BETA BASIC NEWSLETTER No.3

Hello again! Let's start with some info on what your fellow subscribers are up to:

Charles Buszard has written to correct the impression given in the last issue that he is the Secretary of the Royal Photographic Society. He is in fact a secretary at a rather humbler level. Eric Pendleton is a nuclear engineer. He uses his Spectrum to do the accounts of the SDP in Warrington, among other things. Dominic Handy is training to be a P.E. teacher and writes part-time for CRASH, AMTIX and ZZAP 64 magazines. Per Dahlin is a programmer in Sweden. Beta Basic is the most powerful Basic he has seen, although he doesn't like the Spectrum's speed.

****

PROC C - keeping CAT data on-screens.

This contribution from Manlie Reeve of Birmingham catalogues the Microdrive on the extreme right of the screen. Your program listings are restricted to the left of the screen so that the catalogue is not over-written and can be consulted at need. Unfortunately, this procedure showed up a Beta Basic bug in that an error message is given if the CAT involves scrolling. It took a while to work out what was happening - partly because the problem is not apparent with the OPUS version I use most of the time. The problem was caused by common use of a temporary storage area by CAT and the SCROLL routine, when a window is used. (I hadn't realised the problems of one "command" occurring during another!) It is not important unless your catalogues take more than 22 lines, but it can be cured quite easily by:

```
POKE 60082,12: POKE 60088,14: POKE 56097,12: POKE 56100,14
```

The procedure has a one-letter name for ease of use:

```
9970 REM press "C" to CATalogue
9971 DEF PROC C
        WINDOW 1,0,175,188,176
        WINDOW 1
        INK  0
        PAPER 7
        CSIZE 4,8
        WINDOW 2,199,175,56,176
        WINDOW 2
        INK 4
        PAPER 0
        CSIZE 5,8
        WINDOW 0
        WINDOW 2
        CLS
        CAT
        WINDOW 1
        CLS
        LIST
     END PROC
```

****

PROC prtstr - printing text without word-splitting.

This procedure is a slightly modified contribution from Geoff Stillwell of Borehamwood. Its purpose is to print long strings to any window at any CSIZE without splitting words at the end of lines (except at hyphens). The BB system variable CHPL is PEEKed to out the current number of characters per line, and this is put into the variable *width*. The string is printed in pieces that will fit onto separate lines, using the outer DO-loop. The variables *first* and *last* (or *temp*) indicate the extant of the current piece. Initially *first* is the first character in the string, and *last* is either the end character in the string (when the remaining length is less than a full line) or the last character that might just fit onto than current line. The middle loop (DO WHILE temp<end ...LOOP UNTIL 1) is equivalent to: IF temp>end... END IF. (This structure was discussed in issue, 2.) If temp<end we have to break the string; the inner DO-loop moves last backwards through the string looking for a space or hyphen where we can do this. The piece is then printed, and the EXIT IF finishes things if the and of the string has been reached. Otherwise, *first* and *temp* are adjusted for the next piece and the outer DO-loop is gone round again.

```
10   prtstr "a very long string - several lines.."

100  DEF PROC prtstr z$
       LOCAL first,last,width,end
       LET width=PEEK 57391,end=LEN z$,first=1,temp=width
       DO
         LET last=end
         DO WHILE temp<end
           LET list=temp
           DO UNTIL z$(last+1)=" " " OR z$(last)="-"
             LET last=last-1
           LOOP
         LOOP UNTIL 1
         PRINT z$(first TO last)
       EXIT IF last>=end
         LET first=last+1+(z$(last+1)=" "),temp=first+width-1
       LOOP
     END PROC
```

****

PROC centre - printing centred text.

This contribution is from Lars Hult of Goteborg, Sweden. A$ is the string to centre and x is the line to print it on. The default line is in the middle of the screen. You could use:

```
DEFAULT x=24-PEEK 23689
```

if you wanted the default line to be the current screen line instead.

```
100  DEF PROC CENTRE a$,x
       DEFAULT x=11
       PRINT AT x,(31-LEN a$)/2;a$
     END PROC
```

****
THE HANOI TOWERS

This contribution is from Guy Loui of Belgium. He enclosed a minimal version and a longer one that looks prettier but unfortunately is too long to include. He writes:

I have been an informatician for 15 years. I am amazed yet by your extension to Sinclair Basic. One of the very nice features of this language is the implementation of procedures., especially recursive ones. Most microcomputer users are not familiar with recursivity, so I am sending you a program which solves the well known problem of "The Hanoi Towers". I recall the story of this problem: in a Buddhist monastery at Hanoi, there are three great pegs with thirty-two golden rings of different sizes. At the beginning, the rings were placed on the first peg, in ascending order, the little one on the top. The problem is to move all the rings from the first peg to the second, with the following constraints:

1. Movements may involve any of the three pegs.
2. Moving more than one ring at the same times is not allowed.
3. A bigger ring cannot be put on a smaller one.

