BETA BASIC NEWSLETTER No. 7

****

Welcome back to all the re-subscribers, and greetings to the many new subscribers! Most new subscribers have ordered the 6 previous issues as a set. I would recommend those who have not done so to do so now, because the information in the earlier Newsletters is still very useful, and I do not intend to cover the same topics again.

There are, however, two points to note when reading Back-issues:

1. References to bugs in earlier Newsletters may no longer be valid. Before "correcting" a copy of Beta Basic, make sure that it has not already been corrected - see if you can duplicate the bug - or the suggested cure may cause problems.

2. If you are using Beta Basic 4.0, examples that POKE machine code into the former printer buffer should be modified to use some other location, such as just below BB's code, or in the UDG area.

****

BETA BASIC 4.0 BUGS - AND AN EXTRA FEATURE.

Beta Basic 4.0 seems to be fairly bug-free, fortunately. The two bugs so far known are easily cured. Both were reported by Charles Buszard, for which I thank him. The bugs may have been mentioned in labels inside the cover of your BB 4.0 manual.

The first bug interfered with Beta Basic's operation of parallel printers. Location 63866 should hold 150 - if it doesn't, POKE it.

The second bug was a corruption of the RENUM routine which deals with GOTO/GOSUB etc. expressions. This caused a crash when e.g. GO TO x was renumbered. It can be cured by:

```
POKE 60413,40: POKE 60414,15
```

I managed to add one extra feature between the time the manual went to the printers and the time the cassettes were copied. The LENGTH function works with RAM disc arrays to give their dimensions; e.g.:

```
10   DIM !fred$(100,10)
20   PRINT LENGTH(1,"!fred$()")
30   PRINT LENGTH(2,"!fred$()")
```

This gives 100 and 10. LENGTH with a zero parameter gives a meaningless value with RAM disc files.

****

NEW INSTRUCTIONS FOR THE DISCIPLE DISC DRIVE INTERFACE.

Unfortunately, the instructions in the last Newsletter and the BB 4.0 manual for using BB with the Disciple have been made obsolete by the release of version 3 of the interface. The Newsletter item also sparked a lot of "what do I do if I have BB 3.0? letters. Well, the item was for either BB 3.0 or 4.0 - but particularly for BB 3.0 users who didn't have the advice contained in the BB 4.0 manual. Sorry this wasn't clear.

With the newer Disciple version 3 (and probably earlier versions too) I now suggest the following:

To copy Basic 3.0 or 4.0 to Disciple disc, load the program from tape, then MERGE the first (Basic) part again. Alter the SAVE and LOAD commands in lines 1 and 2 to the Disciple syntax. You might want to alter "Beta Basic" in line 1 to "Autoload". RUN to save the program to disc.

The disc commands will work normally as direct commands, with the exception of CAT, which requires CAT #2;1. You can write a procedure to avoid this if you wish:

```
10   DEF PROC dir device
       CAT #2;device
     END PROC
```

Now DIR 1 will catalogue disc one.

Disc commands used in programs may cause Beta Basic to lose control of the system. It is a good idea to turn Beta Basic off using RANDOMIZE USR 59904, use the disc command, and then turn Beta Basic on again with RANDOMIZE USR 58419. Disc errors will always turn Beta Basic off and require USR 59419 to turn it on again. Use of DEF KEYS can make life easier.

NOTE: I have been using the disc drive from my OPUS with the Disciple interface. People tend to say that "the OPUS is a slow disc drive", but it appears that the interface is slow, not the drive. The disc drive runs 5 or 6 times as fast when connected to the Disciple unit. To make the connection, I used a couple of 2*17 way dil IDC (insulation displacement - no soldering!) sockets and a length of cable, bought from Maplin, to make up a lead. You have to remove the OPUS cover and connect the cable to the back of the disc drive in place of the one that comes from the circuit board. (If you connect the socket upside-down, the system won't work but you can just turn the socket over and try again. By the way, I am not responsible if you damage something or electrocute yourself!) The OPUS unit needs to be switched on to power the drive, but it is not attached to the Spectrum except via the cable to the Disciple.

My thanks to ROCKFORT PRODUCTS for the loan of the Disciple.

