top-of-the-list string straddle two blocks? etc. etc.) However once something runs correctly using five 16K blocks, it is fairly simple to get it to run using fifty 16K blocks. It would be nice if a 256K machine came along - RAM is cheap enough. At the moment I can SORT all the Newsletter subscribers as one array, but I hope to run out of memory eventually. Even inconveniently paged RAM is *much* faster than even a hard disc. RAM discs are going to become more and more important in the future.

I mentioned in the last News letter that there is a minor bug in the 48K 12 Spectrum ROM which allows ordinary strings to be saved as DATA, but not loaded back correctly. I also mentioned that bug fixes to complex, programs are famous for causing almost as many problems as they correct. By coincidence, the only bug in the ROM that was altered in the new 128K ROM was this one-and it is now different, but no better. You can still SAVE normal strings, but when you LOAD back a string or array, the type is always set to normal string, unless an array with the same name already exists. So if you save and reload a normal array, it is converted to a very long string with a few junk characters (the array dimensions) at the front. The solution is to DIM an array with the same name (e.g. DIM a$(1)) before loading the array from RAM disc or tape. I cannot crow, though one of the bugs I corrected in Beta Basic 3.0 in early 1986 was the lack of a "Failed" report when RENUM finds LIST with an expression. Unfortunately, TURTLE contains LIST with an expression, and the TO procedure uses RENUM. This spoils the display (harmlessly) when the TO procedure is used, unless you POKE 62356,205 to prevent it.

****

A GRAPHICAL DOODLE

This little routine produces random patterns of horizontal and vertical lines.

```
100  OVER 1
110  LET x=0
120  DO
130     LET  y=RNDM (175)
140     DRAW TO x,y
150     LET x=RNDM(255)
160     DRAW TO x,y
170  LOOP
```

If you are using Beta Basic 4.0, you can use the whole screen if you alter line 130 like this:

```
130  LET y=RNDM(l91)-16
```

****

EVOLUTION

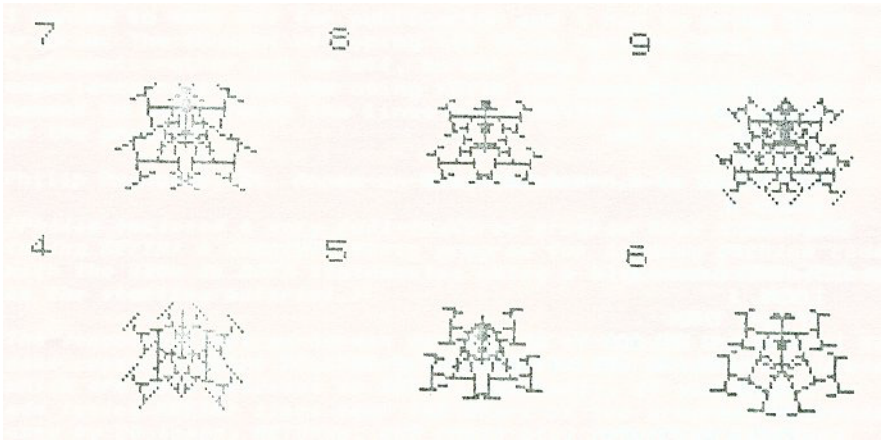
This contribution is from Steve Drain (Portland, Dorset). It is based on the ideas of Richard Dawkins, the biologist who wrote "The Selfish Gene" and "The Blind Watchmaker". The programming is completely original, and the "genes" used here are certainly not the exact ones used by Dawkins, but the results are similar Strictly speaking, the program is a simulation of selective breeding, rather than natural selection and evolution. You set up the initial values of nine genes. These are used is a starting point to produce nine "mutants" (one for each gene) from which you select one to survive and provide the basis of the next generation. Gradual, part-directed and part-rand changes occur.

The serious point behind the computer fun is this: A completely random process, such as mutation, would take forever to produce even a simple structure, if left to itself. But when *selection* is included, either by a human, or by better-adapted mutants producing more offspring (natural selection) complex structures can evolve much more readily. The illustration shows some of my attempts at "breeding" a fly shape. The gene values were -5,2,3,-4,5,7,3,1,5. It is surprising the variety you can get from just 9 genes.

Comment: The program is fairly slow, particularly if gene 9 has a large value. I suggest you run it while doing something else, a BEEP will tell you when a generation is finished.