Every day since the creation of the monastery, a monk moves one ring, and the Buddhists say that the End of the World will happen when the problem is solved. Note that the monks know the solution and have an algorithm to move the rings but we can calculate (see the program) that, in the best case, the time to move the rings is approximately 10 million years! To explain how the problem is solved, we see that to solve the problem of moving N rings from peg 1 to peg 2 (using peg 3 as a working peg), we must:

First, move the first (N-1) rings from peg 1 to peg 3
Then, move the Nth ring from peg 1 to peg 2
Finally, move the (N-1) rings from peg 3 to peg 1

So, the solution of the problem is typically recursive. Note carefully how we get out of recursivity: when N is equal to 0 within the HANOI procedure, we do nothing and just quit this level of recursion.

```
10 INPUT "Number of rings?",n
   hanoi 1,2,3,n
   PRINT "Number of moves: ";2↑n
   STOP

20 DEF PROC hanoi A,B,C,n
     IF n>O THEN
     hanoi A,C,B,n-1
     move-ring n,A,B
     hanoi C,B,A,n-1
30 END PROC

40 DEF PROC move-ring n,X,Y
     PRINT "Move ring ";n;" from peg ";X;" to peg ";Y
50 END PROC
```

****

CASE - a procedure to simulate a CASE statement.

CASE is a statement that is provided in some computer languages to handle a series., of conditions. Where you might have used:

```
10 IF a=1  THEN PRINT "one"
20 IF a=4 THEN PRINT "four"
30 IF a>10 THEN PRINT "too big"
```

you something like this (don't types this in!):

```
10 CASE OF a
   =1: PRINT  "one"
   =4: PRINT "four"
   >10: FRINT "too big"
```

Some versions allows just the variable given after CASE OF to be tested, others allow any variable to be used Sometimes complex expressions are allowed. Beta Basic does not have a CASE statement, but it is possible to simulate one with a procedure. (Alternatively, some applications can use ON - see the method used in PROC bplot Newsletter 1.)

The procedure at line 100 takes a list of expressions as its parameters. They come in pairs, the first one in each pair being a comparison (which gives a numeric value equivalent to true or false when it is READ) and the second being a string containing the commands to be executed if the preceding comparison is true. The comparisons are tested until one is found to be true:

```
EXIT IF expr
```

is equivalent to:

```
EXIT IF expr=1
```

(i.e. EXIT IF expr is true.) Next, KEYIN is used to execute the associated string of commands. If no comparisons are true, ITEM() will detect the end of the DATA list, and a null String is used instead of a command string. If several comparisons are true, only the first on is responded to.

```
100  DEF PROC case DATA
       LOCAL expr,s$
       DO WHILE ITEM()
         READ expr,s$
       EXIT IF expr
         LET s$=""
       LOOP
       KEYIN s$
     END PROC
```

Now type in some lines to test the procedure:

```
10 DO
     INPUT x
20   case
     X=1,"print 1:beep .1,1",
     x=4, "print 4",
     x>10,"print too big""
30 LOOP
```

Line 20 has been "prettied up" using CAPS SHIFT/ENTER to cause printing of one statement on several screen lines. It would otherwise look like this:

```
20 case x=1,"print 1:beep  .1,1",x=4,"print 4" x >10, "print
   ""too big"""
```

Notice the extra quote marks needed to enter some strings containing their own quote marks. If you get confused, try printing the string you are using - what you see on the screen is what KEYIN is going to "type in" to the Spectrum.

The procedure will work with more complex comparisons of several (possibly string) variables - anything which could be used after an IF statement.

****

PROC axes - drawing graph axes with scale markings.