****

PRINTERS - A BB BUG

I recently discovered a fairly nasty bug in BB 3.0 which corrupts the printer buffer when a vertical ROLL or SCROLL which includes the attributes is carried out. This unfortunately happens when you use the up and down cursors to scroll through the listing: the same routine is used. The problem is that a line of machine code that was supposed to say: LD DE,5C92H became: LD DE,5892H. The line gives the start of a memory area to be used as a temporary store for a row of attributes. The error means that instead of 32 bytes in the system variables area (MEMBOT and 2 bytes labelled "not used") being used, a 32-byte chunk of the printer buffer is altered. The normal attributes or black-on-white are 00111000 in binary; this caused some users to observe a line of dashes in the first line of a listing produced by a ZX-type printer. (I was never able to reproduce this fault - I suppose I never scrolled through the listing at the right time!)

The ZX printer effect is harmless, but if your printer is run by software that uses the printer buffer it may go wrong (perhaps only when you use particular features of the software.) I am very sorry for the problems that this must have caused some users - I am sure people must have noticed but never found an easily reproducible example to send me! To correct the fault, you can simply POKE 59727,92 to force the code to its original form. (This fix is included in the first versions of BB 4.0 - DPEEK(59726) will give 23698.) However, the original form may not be very good either, because I have since learned that both the Kempston "E" and the ZX Lprint III use the "not used" system variable. Poking in 255 would use the UDG area of UDGs H-L, which is a simple correction. Alternatively, use the program below to create and use a 32-byte attribute storage area above RANTOP. (As implemented in later BB 4.0's.) Account is taken of any DEF KEYS and WINDOWS you may have defined.

```
10   CLEAR 32
20   LET S=DPEEK(23730)+2
30   DO UNTIL PEEK (s-1)=0
40      LET s=s+3+DPEEK(s)
50   LOOP
60   DPOKE 59726,s
```

****

CONCERNING LETTERS

Just a quick aside to fill the bottom of this page! I get a lot of very interesting, encouraging and friendly letters from all over the world, which I enjoy, reading. When there is some particular problem involved, I almost always manage a reply. (The exceptions might be an obvious "pirate" lacking a manual, or when the topic is about to be covered in a Newsletter.) Even when there is no problem to solve, I often manage a quick "Glad you are enjoying yourself!"-type note with, say, an upgraded tape, in the U.K. Unfortunately, I cannot usually send such personal notes with foreign packages - a higher rate of postage is involved! This is because "Small Packets" cannot contain personal letters.

****

PRO- mirror - mirroring characters

This contribution is from Robert Dickson of London. The procedure will mirror a character, either on the screen, or elsewhere in memory, such as the UDG area. The first parameter is the address to be, mirrored, and the second is the destination of the data (usually the same). Line 1010, checks to see if the source of the data to be mirrored is in the screen memory, and account of its odd arrangement, if necessary.

```
10   PRINT "<this is a test of mirror PROC>"
15   FOR n=16384 TO 16384+31
20      mirror n
30   NEXT n

1000 DEF PROC mirror p,q
        DEFAULT p=16384,q=p
        LOCAL n,a$,b,x,y
1010    IF  p>=16384 AND p<=22577 THEN
           LET x=2043,y=256
        ELSE LET x=8,y=1
1020    FOR n=p TO p+x STEP y
           LET a$=BIN$(PEEK n),b=0
1030       LET b=(a$(1)=”1”)+2*(a$(2)=”1”)+4*(a$(3)=”1”)+8*(a$
           (4)=”1”)+16*(a$(5)=”1”)+32*(a$(6)=”1”)+64*(a$(7)=
           ”1”)+128*(a$(8)=”1”)
           POKE q+n-p,b
        NEXT n
1060 END PROC
```

An alternative shorter version of line 1(030 is given below. BIN should be the keyword.

```
1030    LET b=VAL ("BIN "+a$(8)+a$(7)+a$(6)+a$(5)+a$(4)+a$(3)+
        a$(2)+a$(1))
```

Robert also asked if I could add a GOTO line, statement command to Beta Basic. I can't see much call for this, unless you *like* confusing programs, but a PROC to do it arrived as one of a big batch from Lasso Hult, Gothenburg, Sweden. They follow (I rather like short PROCs - they are easy to understand, and I don't have to do much typing to get something I can test. Structured programming, I suppose!)