Some genes can be negative. The property controlled by each gene is given below:

1 stalk length	4 initial angle	7 length up
2 colour	5 angle up	8 length down
3 initial length	6 angle down	9 depth of recursion



```
5     evolution
10    DEF PROC evolution
        LOCAL g(),r(),i,au,ad,ua,lu,ld,ul,lf,extinct
        creation
        DO
          reproduction
          development
          selection
        LOOP UNTIL extinct
        PRINT "The line has just died out"
      END PROC
20    DEF PROC creation
        DIM g(1,9)
        DIM r(1,9)
        FOR i=1 TO 9
          INPUT "Enter value for gene "; (i);" : ";g(1,i)
        NEXT i
        LET xrg=512,yrg=352
        LET ul=1,lf=5,ua=PI/12
      END PROC
30    DEF PROC reproduction
        CLS
        PRINT "G-";LENGTH(1,"r()");
        FOR i=1 TO 9
          PRINT TAB 3+i*3;g(1,i);
          COPY g(1) TO g()
          LET g(i+1,i)=g(i+1,i)+SGN (RND-0.5)
        NEXT i
        JOIN g(1) TO r()
      END PROC
40    DEF PROC development
        FOR i=1 TO 9
          LET xos=90+170*MOD(i-1,3),yos=40+110*INT ((i-1)/3)
          PLOT -90,40;STR$ i
          LET ik:=MOD(g(i,2),8),au=g(i,5)*ua,ad=g(i,6)*ua,lu=g(i
          ,7)*ul,ld=g(i,8)*ul
          PLOT INK: ik;0,0
          DRAW INK ik;0,g(i,1)*lf
          growth 0,g(i,1)*lf,g(i,3)*lf,g(i,4)*ua,g(i,9)
        NEXT i
      END PROC
50    DEF PROC growth x,y,l,a,n
        LOCAL dx,dy
        IF n THEN
          LET dx=l*SINE(a),dy=l*COSE(a)
          PLOT INK ik;+x,y
          DRAW INK ik;+dx,dy
          PLOT INK ik;-x,y
          DRAW INK ik;-dx,dy
          growth x+dx,y+dy,l-lu,a+au,n-1
          growth x+dx,y+dy,l-ld,a-ad,n-1
60    END PROC
70    DEF PROC selection
        PRINT #1;"Select Survivor (1-9)"
        BEEP 1,1
        DO
          GET i
        LOOP UNTIL 1<=i AND i<=9
        COPY g(i) TO g()
        DELETE g(1 TO 9)
        LET extinct=NOT g(1,9)
      END PROC
```

****

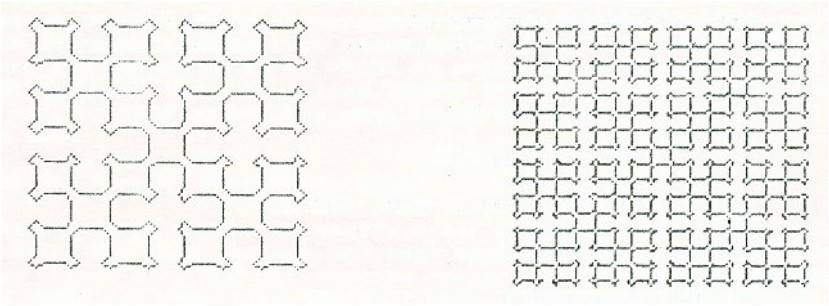
SIERSPINSKI CURVES

This contribution is from Ettrick Thomson (Aldeburgh, Suffolk), He writes:

"The program is quite a sophisticated one. It draws the Sierspinski Curve of order n; this is one of the Peano Curves, which are continuous "curves", but as n tends to infinity, change direction everywhere (or, if you like, have corners everywhere); they traverse every point of a square."

"The recursive procedure SIERT is interesting, for the parameter Q$ is effectively a procedure call; your KEYIN allows the call to be very simply executed."

Below are the curves of order 3 and 4.