Sometimes a programming idea turns out to require more work than one might expect; PROC axes is an example. Drawing the axes of the graphics system and marking them sounds pretty simple. But there are problems if this is to work with different scales. For example, you will want the scale markings to be the same size even if XRG and YRG change. PROCs gdraw and gplot are subprocedures of PROC axes which provide draw and plot commands which are not affected by the current XRG or YRG. PROC axes takes 2 parameters which specify the space between markings on the x and y axes (scaled according to the current XRG and YRG). It draws both and than uses 4 FOR-NEXT loops to tic and label the 4 "arms" of the cross thus formed.

```
10   LET xrg=128,xos=50,yos=30
20   axes 20,50

100  DEF PROC axes xstp,ystp
       LOCAL xr,yr,x,y
       LET xr=xrg/256
       LET yr=yrg/176
       PLOT -zos,0
       gdraw 255,0
       PLOT 0,-yos
       gdraw 0, 175
       FOR x=xstp TO xrg-xos-12*xr STEP xstp
         PLOT x , 0
         gdraw 0,2
         PLOT CSIZE 4,8;x-4*xr,10*yr,STR$ x
       NEXT x
       FOR x=-xstp TO -xos+6 STEP -xstp
         PLOT x,0
         gdraw 0,2
         PLOT CSIZE 4,8;x-6*xr,10*yr;STR$ x
       NEXT x
       FOR y=ystp TO yrg-yos-1 STEP ystp
         PLOT 0, y
         gdraw 2,0
         PLOT CSIZE 4,8;3*xr,y:STR$ y
       NEXT y
       FOR y=-ystp TO -yos STEP -ystp
         PLOT O, y
         gdraw 2,0
         PLOT CSIZE 4,8;3*xr,y;STR$ y
       NEXT y
     END PROC
```

```
500  DEF PROC gplot x,y,z$
       LOCAL sxrg,syrg
       LET sxrg=xrg,syrg=yrg,xrg=256,yrg=176
       DEFAULT z$=""
       IF z$="" THEN PLOT x,y
       ELSE
          PLOT CSIZE 4,8;x,y,z$
510    LET xrg=sxrg,yrg=syrg
     END PROC

600  PROC gdraw x ,y
       LOCAL sxrg,syrg
       LET sxrg=xrg,syrg=yrg,xrg=256,yrg=176
       DRAW x,y
       LET xrg=sxrg,yrg=syrg
     END  PROC
```

****

PROC down - printing text vertically.

This is another contribution from Lars Hult. AZ is the string to print and x and y are the AT coordinates to use. They default to the top left-hand corner.

```
200  DEF PROC  DOWN a$,x,y
       DEFAULT x=0,y=x
       LOCAL z
       FOR z=1 TO LEN a$
          PRINT AT x+z-1,y;a$(z)
       NEXT z
     END PROC
```

****

"AUTO SAVE"

Daniel Ben-Sefer of Israel writes:

"I include my "auto save" subroutine, which is part of every program which is in the process of being written. It causes an automatic SAVE to cartridge every 20 minutes. This way, if there is an electrical failure, or your 5 year old boy jokingly pushes the reset button, the most you have lost is 20 minutes of programming time."

Once line 8500 has been RUN (usually by auto-running) line 9000 Will "be GOSUBed every 20 minutes, causing the auto-SAVE. Lines 9010 and 9020 reset the CLOCK alarm for 20 minutes in the future. I have altered line 9010 to make it slightly more flexible - the "20" at the end of the second statement is the number of minutes between SAVES. If the alarm goes off while you are writing or editing a program line, the automatic SAVE will not interrupt you - it will be delayed until you next execute a program line (with RUN or GO TO).

```
8500 CLOCK   6                    (or  CLOCK 4, 5 or 7 )
     CLOCK 9000
     GO SUB 9010
     GO TO 1                      (or main program)
9000 LET a$="prog name"
     BORDER 5
     ERASE a$
     BORDER 6
     SAVE 1;a$ LINE 8500
     BORDER 7
     VERIFY 1;a$
```

```
     9010 LET a$=TIME$
          LET mins=VAL a$(4 TO 5)+20
          IF mins>=60 THEN
          LET mins=mins-60
          LET a$( TO 2)=USING$("00",VAL a$( TO 2)+1)
     9020 LET a$(4 TO 5)-USING$("00",mins)
          LET a$="a"+a$
          CLOCK a$
          RETURN
```

****

USE OF SHIFT AND INSTRING ON INPUTS.