****

PROC jump - to a line and statement

The PROC alters system variables to achieve its affect. Note: It clutters up the stack with unused procedure returns.

```
10   jump 20,2
20   PRINT 1
     PRINT 2
     PRINT 3
20   DEF PROC jump lno,stat
        DPOKE 23618,lno
        POKE 23620,stat
     END PROC
```

****

PROC edge - "A way to produce your own beautiful borders"

The procedure produces a screen border of any given character (computer's or UDG). It uses DH's control codes to keep the cursor in the correct place as the right-hand margin is printed. The PRINT statement could also include INK or PAPER.

```
10   KEYWORDS 0
       edge CHR$ 134
     KEYWORDS 1

20   DEF PROC edge a$
       LOCAL b$,c$,d3,a,b
       LET a=21,b=0,b$=STRING$(32,a$),
       c$=STRING$(a,a$+CHR$ 13),
       d$=STRING$(a,a$+CHR$ 10+CHR$ 8)
       PRINT AT b,b;b$;AT a,b;b$;AT b,b;c$;AT b,31;d$
     END PROC
```

****

PROC byte - splitting integers into high and low bytes

A number of readers have noticed that if you RANDOMIZE x, the number x is placed in the system variable at 23670/23671 (SEED) as x-INT(x/256) *256 and INT(x/256). This form can be useful, as it is the normal way the Z80 handles numbers from 0-65535. (Another way would be to use the CODES of the two characters in a CHAR$ - these are in the opposite order: most significant, then least significant bytes.)

```
30   DEF PROC byte x
       RANDOMIZE x
       PRINT "LO ";PEEK 23670,"HI ";PEEK 23671
     END PROC
```

****

PROC key - An impressive "Press any key to continue"

This deceptively simple procedure fills a common need in a good-looking way. The message scrolls in the bottom part of the screen by repeatedly moving the current first character of the message to the end before printing it. The message must be 32 characters long.

```
80   KEY

90   DEF PROC key
       LOCAL b$
       LET b$="PRESS ANY KEY TO CONTINUE "
       DO
         PRINT #0;AT 1,0; PAPER 6;B$
         LET B$=B$(2 TO )+B$(1)
       LOOP UNTIL INKEY$<>""
     END PROC
```

****

KEYWORDS ON/OFF

These are very simple procedures, but they illustrate something you may not have thought of any command that might take a shift or two, or a bit of typing, can be abbreviated to a single letter with a suitable procedure.

```
100  DEF PROC G
       KEYWORDS 0
     END PROC

110   DEF PROC K
       KEYWORDS 1
     END PROC
```

****

PROC splay - playing strings

Actually, this last of the Lasse Hult contributions was called PLAY that would be "tokenised" on a 128K Spectrum. Lasse, Who works for the Swedish Railroads, writes

"Now, put the dust out of those old song books, or relearn those children-songs to sing for the kid. Here is PROC (S)PLAY, a note-simulating PROC even for 48"ers. Just put the notes you want to play in a long string and follow the simple rules!"

```
c  =simple note of 0.25 sec duration (default value)
C  =same but sharp note (c#)
c1 =full (1 sec duration),c2 for 1/2 sec, c8 for 1/8 sec
<  =raise one octave
>  =drop one octave
:  =end maker of the string
```

(I guess the ":" could be added automatically - or you could use LEN. - Ed.)

```
10   LET a$="<CCCedddfeeddc1:"
       splay a$

100  DEF PROC splay  a$
       LOCAL b$,L,s,n,m,a,b
       LET b$="cCdDfFgGaAB"
       LET L=0,m=L
       DO UNTIL a$(L+1)=":"
         LET L=L+1,a=(a$(L)="<"),b=(a$(L)=">")
         LET m=m+(12*a)-(12*b),L=L+(a OR b)
         LET s=.25 AND CODE a$(L+1)>57
         IF NOT s THEN LET s=1/(VAL a$(L+1))
130      LET n=INSTRING(1,b$,a$(L))
         BEEP s,n+m
         LET L=L+(CODE a$(L+1)<57)
       LOOP
     END PRCC
```

Beta Basic 4.0 allows some really impressive sounds - and I have reports of impressive achievement. A BB 4.0 string-playing PROC like the one above, but using three channels, and interrupt-driven sound, would be very useful.

****

ADDING A NEW BETA BASIC COMMAND

Some time ago I got the following letter:

Hi Andy

What is the possibility of using the blanks (B and H) in BB BB.0 for the two commands INCrease and DECrease? Yes, I can hear you say: That is easy to do with a line like:

```
1 DEF PRCC inc REF a: LET a=a+1: END PROC
```

but to have good use of a command like INC it should not take longer time to execute than the command: LET num=num+1 and my tests show that the inc PROC takes about 0.06 sets longer if the DEF PROC is early in a program and about 0.08 sets longer if the definition is at the end of a long program. I suppose it is all that passing of variable names that takes some time to do. So what do you say? The commands already exist in the Z8O so it should not be so hard to implement or is that a misjudgment from my side? I'm looking forward to enjoying your coming newsletters!!!