The program requires some procedures from "TURTLE" to run. Load your copy of "TURTLE" (if you can find it!) and type:

```
QUICKTURILE
```

Then add these lines:

```
5     REM DRAW SIERSPINSKI CURVE
10    INPUT "DEPTH (>=1):";N
      PRINT N
20    PQ
      SETPOS -32,-32
      SETH 225
      PD
30    LET L=32/2↑N,M=2*L
40    SIERT N,"A"
      SIERT N,"A"
      SIERT N,"A"
      SIERT N,"A"
50    STOP

100   DEF PROC SEQA
        RT 90
        FD L
        RT 90
        FD L
        LT 45
        FD M
        LT 45
        FD L
      END PROC
```

```
200  DEF PROC SEQB
       LT 45
       FD M
       LT 45
       FD L
       LT 45
       FD M
       LT 45
       FD L
     END PROC

300  DEF PROC SIERT N,Q$
310    IF N=1 THEN
         KEYIN "SEQ"+Q$
       ELSE
         SIERT N-1,Q$
         SIERT N-1,"A"
         SIERT N-1,"B"
         SIERT N-1, "A"
320  END PROC
```

****

BIT MANIPULATION - SETTING, RESETTING AND TESTING BITS

To my machine-code saturated brain, these procedures look comfortably familiar. They might be useful in character design, or in storing information compactly (8 yes/no items per byte!). In the usual convention, the bit order is 76543210. Line 10 demonstrates setting and resetting of bits; line 20 does the same for bit testing.

```
10   LET x=0
     DO
       INPUT "bit to set:";b
        set b,x
       PRINT BIN$(x)
       INPUT "bit to reset:";b
        reset b,x
       PRINT BINS(x)
     LOOP
20   LET x=RNDM(255) P
     PRINT BIN$(x) ``
     FOR b=7 TO 0 STEP -1
       PRINT FN b(b,x)
     NEXT b

100  DEF PROC set b, REF num
       LET num=OR(num,2↑b)
     END PROC

200  DEF PROC reset b, REF num
       LET num=AND(num,255-2↑b)
     END PROC

300  DEF FN b(b,n)=AND(2↑b,n)<>0
```

****
BIG CLOCK - a giant--size digital clock

These line produce a giant version of the normal CLOCK function, which looks fairly impressive. Line 20 is used to make the area used for the normal CLOCK output have black INK attributes, matching the PAPER colour we have set for the whole screen. The normal CLOCK is therefore invisible, but it still can be read be by GET, and reproduced at a larger size by PLOT.

```
10   BORDER 0
     PAPER 0
     INK 7
     CLS
20   PRINT AT 0,24; INK 0;"  "
30   CLOCK 1
40   DO
50     GET a$;192,175,8,1
60     PLOT CSIZE 32,40;0,100;a$
70   LOOP
```

****
PROC short improvement - truncating strings from arrays

Keith Davies (St. Albans, Herts.) writes:

"I have found that PROC short (issue 5) is very slow for some applications. This is my suggestion which uses the very fast INSTRING. It can be used provided the string array elements do not contain two or more embedded blanks. Of course, the number of blanks searched for can always be increased as required."

Notes: Each set of quotes in line 9000 contains two spaces. I've included line 10 to demonstrate the speed improvement.

```
10   DIM a$ (63)
     LET a$="test"
     short a$,b$
     PRINT

9000 DEF PROC short x$, REF y$
       LOCAL L
       LET L=INSTRING(1,x$+"  ",  ")-1
       LET  y$=x$(1 OR L  TO L )
     END PROC
```

****
Improved PROC centre

Robert Dickson (London) sent this improvement to PROC centre (issue 3) which uses BB system variables (see issue 5) so that it works with different CSIZEs.

```
10   LET a$="Hello World!"
20   centre a$,2
2000 DEF PROC centre a$,x
       LOCAL cpl
       DEFAULT x=(175-PEEK 57365)/PEEK 57371
       LET cpl=PEEK 57391
       PRINT AT x,(cpl-1-LEN a$)/2;a$
     END PROC
```

****

PRINT AT/PLOT COORDINATE CONVERSION

Robert Dickson also sent these functions to convert between the PRINT AT and the PLOT coordinate system.

```
DEF FN b(y)=INT ((175-y)/8)
DEF FN h(y)=175-y*8
DEF FN g(x)=TNT (x/8)
DEF FN j(x)=x*8
```

FN b(y) converts from PLOT to PRINT AT (vertical axis).
FN h(y) converts from PRINT AT to PLOT (vertical axis).
FN g(x) converts from PLOT to PRINT AT (horizontal axis).
FN j(x) converts from PRINT AT to PLOT (horizontal axis).

Remember that you also need to reverse the order of the two coordinates.