Rod Macaulay of Aberdeen writes:

"Thanks for a superb program. ...I am an English teacher turned remedial teacher in Aberdeen. I write short. programs (educational and otherwise) for my own children and also for my pupils at school. I was interested in the keyprint procedure in newsletter 1 as it seemed an intersting way to make children's programs more user friendly rather than the usual INPUT - NAME etc. I thought you might be interested in the way I have used SHIFT$ and INSTRING to give upper case when necessary and select first name only for friendly computer greeting!"

First, give a prompt and get the name in n$ using the keyprint procedure, or the ReadString procedure in this issue, or other means, then use the lines below - which I have edited slightly.

```
     100  INPUT n$
     120  CLS
          LET fnm=INSTRING(1,n$," ")
          IF fnm=0 THEN LET fnm=LEN n$
     130  PRINT CSIZE 8,16;AT 5,1;"Hello There"';"   ";
          SHIFT$(1,n$(1));SHIFT$(2,n$(2 TO fnm));"!!"
```

****

PROC ReadNumber

Judging by our correspondence (including many unpublished contributions) there is a lot of interest in INPUT subroutines (like PROC keyprint in issue 1) particularly ones that provide INPUT AT. Ian Brown of South Africa has submitted a nice pair of procedures, one for numeric and one for string input. He writes:

"This routine accepts *numeric input only* at any screen position, and checks that the value lies within the specified range. Additional calling parameters specify whether the number must be an integer value only, or whether it may contain a decimal point; and whether the value may be negative. As presented below, a negative value may only be input with the minus sign as the first character. I use this routine by putting all the prompting text on the screen first, and then using ReadNumber or ReadString to accept the input at the required positions."

I have altered Ian's examples slightly. At program line 20 a number to be put into variable *Fred* is accepted at line 10, column 18. It must lie between 0 and 31 and (by default) it must be a positive integer. At program line 40 a number to be put into the, variable *crud* (Ian's idea!) is accepted at line 13, column 18. It must lie between -100 and +220.5 - the last two parameters specify that decimal and negative numbers are to be accepted. DELETE can be used normally.

The assignment to C$ in line 1030 makes the cursor a flashing space (square) character but this could be altered if desired. Maybe a flashing "N" for numeric?

```
     10   PRINT AT 9.0;”Enter a number”’”between 0 and 31:”
     20   ReadNumber fred,10,18,0,31
     30   PRINT AT 12,0;"Enter a number between"'"-100 and +220.5:"
     40   ReadNumber crud,13,18,-100,220.5,1,1
     50   CLS
          PRINT fred,crud

   1000   DEF PROC ReadNumber REF number,lin,col,min,max,decimal,ne
            gative
   1020     DEFAULT decimal=0,negative=0
   1030     LET valid=0,b$=“”,c$=CHR$ 1+” “+CHR$ 18+CHR$ 0
   1040     DO UNTIL valid
   1050       LET length=LEN b$+1,a$=“”,b$=“”
              PRINT AT lin,col;c$+STRING$(length,” “)
   1060       DO UNTIL a$=CHR$ 13
   1070         GET a$
                IF a$=CHR$ 12 AND LEN b$>0 THEN
                  LET b$=b$(  TO LEN b$-1)
   1080         IF negative AND a$=“-” AND b$=“” THEN
                  LET b$=a$
   1090         IF (a$>=“0” AND a$<=“9”) OR (decimal AND a$=“.”)
                THEN
                  LET b$=b$+a$
   1100         PRINT AT lin,col;b$+c$+” “
   1110       LOOP
   1120       IF VAL b$<min OR VAL b$>max THEN
                BEEP .1,10
              ELSE
                PRINT AT lin,col;b$+” "
                LET number=VAL b$
                LET valid=1
   1130     LOOP
   1140   END PROC
```

****

PROC ReadString

This is the second of the pair of procedures from Ian Brown:

"This procedure accepts any string input at any position on the screen, up to the maximum length specified. As in the case of ReadNumber, the normal deletion method on the Spectrum is used. The cursor, C$, is as defined for ReadNumber. Exit from this routine occurs, when either the ENTER key is pressed or the string has reached the length specified.