Dan Olsson, Helsingborg, Sweden

Well, I have been hoarding the last two unused keys or tokens for something really important - I'm not sure what! On the other hand, Dan's letter reminded me that some readers might be interested in adding new BB commands at the machine code level, rather than via procedures. The main advantage is, of course, speed. Passing variables does take some time; particularly since Beta Basic is "grafted on" to Spectrum Basic, rather than a total re-write. (This saves lots of memory!)

I will describe how to add Dan's INC command. It is not quite as simple as Dan guessed, since the Z8O INC command just increments a register or a memory location, whereas we have to check the command's syntax, find the variable, perhaps give an error message, and alter two memory locations. (Two memory locations allow us to deal with positive whole numbers between 0 and 65535. Things get more complicated if we want to INC negative or floating-point numbers.) All this would use hundreds of bytes if we wrote it from scratch, but fortunately the ROM is full of useful routines to help. Anyone wanting to write their own commands would be well advised to obtain Ian Logan's ROM Disassembly so that they can see how things are done. (Actually, more useful and less tedious would be something like a two page list of important routines and their entry and exit conditions - but I don't know of one!)

The syntax of all the new Beta Basic keywords is defined in a table in memory. The interpreter has to find the syntax entries as fast as possible; since they are of variable length, another table (a pointer table) is used to supply the location of each entry in the syntax table. The way this is done only really makes sense in hexadecimal. The pointer table is at F800H to F824H. The first entry is for the KEYWORDS command (CHR$ 128). The pointer value is 25H, and the syntax entry is at F800H+25H. The next entry is for DEF PROC (CHR$ 129); it has a pointer at F801H of 29H, and the syntax entry is at F800H+29H. Each syntax entry starts with one or more bytes which specify the type of

command. The ROM uses the same system. For example, a byte of "allow just the command, with no parameters" - e.g. CLS or COPY type. A byte of 6, then a 0, means "insist on a numeric expression following" - e.g. BORDER, GO TO or KEYWORDS. There is also a "miscellaneous" type, specified by a byte of 5. This means that the command itself must do the syntax check, because it is a one-off, rather than one of a class of commands. An example would be DATA or DIM. After the type byte(s) are two bytes giving the address of the routine for that command, with the least significant byte first in the usual manner.

The Graphic-B BLANK keyword (CHR$ 145) has a pointer at F811H and the syntax entry is at F892H (63634 decimal). The entry is 5, (miscellaneous type) then the address 7306 decimal. If you RANDOMIZE USR 7306 you will get an error report. This explains why BLANK normally does nothing; the error is generated when you try to enter the command, and the line is rejected. If we modify the type byte to zero (POKE 63634,0) a syntax check will be done for a no-parameter command, and you will be able to type in the BLANK keyword. The error, will still be generated when you RUN, however, since that's what the address 7306 does.

In any case, there is no "numeric variable only" type, so we will have to use a type byte of 5 (miscellaneous) and provide some code to check the syntax. The program below does this. It POKEs the code into the UDG graphics area; you should place it below BB's CODE instead if you want to incorporate the INC command permanently. See earlier issues for advice on this.)

```
10   LET adr=65376
20   POKE 63634,5
30   DPOKE 63635,adr
40   FOR n=adr TO adr+26
50      READ a
        POKE n,a
     NEXT
60   DATA 205,178,40,253 203,1,118,202,138,28,205,238,252,218,
     46,28,35,35,35,94,35,86,19,114,43,115,201
```

The assembly language translation of the DATA statement is as follows, if you are interested:

```
CALL LOOK VARS            ;check: for a variable
BIT 6,(FLAGS)
JP Z,N,NONSENSE_IN_BASIC  ;give an error if it is a string
CALL CHECK_END            ;check this is the statement end.
                          ;Stop here if this is a syntax
                          ;check.

JP C,VARIABLE_NOT-FOUND   ;error if variable didn't exist.
INC HL                    ;The HL register has been set
INC HL                    ;to point before the variable's
INC HL                    ;value, so advance it.
LD E,(HL)                 ;Get the value into
INC HL                    ;the DE register.
LD D,(HL)
INC DE                    ;INC value (or DEC if you want a
LD (HL),D                 ;DEC command) and shove it back
DEC HL                    ;into the variable.
LD (HL),E
RET
```