****

ATTACK OF THE NUTANT LETTERS

The first program I ever sold was something like this, written in machine code for the 1k. ZX81, called IMPACT. This version isn't as good, but I hope it will amuse some of you. You control your "ship" with the cursor left and right keys; alter lines 60 and 70 if you want to change this. Avoid the "asteroids" and beware - they get pretty fast!

```
10   OVER 2
20   BORDER 6
30   LET c=16,p=c
40   FOR s=1 TO 16
50      FOR n=1 TO 50
60         LET p=p-(INKEY$=CHR$ 8 AND p>0)
70         LET p=p+(INKEY$=CHR$ 9 AND p<31)
80         FOR t=0 TO s/2
              PLOT RNDM(247),175;"O"
           NEXT t
90         PLOT OVER 0;c*8,8;" "
100        PLOT p*8,7+s;"A"
110        SCROLL 6,s
120        IF SCREEN$ (21,p)<>"A" THEN FOR t=1 TO 6
              PLOT INK 2; OVER 1;p*8,7;"*"
              BEEP .03,t
           NEXT t
           PLOT OVER 0;p*8,7;" "
           FOR t=-10 TO -40 STEP -1
              BEEP .03,t
           NEXT t
           RUN
130        LET c=p
140     NEXT n
150  NEXT s
160  FOR n=1 TO 11
        SCROLL 6,16
     NEXT n
170  PRINT CSIZE 16;AT 5,:;"YOU MADE IT!"
```

****

BETA BASIC AND THE DISCIPLE DISC INTERFACE.

The Beta Basic 4.0 manual includes instructions on how to use this disc interface with Beta Basic. You need to make some alterations to Beta Basic's code, and the disc "system" file.

Below are POKES to remove control of certain commands from Beta Basic. You will lose Beta Basic features like slicer and variable SAVES. The original values are given in brackets - they can be POKEd back to restore control.

```
SAVE: POKE 64844,223: POKE 64845,26: (45,237)
LOAD: POKE 64826,224: POKE 64827,26: (42,237)
CAT,FORMAT,MOVE,ERASE,OPEN,CLOSE,MERGE,VERIFY:
POKE 64609,215: (207)
```

You need to make a modified copy of the Disciple's "system" file. Have ready a disc with at least 35K of free space. Initialise the disc system using RUN, then LOAD Beta Basic from tape. Type in and RUN the program below. Do NOT include a CLEAR statement. The program searches the system file in memory for particular bits of code and then alters them. The modified file is then saved as "bbsys".

```
10   LOAD dl"system"CODE 33000
20   LET x=INSTRING(1,MEMORY$()(33000 TO 39143),"!"+CHR$
     244+CHR$ 27)
3O   POKE X+33001,252
40   LET x=INSTRING(1,MEMORY$()(33000 TO 39143),"!"+CHR$
     125+CHR$ 27)
50   POKE X+33001,252
60   LET x=INSTRING(1,MEMORY$()(33000 TO 39143),"!"+CHR$
     73+CHR$ 19)
70   POKE X+33000,75: POKE X+33001,251
80   SAVE d1"bbsys"CODE 33000,6144
```

You can now delete these lines. MERGE from tape a copy of Beta Basic's line 1 and 2. Modify them to the Disciple syntax you might want to alter "Beta Basic" in line 1 to "Autoload". Delete the final DELETE statement in line 2. Include just before RAND USR 58419:

```
LOAD dl"bbsys"CODE 0
```

Add a line:

```
3 DELETE 1 TO 3
```

Now RUN to save the program to disc.

The Disciple interface turns off Beta Basic when a disc error such as "File not found" occurs. It can be turned on again by RANDOMIZE USR 58419.

****

A FAST JOYSTICK CURSOR

Jan Biemans (Zoetermeer, Netherlands) sent me a JOYSTICK procedure listing which he wondered if I could speed up. I was fairly successful; the procedure below will let you move the cursor from one side of the screen to the other in about 2 seconds. You could control the cursor from the keyboard if you like. Because IN is used to read the keyboard or joystick, the cursor will respond to combinations of inputs that specify diagonal movements.

Line 40 sets the default limits the cursor will be allowed to move in, to be the whole screen. (The limits are in the order left, right, top, bottom, and they are in the PRINT AT coordinate system.) You can set smaller limits by including them in the procedure call. (e.g. CURSOR 10,20,5,15)

Line 50 creates a string of 8 characters whose CODEs match the INed values for left, right, down, up, left+up, right+up, right+down, and left+down. I found the values by plugging in a joystick and printing the INed values while moving it about; you may have to alter them for your own joystick. (You may have to alter the port looked at by IN, too.)