Example: ReadString N$,15,22,10 - accept astring of up to 10
characters at screen position 15,22 find out it in N$.

```
2000    DEF PROC ReadString REF b$,lin,col,length
2010      LOCAL a$,c$
2020      LET b$="",ok=0,c$=CHR$ 10+CHR$ 1+"  "+CHR$ 18+CHR$ 0
          PRINT AT lin,col;c$;
2030      DO UNTIL ok
2040        GET a$
2050        IF a$=CHRS 12 AND LEN b$>0 THEN
              LET b$=b$( TO LEN b$-1)
2060        IF aS<>CHR$ 12 AND a3<>CHR$ 13 THEN
              LET b$=bS+aS
2070        PRINT AT lin,col;b$+c:$+" ";
2080        IF aS=CHR$ 13 OR LEN b$=length THEN
              LET ok=1
              PRINT AT lin,col;b$+" "
2090      LOOP
2100    END PROC
```

****
****

READERS LETTERS

Dear Andy,

I use Beta Basic with my SPDOS disk system, and appreciate the fast loading. The only snag is that the BB code wipes out the DOS code on loading ... do we have many newsletter subscribers in the same position..? Can any fellow subscriber help me locate down-to-earth practical information to enable me to interface BBC 5.25" disk-drives to any Z-80 computer, and to control such drives direct, without the need for a disc operating system as such?...

Ken Rendall, North Berwick

Sounds like a difficult project to me ... Can anyone help?

****

Dear Sirs,

When going through the Beta Basic manual I have decided to try a random (RNDM) plotting of a point on the screen to see how fast it is. It produced quite a random pattern. When I tried to do the same in Sinclair Basic the pattern does not seem to be as random, it seems to have a pattern of vertical lines.... perhaps you can explain....

Peter Safranek, Ashford

Mr. Safranek enclosed some nice screen copies showing the phenomenon clearly. The lines used were:

```
10   FOR n=1 TO 10000: PLOT RNDM(255), RNDM(175): NEXT n
10   FOR n-1 TO 10000: PLOT RND*255,RND*175: NEXT n
```

Software cannot produce numbers which are really random. Instead, a series of values is calculated, in such a way that the values look, fairly random. The last "random" number is used as the starting point when calculating the next one. In practice, the series will contain patterns which will be pore obvious if the calculation method isn't very good, or if the application shows up a particular type of non-randomness. I think that that is what is happening in the RND version. Before Beta Basic was published, various possible ways of writing the RNDM function were to fact tested using exactly the method that Mr. Safranek used. The version I used gave the most "random" patterns - thanks to one of our users! See below...

****

Dear Andy,

I was pleasantly surprised to find that the successive values of SEED (*the random number system variable used by* RND *and* RNDM - *Ed.*) are calculated by an improved version of a routine of mine that appeared in PCW's Sub Set ,June 82...

...It would be a great help in loading machine code above RAMTOP if Beta Basic had a System Variable that recorded the start of BB's code. .

W. E. Thomson, Suffolk:

Thanks for pointing out your early contribution! The "start of BB" system variable is a good idea, which I will leave up to code users to implement for now. (Hr. Thomson is suggesting a double-location which can be DPEEKed and DPOKEd as extra code routines are added to BB. RAMTOP itself is not satisfactory as it moves about as keys and windows are defined.)

Mr. Thomson's letters have included many interesting points for assembly-language or math's. Recently he submitted improved versions of RNDM and SOR (in assembly language) and a set of procedures to deal with integers of any size (add, subtract, multiply, etc.) with complete accuracy. This allows operations like testing for primality to be done or, large numbers. (The limited accuracy of most micros ordinarily makes such things very difficult.) If any reader's interests are a bit esoteric, we will try to put them in touch with like-minded subscribers, without necessarily publishing letters or contributions in full.

****

Dear Dr. Wright,

...I have found the key click very offensive to my ear, also the error beep rather short My solution has been to alter line 2 of Beta Basic by poking numbers after the instruction (RANDOMIZE USR 58419 - Ed.) invoking the program.

```
POKE 23609,0: POKE 23608,255
```

The first turns off the key click and the second lengthens the beep.

G.J. Doodson, Telford

****

Dear Dr. Wight,

What's that REM and RANDOMIZE USR 0 doing in n line zero?

(Various correspondents)

Difficult to explain - but I'll try! The answer is relevant to the use of PROC hide (BB News no. 2) with Microdrive commands.