You can use the following lines to demonstrate the command:

```
70   LET test=0
80   BLANK test
90   PRINT test
100  GO TO 80
```

The command still looks like BLANK, rather than INC, but we will deal with that in a minute. The new command is considerably faster than LET test=test+1; just how much faster depends on how you measure it. I used:

```
10   DPOKE 23672,0: REM use FRAMES system variable
20   LET test=0
20   FOR n=1 TO 1000
30      BLANK test
40   NEXT n
50   PRINT DPEEK(23672)/50
```

I noted the result, then deleted line 30 and ran again to find the difference (1.66 sets.). You might prefer 30 REM or 30 (space) as the basis for comparison. Using: 30 LET test=test+1, minus the time without line 30, gave 4.86 secs. - so INC is almost - times faster.

****

MODIFYING YOUR KEYWORDS.

Continuing with the addition of the INC command, we now need to modify the keyword list so that the former BLANK keyword (graphic B) is listed as INC, and typed as i-n-c. (Beta Basic 3.0 has two BLANK keywords - we are altering the first one. Beta Basic 4.0 has a half-implemented END IF attached to graphic H in place of the second BLANK.)

There is no reason why you cannot modify the entire keyword list, provided it does not become too long. (The program checks for this.) Similar lists exist for the numeric and string function names. You can find them with INSTRING and MEMORY$ and alter them, as you can the main keyword list - I leave the details as an exercise for the student! Each keyword has 128 added to its last letter.

The program below creates a large LET statement at line 110 containing all the keywords. Edit this line, altering BLANK to INC, then RUN 110 to POKE the new list back into memory.

```
10   LET a$="110 let a$="""
20   FOR n=60721 TO 60905
30      LET p=PEEK n
40      IF p>127 THEN LET p=p-128
50      LET a$=a$+CHR$ p
60      IF PEEK n>127 THEN LET a$=a$+","
70   NEXT n
80   LET a$=a$+""""
90   KEYIN a$
100  STOP
```

```
120  LET b$=""
130  FOR n=1 TO LEN a$
140  IF a$(n+1)="," THEN LET b$=b$+CHR$ (CODE a$(n)+128)
       LET n=n+1
     ELSE LET b$=b$+a$(n)
150  NEXT n
160  IF LEN b-$<=185 THEN POKE 60721,b$
     ELSE PRINT "Too long by ";LEN b$-185
170  STOP
```

****

MARKING OUT PROCs IN A LISTING.

I think a lot of BB users do not know all the things you can do with ALTER (the program). Remember to use brackets around a variable or expression if you want to look for the result of that expression rather than the expression itself. (Read that again!) For example, in the lines below we want to alter occurrences of END PROC to make them stand out in a listing. We could use:

```
110 ALTER "DEF PROC" TO "DEF PROC------------------"
```

making sure the DEF PROC is a keyword (Graphic-1). However, the line will find and alter itself, which means it will only work correctly *once*. It is better to use (CHR$ 131) which evaluates to DEF PROC (try: PRINT CHR$ 131) but which will not be altered, since it is not *actually* DEF PROC.

Lines 120 and 130 follow a suggestion of Francisco Riera Alibes (Barcelona, Spain). The STRING$s can be longer if desired. Lines 140 and 150 are my own preference - I usually have an extra ":" at the end of a procedure to force a line feed in LIST FORMAT 1 or 2. Oh yes, these lines don't make much sense all together like this - execute them one at a time when you have got some PROCs loaded, and look at the listing afterwards!

```
120  ALTER (CHR$ 131) TO (CHR$ 131+STRING$(18,"-"))
130  ALTER (CHR$ 131+STRING$(18,"-")) TO (CHR$ 131)
140  ALTER (CHR$ 131) TO (CHR$ 131+":")
150  ALTER (CHR$ 131+":") TO (CHR$ 131)
```

Another few points about ALTER:

If you want to ALTER "123456" say, prevent the line itself being altered by using e.g.; ALTER ("123"+"456") TO whatever.