Line 60 sets up the initial cursor position.

Lines 70 to 90 are the main control loop; the string we set up earlier is scanned rapidly by INSTRING to see if the value from the port matches a direction code. This gives a number for ON to work on; "no match" means that the LOOP just after ON is executed, and we loop back to look at the port again. If a match is found, the required single statement in the ON list is executed and the position variables are adjusted to move the cursor. Note: I have been naughty and used two LOOPS for one DO, which throws out the indentation a bit. Well, it works fine and I leave you to re-write it if you like!

```
1     REM KEYS 7/8/9/0=UP/DOWN/LEFT/RIGHT

10    cursor

20    DEF PROC cursor l,m,t,b
30       LOCAL r,c,p,q,a$
40       DEFAULT l=0,m=31,t=0,b=21
50       LET a$=CHR$ 189+CHR$ 190+CHR$ 187+CHR$ 183+CHR$ 181
         +CHR$ 182+CHR$ 186+CHR$ 185
60       LET c=15,r=10,p=c,q=r
         PRINT AT r,c;"*"
70       DO
80          ON INSTRING(1,a$,CHR$ IN 30)+1
            LOOP
            LET p=c-(c>l)
            LET p=c+(c<m)
            LET q=r+(r<b)
            LET q=r-(r>t)
            LET p=c-(c>l),q=r-(r>t)
            LET p=c+(c<m),q=r-(r>t)
            LET p=c+(c<m),q=r+(r<b)
            LET p=c-(c>l),q=r+(r<b)
90          PRINT AT r,c;" ";AT q,p;"*"
            LET r=q,c=p
         LOOP
      END PROC
```

****

READERS' LETTERS

Dear Andy,

First of all I want to introduce myself: My Christian (or Jewish) name is Eloi, the others are family names we use in Spain. I am a Catalan, 45 years old, Industrial Engineer working in Informatics at Honeywell Bull. Receive my sincere congratulations for your Newsletters, which are as good as Beta Basic is... I want to pose two questions:

PEEKing 47272 (version number) says 39, while in the Newsletter you speak only of 21! Have I a future version of Beta Basic?

I want to pass some character arrays, with numbers converted by function CHAR$, to a TRL disc file. But to write the record I must use PRINT# stream-num;record-num,record-data and it fails with characters whose code means ENTER, PRINT options, etc. Is there any manner to force PRINT to write bytes without interpreting them? (Something like TOKENS OFF in printer interfaces.) If not, is STR$ number the most space-spare conversion that I Can use?

Eloi Gil Benito, Barcelona, Spain

I made a large number of mostly minor changes to Beta Basic 3.0 in March 86. They include any bugs mentioned in the Newsletter, plus things like MOD not working correctly with negative numbers and RENUM not reporting "Failed" when it encounters LIST with an expression. It is a good idea not to perform any bug fix described in these Newsletters unless you observe the bug - it may have been fixed already in your copy, and you say cause a problem.

The code in control of a particular stream determines what happens to any characters PRINTed to that stream. For a disc file, the code will be quite different from that for screen or printer output, so I cannot work out a general solution by modifying PRINT. I face a similar problem transmitting files from my Amstrad 6128 to my Spectrum on the RS232 link. The fastest method is to PRINT every byte as CHR$ (PEEK n) but certain values are interpreted by the Amstrad as control codes, so I have to send e.g. 1-6-0 instead of CHRS 160. For your application, the most space-saving method of coding numbers as strings depends on the range of numbers. Avoiding CHR$ 0-31 allows you to code numbers 0 to 223 as CHR$ (number +32.), using one character instead of 1-3. Numbers 0 to 50175 can be coded as characters in that range, and so on.

Dear Dr. Wright,

I sincerely hope that you are not intending to discontinue the Newsletter. Personally, it is the magazine that I most look forward to reading and I would happily pay more for a regular and/or more frequent issue, but I get the impression that you find it a bit of a nuisance to write!

Ian Charles, Shrewsbury, Shropshire

I intend to keep running the Newsletter for at least six more issues - there are a large number of eager subscribers. I get a lot of fun out of editing it, but it is quite a lot of work too! I am sorry if the irregular delivery is annoying. This is partly because I often work on other projects with deadlines and penalty clauses, which take longer than but also because it takes 2 or 3 months to accumulate the in pages of material for a Newsletter. (I originally thought in terns of a 6-page issue.) I could cut the length, I suppose. In this issue I am including illustrations that may make the Newsletter easier to fill, as well as more interesting.