The Spectrum was never designed to have an "expandable" Basic. Beta Basic therefore has to keep control of the machine by running vital routines in RAM. It makes extensive use of the original ROM, but treats it as a huge block of useful subroutines. If we want to change the way one of the ROM routines works. a similar routine will have to be provided in RAM. For example, the expression evaluator could be re-written to be much faster and include new functions without using DEF FN - but this would use a big chunk of RAM. In addition, every command in the ROM that is used by Beta Basic would have to be re-written to use the new expression evaluator rather than the old one. This is tremendously inconvenient and is avoided on better designed machines by having ROM routines look in RAM for the addresses of various important routines - like the expression evaluator.

The vain ROM is bad enough - the ROM in Interface 1 is worse! At the end of various large Microdrive routines, there is no RETURN - instead, the stack is cleared, and a "GOTO rain ROM" is performed. This will cause Beta Basic to lose control of the system (although RANDOMISE USR 56419 will regain control) unless something special happens. This is were the RANDOMIZE USR statement in line 0 comes in; before a Microdrive command is executed, BB sets up some system variables (NEWPPC and NSPPC) so that when the main ROM takes control, the RANDOMIZE USR 0 is executed next. The invisible 5-byte form of the number which follows "0" is the important thing - and it gives the address of some code which turns BB back on and then goes back to the program at the next statement after the Microdrive command, using a machine code equivalent of G0T0 line and statement. This can cause problems if the program is running an additional line zero (as suggested in Newsletter no. 2 under PROC hide) because the first line zero will be GOTO ed. This limits the usefulness of PROC hide for hiding procedures that use Microdrive commands.

If line zero has been deleted, BB will still regain control when the next new keyword is encountered, using the Interface 1 system variable VECTOR which allows possible syntax errors to be intercepted. However, Sinclair Basic keywords like RETURN that work differently under Beta Basic (in this case, faster) but have a normal syntax can cause problems.

The REM in line zero prevents the RAND USR statement being executed except from machine code. (REM doesn't prevent a DEF FN being found.)

****

Dear Sirs,

...once thing that puzzles/worries me is a delay of approximately 1 second in executing immediate command ...the delay does not occur every time and seems unpredictable...

Peter Erskine, Colchester

Eventually I got a reproducible example and found the problem. If you POKE 49412,0 things will speed up considerably. If you have not been troubled by this problem, don't make the POKE unless PEEK 49412 gives 223. The POKE may not be applicable to your version of the program.)

****

Dear Andy,

It would be nice if the ALTER attributes command could work with A specified window...

Geoff Stilwell, Borehamwood

Yes it would! A primitive version of what you want can be improvised for each third (top, middle or bottom) of the screen. Location 62608 holds "number of thirds to ALTER" and is normally 3, but it can be POKEd with 1 or 2. Location 62611 holds the most significant byte of "where to start ALTERinq" and is normally 88. Make it 89 to start with the middle third, or 90 to start with the bottom third. Beware! You will have to reduce the "thirds" count if you start lower or, the screen, - or you will ALTER things you shouldn't

****

Dear Dr. Wright,

Is there a way to get rid of the "Scroll?" prompt?

Well, in normal basic you can POKE the system variable SCR CT (23692) with zero every now and then, or you can use INPUT "" which has a similar effect. If Beta Basic is not in CSIZE 0, you can use POKE 51649,24 permanently get rid of the prompt ,even for listings. POD 51649,32 will return you to normal.

****

Dear Dr. wright,

I had hoped that CSIZE 4,4 would allow 42 lines instead of the normal 21. However the CSIZE operation for character height *h* gives a character height of 8*INT(h/8+0.5) and not h. The same problem occurs with say a height of 12 or 13; each line printed obliterates a part of the previous line.

F.A. Records, Camberley

The character size possible is limited by the screen resolution of the Spectrum CSIZE 4,4 leaves just too little room to design a legible character set. (CSIZE 4,0 uses a special character set.) CSIZE will squash graphics to half normal height, but normal characters are left alone. Their spacing will be modified - you can set a temporary height of 3 or 4 and obtain, subscripts

or superscripts using the cursor control codes. The various large characters are created by multiplying the size of the normal characters to 2,3,4, etc. times normal. You cannot make characters, say, 10% bigger than normal without having a redesigned character set - you can only use 2 or 3 or 4 pixels per original pixel, not 1.1! See the second paragraph under CSIZE in your manual. At certain CSIZEs it is useful to enter OVER 1 to minimise overlap problems. You can squeeze 29 lines onto the screen with - OVER 1: CSIZE 4,6 - with moderate readability if you use lower-case letters. OVER 1: CSIZE 4,7 gives 25 lines with good readability.