You can ALTER or REF "a(" or "a(x)" but you will not find "a(1)" (unless it is part of a string) because something which appears in a listing as:

```
PRINT a(1)
```

actually exists in the program as:

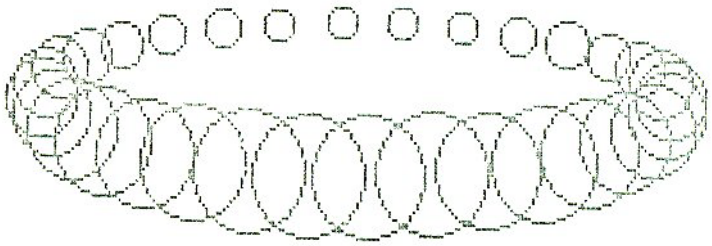
```
PRINT /a/(/1/CHR$ 14/CHR$ 0/CHR$ 0/CHR$ 1/CHR$ 0/CHR$ 0/)
```

I have used "/" to separate individual characters.

****

ANIMATION

Beta Basic's POKE and MEMORY$ features provide a simple means of animating graphics. I am sure you will be able to apply the basic idea below to other scenes. I tried for a rotating cube, but I must have tangled my COS`s and SIN's somewhere, so I gave up and settled on a doughnut-thing. Altering the values in line 80 will change the doughnut position, x and y size, and ring thickness. The illustration shows the doughnut, which is animated to spin at a very respectable speed by the program. Not shown is a central coloured shaft. This is red INK on red PAPER in its upper half, so that any circles there are invisible and appear to pass "behind" the shaft. The lower half of the shaft has normal INK and circles seem to pass in front of it.



Animation is achieved by creating each frame in the central third of the screen, and then storing it in an array (line 50). See the BB 3.0 manual under MEMORY$ (function). Later (line 200) the frames are POKEd back. Only 4 frames are used here, which gives a surprisingly good effect. Many games use just 3 frames for character animation. The array a$ uses about 8K. BB 4.0 owners can use RAM disc arrays of up to 64K, giving many more frames or larger areas of the screen, with some sacrifice of speed.

```
10    DIM a$(4,2048)
20    LET f=1
30    FOR z=0 TO PI/16 STEP PI/64
40       torus z
50       LET a$(f)=MEMORY$()(18432 TO 20479)
         CLS
         LET f=f+1
         NEXT z
60    shaft
      Animate

70    DEF PROC torus ang
80       LET x=128,y=86,xm=90,ym=20,r=10
90       FOR c=PI/16+ang TO 2*PI+ang STEP PI/16
100         CIRCLE x+SINE(c)*xm,y+COSE(c)*ym,r-COSE(c)*6
110      NEXT c
120   END PROC
```

```
130  DEF PROC shaft
140     FOR t=0 TO 9
           PRINT INK 2; PAPER 2;AT t,15;" "
        NEXT t
150     FOR t=10 TO 21
           PRINT PAPER '-;AT t,15;" "
        NEXT t
160  END PROC
170  DEF PROC animate
180     DO
190        FOR f=1 TO 4
200           POKE 18432,a$(f)
           NEXT f
210     LOOP
220  END PROC
```

****

PROC KEYBOARD - a musical PROC

This contribution is from T. Holland of Whitley Bay, Tyne & Wear. He says: "..it is a very simple (48k) two octave BEEP Keyboard malting use of GET and ON. Although very simple it may be of some interest to those subscribers who have children who "dabble" on the keyboard."

The notes are on 1-8, 9 is change octave and 0 is exit the PROC.

```
1    KEYBOARD

100  DEF PROC KEYBOARD
110     POKE 23609,0
        PRINT AT 11,8; CSIZE 16,32;"KEYBOARD"
        DO
           PRINT AT 0,12;"OCTAVE 1"
           DO
              GET NUMBER
              ON NUMBER
                 BEEP .3,-12
                 BEEP .3,-10
                 BEEP .3,-8
                 BEEP .3,-7
                 BEEP .3,-5
                 BEEP .3,-3
                 BEEP .3,0
120        LOOP UNTIL NUMBER=9 OR NUMBER=0
130        PRINT AT 0,12;"OCTAVE 2"
           DO WHILE NUMBER<>0
              GET NUMBER
              ON NUMBER
                 BEEP .3,0
                 BEEP .3,2
                 BEEP .3,4
                 BEEP .3,5
                 BEEP .3,7
                 BEEP .3,9
                 BEEP .3,11
                 BEEP .3,12
140        LOOP UNTIL NUMBER=9
150     LOOP UNTIL NUMBER=O
160     POKE 2=609,52
     END PROC
```

****

READERS' LETTERS

Dear Dr. Wright,

I particularly like the idea of combining INSTRING and CHAR$, to look for a particular number in a specific data field. However, the fact that "#" means "any" when it occurs in the second (or subsequent) character(s) of an INSTRING search prevents it from finding a unique number. Is there a handy POKE to disenable the normal INSTRING usage of "#"?