Dear Dr. Wright,

I am doing part time research for a higher degree at the Open University and have found BB very useful for program development even though, since I have some massive data arrays, it is not present when the programs are run. Because of the research I have less time available for massing about than I would like. I have several goodish ideas that are dormant. Most of them are relating to operating systems for photoelectric photometry - I am a serious amateur astronomer, but a Civil and Structural engineer by profession.

Andrew Hollis, Cuddington, Cheshire

Your array problems would be solved by BB 4.0 and a 128K Spectrum (allowing 64K arrays). I take it NUMBER and CHAR$ cannot be used to reduce the array size - perhaps you require better precision than they supply? BB 4.0 allows coding of floating point numbers as 3, 4 or 5 character strings, according to the desired precision. If there is a demand, I could include a BB 3.0 "add-on" version of the new function in a future Newsletter.

Dear Sir,

I am sixteen years old and I study electronics at the secondary technical school. In school they said my computer is a "worth-less" thing, because of the Basic and the keyboard. But now I have a DK-tronic keyboard and Beta Basic, so it is not a "worthless" thing as they said before. Questions: Can you make an EDIT for DEF KEYs? Is it possible to make a new range for plotting points? (0-191)

Harry Friemann, Oldehove, Netherlands

You can edit a DEF KEY by typing a line number, then the DEF KEY. This will give you an editable line. At the start of the line, add DEF KEY (key), then colon or semi-colon, and enclose the text in quotes if necessary. You may need to add a trailing colon, too. Then RUN the line to re-define the DEF KEY.

Beta Basic 4.0 uses a PLOT/DRAW/CIRCLE y-coordinate range of -16 to 175, allowing the entire screen to be used. I do not plan to add this feature to a 48K Beta Basic.

Dear Andy,

I am working as a Radio Operator on an offshore rig for the last 10 years. Prior to that, I was a science teacher for a couple of years. (My subject was Botany.)... The idea of having your photo on the Newsletter is a sound one. However, the effect looks dismal... Even though you have reduced the size of the print and paper, I notice that the Newsletter actually now contains more material. And that is what counts. Please don't be distracted by the critics.

P.A. Basheer, Dubai, U.A.E.

I wasn't very serious about that picture, I just wanted to test PROC dump! I will use more interesting illustrations than my face, in future. Recent subscribers may be confused by the reference to reduced size, since even back issues now are in the smaller A5 size. (If you have the A4 versions of early Newsletters and want an A5 copy for convenient filing, send 50 pence per issue and I'll post the copies with your next Newsletter.)

Dear Sirs,

Nearly offended by ommision of Switzerland in your list of reliable Posts. (BB News no. 5) We definitely have a very good post service here... compliments for BB 3.0 and BB Newsletter.

Piotr Mazanowski, Burgdorf, Switzerland

Betasoft's correspondence with Switzerland has been 100% O.K., but the sample size is too small for that to be a good guide.

Deer Andy,

...I am 31 years old, and a programmer at British Aerospace... I enclose my Beta Basic contribution. I don't expect you to use it, it is rather large, I just wanted to show you how much time this program has saved me. I wrote a 1300 byte m/c routine which I wanted to send off for publication and I had to write a loader program. The thought of typing all those numbers into DATA statements rather put me off, so out came Beta Basic... I did experience some problems using KEYIN inside a DO LOOP. This seemed to happen when the lines being KEYIN'd were at the start of the program.

Philip Reynolds, St. Annes-on-Sea, Lancs.

(Philip enclosed a very full-featured program that creates (using KEYIN) a complete Basic program for loading machine code, with the code in DATA statements, using a block of machine code as the starting material.)

The second-to-last paragraph on page 26 of the BB 3.0 manual has some comments about DELETEinq lines while the program is running which are also relevant to KEYIN. Basically, if you DELETE or add using KEYIN) lines near the start of the program, addresses used by DO Loops, RETURNS, END PROCs etc. referring to lines lower down the listing will become "out of date" and the program will probably stop. So it is often a good idea to add or DELETE lines below the main part of the program. This should have been mentioned under KEYIN -- sorry!

Hi Andy!