****

Dear Dr. Wright

I would be very grateful for some information about the location of one of, the Bata Basic 3.0 system variables. My problem is that I want to be able to return, at the end of a procedure, to the window in use Before the procedure, which uses, another window was called. I presume that the number of the current window must be store somewhere and it would be very useful if I could PEEK this.

....If you are thinking of transferring your attention to any other machine please let me Know and I will instantly go out and buy one. If you can do this much to the Spectrum it is hard to imagine what you could achieve with something a little more sophisticated.

Dr. R.H.K. Marsh, Northampton

Beta BASIC uses two window-number variables called WINDOW-T (address 57376) and WINDOW-P (address 57407). These hold the Temporary and Permanent window numbers which are usually the same. However, if you use WINDOW in a PRINT statement, only the WINDOW-T variable changes. WINDOW-P is later used to reset WINDOW-T to the permanent window value. PEEKing either variable will give zero, for window zero ,or a window number plus 128; i.e. bit 7 is set to 1. You may find it convenient to use AND to set bit 7 to 0 e .g.;

PRINT AND(PEEK 57407,BIN 01111111).

Another machine Well first I should say that Beta Basic was not written in a day, and it interacts very closely with the Spectrum ROM. So a version for another machine will not be written overnight. The easiest conversion will be for the 128k Spectrums, where the main advantage will be more free memory, and perhaps better speed. (The extra memory will allow more of the sluggish Spectrum ROM to be replaced by Beta Basic.) Of course any large-memory machine will allow even more commands to be added. It will be interesting to see how, Amstrad treats the Spectrum 128k. Their own, CPC6128 Is a nice piece of hardware, And the whole system is cleverly designed for RAM and R0M expandability. Unfortunately, Amstrad wi11 not divulge any details of their BASIC interpreter, the system variables, variable formats, etc. This means a lot of work, disassembling their ROM before work can even begin.

****
Dear Andy,

Could you please give me details of how the window-map data can be accessed for a given window and a translation of each of the bytes in the data?

G.M. Smith, Suffolk

I'm not sure how I'd access the window data from Basic - I'll tell you how the data is arranged and you can work something out! The DEF KEY and WINDOW data is held just above RAMTOP. RAMTOP is moved up or down as you change the amount of data. A block of DEF KEY data starts with the key character (lower case if it's a letter) and the length of the assigned string or line (2 bytes, less significant first). A window data block starts with the window number plus 128 and the length of the data (2 bytes, always showing a length of 12). The 12 bytes control the following, aspects of the specified window:

1 ATTRIBUTE COLOURS
2 ATTRIBUTE MASK (USED FOR INK AND PAPER 8)
3 P-FLAG (SETS OVER AND INVERSE)
4 OVER-2 FLAG
5 X COORDINATE OF PRINT POSITION
6 Y COORDINATE OF PRINT POSITION
7 LEFT-HAND EDGE COORDINATE
8 RIGHT-HAND EDGE COORDINATE
9 TOP EDGE COORDINATE
10 BOTTOM EDGE COORDINATE
11 WIDTH OF CHARACTERS
12 HEIGHT OF CHARACTERS

****
LAST WORD

Amazing Free Offer'

Beta soft has a large stock of sticky labels that are just about (well, exactly) the size of the oval hole in a cassette label. The ghosts of my Scottish ancestors will haunt me forever if the labels are thrown away, since they are ideal for labeling cassettes, discs, files, pots of jam, home-made wine, etc., etc. Just say you want some, include 6P in stamps (for the extra weight) and they will arrive with your next newsletter. Alternatively, send a stamped addressed envelope. You can have 20 labels, any colour you like as long as it's white.

We need contributions for the next Newsletter; please send some.

Scanned, Typed, OCR-ed, and PDF by

Steve Parry-Thomas 12th August 2004.

This PDF was created to preserve this
Newsletter for the future.

For all ZX Spectrum, Beta Basic
And www.worldofspectrum.org users

(PDF for Michael & Joshua)