H.N.S. Wijegoonawardena, Edgware, Middx.

There is. For BB 3.0. POKE 63120,x to change the "any" character to CHR$ x. Or POKE 63122,0 for no "any" character, and POKE 63122,241 to restore the "any" feature. For BB 4.0, the equivalent code is part of the "bbc2" file, which is normally loaded via screen memory to banked RAM. The easiest way to alter it is probably without BB present:

```
CLEAR 29999
LOAD "bbc2" CODE 300000
POKE 34789,(new "any" character)
POKE 34791,(0 or 241)
SAVE "bbc2" CODE 30000,6144
```

Use the altered "bbc2" file in future. Alternatively, you could try changing the line 2 loader, inserting the POKES just after "bbc2" CODE has loaded.

Dear Andy,

What's happening about Beta Basic and the PLUS 3?

(Several readers.)

I have seen one and copied the ROMs for disassembly. (60-odd K to check through) The manual is nice for techies like myself. The disc drive works quite nicely, although I was surprised there are no serial files from Basic. I think quite a number of BB users are thinking about this machine. The problem for them and me is price. If it drops below 200 pounds, it should sell O.K. and there would be a BB for it. If it stays at the launch price, I would be discouraged. Beta Basic 4.0 has not sold wonderfully, at least partly because there has been no food of upgrade orders as in the past. It will need a good price to price loose the majority of 48K owners from their machines.

Dear Andy,

I frequently use the BETA as a toolkit to refine programs that are to run without BETA resident and I would therefore like to kill line 0 prior to saving. Is there an easy way?

Andy Hollis, Northwich, Cheshire

Sure! DELETE 0 TO 0 *will do it.*

Dear Dr. Wright,

PROC hide in issue 1 or 2, I think ... if you hide a routine, at line 0, which produces an error ("Failed at:") message from RENUM (when not at line zero) it will crash after it has been hidden! Not worth trying to fix, this I would have thought - just a feature to be noted.

C.A. flash, Bracknell, Berks.

Like many bugs, an unforeseen interaction of several things. . . BB places a double zero on the stack before a RENUM, and stacks the line numbers of any "Failed" lines on top. At the end of RENUM, the messages are printed, down to the double zero, which is junked. Then a RETurn is made to the next thing on the stack. A stacked line number of double zero causes havoc...

Dear Andy,

I wonder if you could say something in the next Newsletter regarding the compatibility of programs written using the 48K version of Beta Basic with your new 128K version. Can they be used in 128K mode?

Leslie Dewhurst, Leamington Spa, Warcs.

I always try to keep successive versions as compatible as possible. Your existing programs should run O.K. in 128K mode - Just rather faster if they use DRAW or CIRCLE. To use the new functions you will have to MERGE-in a copy of the new line zero.

Dear Sirs,

Where can I PEEK the preset alarm time?

Francesco Stajano, Rome, Italy

PRINT MENORY$() (61202 TO 61209) *will do this.*

Dear Dr. Wright,

You may remember some time ago I sent you copies of the first of KEY SOFTWARE's productions. I now enclose a further three programs which have all been written using BETA BASIC. These programs have been well received by the many schools which have purchased copies and have been praised when reviewed in CRASH. As a small token of thanks for allowing us to use Beta Basic, I'd like to offer a discount to Beta Basic users who might be interested in educational programs. ...I could give 1 pound off every tape or Microdrive sold to people who mention the newsletter offer when ordering. Readers can write to me at the address below to receive details of the programs.

Andy Watson, KEY SOFTWARE, 33 Hilton St., Aberdeen, AB2 3QT

These programs are good, well-documented and cheap! Write to KEY SOFTWARE if you are interested in educational software for spelling, decimals, fractions, etc.

Dear Andy,

As the publisher of program material I often receive modified programs based upon original material which I often wish to publish. The biggest problem is extracting all the relevant lines which modify the original. What I need is a "MERGE/DELETE" viz a MERGE which will delete all lines the SAME retaining lines which are different to those in memory. Ah! I hear you say - a file comparison is required! I've written several but never find them entirely reliable. I wonder if perhaps either yourself, or maybe fellow newsletter readers can come up with an idea for me?