I'm very glad that I now have your OPUS version of BB but one thing does not work: LLIST DATA (VAL, VAL$) because the printout is not formatted correctly (as on screen). Can you help?

To get rid of the long keyboard click it is better to load the BB code, POKE 58445, value and then save the code again. This way it will be right every time you initialise.

To Dr. Alain Vezes: You can get a program to stop with any error message including "OK" by making a USR call to the ROM. For "OK'" it is USR 7088. Use the Spectrum ROM Disassembly by Logan/O'Hara to find the others.

Keep up the good work.

Dan Olsson, Heslingborg, Sweden

LIST and LLIST DATA on BB 3.0 use the print comma for formatting, whereas TAB would probably have been a better choice (and I've implemented this in BB 4.0). If your printer interface does not understand the print comma, formatting is messy. You can alter LIST DATA to use TAB 15 instead of print comma; POKE the following bytes above RAMTOP (see issue 3): 62,23,215,62,15,215,215,201. Then DPOKE 49201 and 49130 with the address of the first byte, and POKE 49200 and 49129 with 205. Then save the altered code.

Rather than use Logan & O'Hara, how about:

```
100  DEF FN w(e)=USR (INSTRING(1,MEMORY$(),CHR$ 207+CHR$(e-1)))
```

This looks for the "Call error handler" byte (207) followed by the correct error code in ROM, then calls the required location. So e.g. PRINT FH w(5) gives Report 5.

Dear Dr. Wright,

I have a strong compulsion for filing everything so I have programmed a DATABASE according to my needs. I do not use a Microdrive and every time I start a new file I find the problem that if I want to SAVE it when only a few records have been used, I have to waste a lot of time saving a lot of empty records. I tried to use the possibility in BB 3.0 of deleting the unused strings. It worked, but when loading and trying to JOIN the previously deleted empty strings I just could not do it...

Ricardo Cohn, Montevideo, Uruguay

The example below creates an array of 100 strings, assumes only 50 are used, and deletes the remaining 50. (You could SAVE the array now.) Then an array of blanks to replace the missing strings is created and JOINed on to the first array. (Actually, the second dimension of b$() doesn't matter, since the strings will be "padded" as they are joined to a$().

```
10   DIM a$(100,10)
20   DELETE a$(51 TO )
30   DIM b$(50,10)
40   JOIN b$ TO a$
```

Dear Andy,

Is there no BB command that can clear out UDGs?

Charles Buszard, Chorleywood, Herts.

The lines below will clear the UDGs to blanks, or to their initial state (letters A to U).

```
POKE USR "a",STRINGt(168,CHR$ 0)
POKE USR "a",MEMORY$()(15880 TO 16047)
```

Charles also wanted to change the pitch of BB's error buzz. Here are a few related pokes:

ERROR BUZZ PITCH: DPOKE 56118,pitch e.g. 255 (1280=normal)
Smaller numbers give higher pitch by shortening the time between buzzer movements. This will also affect the length of the sound.

ERROR BUZZ LENGTH: POKE 56121,length e.g. 2 (0=normal)
Related to number of buzzer movements.

KEY CLICK: PITCH: DPOKE 62681,klick e.g. 40 (200=normal)
This is the click with every keypress.

KEY CLICK LENGTH: POKE 23609,length
This poke works even without BB present.

****

MEMBERSHIP SNIPPETS

Congratulations to Ettrick Thomson for winning the October 86 "Numbers Count" competition in Personal Computer World, using his Beta Basic procedures for handling large integers.

Martyn Smith edits the BETA DISC USERS CLUB, which publishes an interesting bi-monthly Newsletter. For details, send an S.A.E. to: BDUC, 2 Downham Avenue, Rawtenstall, Rossendale, Lancs., BB4 SJY. Now that Technology Research has gone bust, a user group is probably the only source of advice and support. The Newsletter includes some BB-related material.

Finn Hansen (Esbjerg, Denmark) plays chamber music for a living, and teaches the Double Bass at the local Academy of Music.

Robert Hartung (Huntertown, Indiana, USA) works with young people "falling through the cracks" of society and broken homes. These have included over 200 foster sons! He is also a county gaol chaplain.

Leslie Dewhurst is a retired Analyst/Programmer, latterly a Team Leader. He started programming in 1964 on an ICL 1301.

A crisp 10 Kronor note goes to the first Swede to send in a publishable contribution to the Newsletter - I have it lying about because it is worth 98 pence and the bank charges a pound for a currency conversion...