I have a small tip for Microdrive users with Interfaces having issue II ROMs. Location 23791 (COPIES) may be POKEd by a direct command to save multiple copies of a file. On my BB 3.0 cartridge I have 20 "run" files,5 "Beta-Basic" and 2 "BB". Loading of BB 3.0 is much quicker than normal. For example, when making a "run" file simply:

```
POKE 23791,20: SAVE *"m";1;"run" LINE 1
```

Paul Newman G4INP, 3 Red House Lane, Leiston, Suffolk IP16 4JZ

Paul runs the Sinclair Amstrad Radio User Group (SARUG) which publishes a bi-monthly newsletter packed with interesting stuff for radio hams with Sinclair or Amstrad computers. I can't solve his problem off the top of my head - anyone out there like to have a go? I have been reduced to comparing two printouts by overlapping them over a bright light in the past - there must be a better solution!

Thanks for the tip! It is a pity that the multiple copies of a short file are not distributed evenly around the cartridge. This would cut the loading time a lot with just 4 or 5 copies. I suppose multiple copies are more reliable; if a sector is damaged in one copy it can be read from the next copy. I presume multiple ERASE commands will be needed to get rid of such files. You imply this only works with version II ROMS - true?

Dear Sirs,

Having been an enthusiastic user of your Beta Basic for some time, I am now considering whether to buy an Amstrad PC 1512. I have however been surprised (and dismayed) to learn that of its 512K RAM, only 21k is available for use with its own Locomotive Basic 2. I am therefore wondering whether you have any plans to market a version of Beta Basic to run on the Amstrad, which would be able to mane full use of the 512K RAM?

John Loncaster, Humberside.

I am not surprised you are dismayed! But I find the situation rather amusing... It would be a lot of work to convert BB to another CPU chip, and make it independent of the Spectrum's ROM. I considered converting Spectrum Basic for the Amstrad CPC and PC, machines. They have Z80 "brains", and the ability to switch in RAM in the lower part of memory occupied by ROM on the Spectrum, so a modified Spectrum ROM code could be loaded there. Unfortunately, this would entail some use of copyright Spectrum ROM code - and Amstrad won't allow this! (I asked?). A pity - it would have allowed a lot of people to transfer programs, and it, could have been most of the work or converting Beta Basic to those machines.

Dear Andy,

Concerning newsletters Nos. 4 and 5, I have been using PROCS dump and rotate to print the screen produced by PROC hilite. However, I've found that the screen is not dumped correctly unless PROC hilite contains a LOCAL n statement after the DEF PROC. Without this, the address of the first byte of data passed by "dump" to "rotate" is altered by "rotate". The difference seems to be 19 bytes. Is this due to the FOR... NEXT control variable being deleted?

Keith Davies, St. Albans, Herts.

Thanks. You are correct. Adding a LOCAL n to hilite ensures that n is deleted earlier and cures the problem.

Dear Dr. Wright,

I have not been able to save data that is made up of screen blocks using GET. Whenever I try to reload them I get the invalid colour error. Is there a reason for this or am I doing something wrong?

A.L. Storm, South Dunedin, New Zealand.

If you tried something like:

```
GET a$,100,100
SAVE "name" DATA a$()
```

then the problem is the Spectrum Basic bug mentioned in issue 5 that allows simple strings to be saved, but not loaded back correctly. The easiest solution is probably something like this:

```
GET a$ (something)
DIM b$(LEN a$)
LET b$=a$
SAVE "name" DATA b$()
```

A better solution when the screen block is large (making the DIM and assignment expensive on memory) is one suggested by reader Bo Leuf of Gothenburg, Sweden. The nature of the bug means that if you add leading characters which mimic part of an array "header" to the string before saving it, the re-loaded string will be of array type, but intact; e.g. before saving:

```
LET a$=CHR$ 1+CHAR$(LEN a$)(2)+CHAR$(LEN a$)(1)+a$
```

or if memory is tight, reduce the space needed temporarily for the assignment by:

```
LET b$=CHR$ 1+CHAR$(LEN a$) (2)+CHAR$(LEN a$)(1)
JOIN b$ TO a$(1)
```

The reloaded string, now a single-dimensioned array, can be PLOTed back to the screen directly.