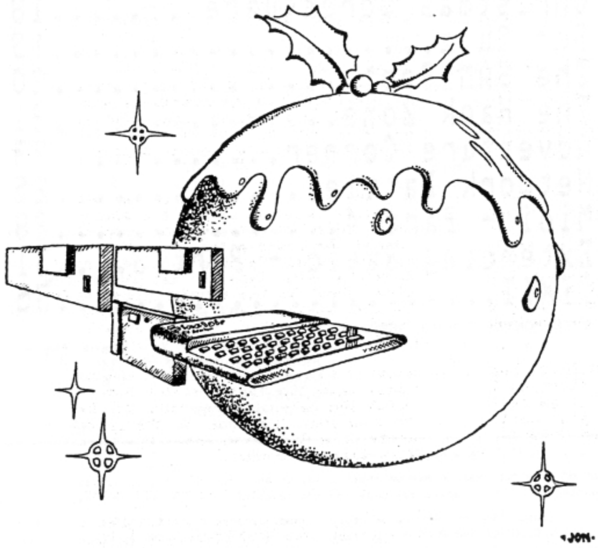
Vol 2 - No 5 December 1988.

FORMAT

THE MONTHLY MAGAZINE FOR
SPECTRUM, DISCiPLE & PLUS D USERS



Merry Christmas
and a Happy New Year

Vol 2 No 5.

CONTENTS

Dec 1988.

FORMAT is published by INDUG. 34 Bourton Road, Gloucester, GL4 OLE, England. Telephone 0452-412572. **DISCiPLE** and **PLUS D** are trade marks of MILES GORDON TECHNOLOGY, Lakeside Technology Park, Phoenix Way, Swansea, South Wales, SA7 9EH. Telephone 0792-791100. The DISCiPLE is marketed by Rockfort Products, 81 Church Road, Hendon, London, NW4 4DP.

Printed by D.S.LITHO. Gloucester. Tel:- (0452) 23198

The Editor Speaks

Welcome to the largest issue of FORMAT we have produced so far. You may also notice that the line lengths on many pages have been incresed slightly to enable me to pack even more onto each page, so FORMAT is now even better value. However, you a'nt seen nothing yet as they say, just wait and see whats coming next year..

Several people have phoned in the last few weeks to ask about the Format Software Service. The three most regular questions are - What has happend to it? Can I still get the tapes? Are there any new items coming through? - So I think an explanation is in order. First, as I've claimed many times before, I'm very overworked (and underpaid) and the software operation has proved very time consuming. Secondly, so many items have gone missing in the post over the last 4 or 5 months that FSS has been costing the user group money not making it. Bearing this in mind I stopped printing the FSS page after the September issue which had gone to print just at the start of the postal strike. What orders have turned up since I try to deal with as soon as I can, but I'm afraid its not as quick as it should be. Now the good news, in the new year I will be producing several discs of software, each with at least six programs (most of which has never been published before). They will be distributed for me by MGT who have far better handling facilities. However the conversion tapes, TASCON and ART STUDIO, will still be available through FORMAT. I hope the new arrangements will work out well.

I hope you enjoy the wide spread of articles we now print. Its good to see several new writers appearing in FORMAT this month, but I would like to see even more. Where are the small programs? Most of you listed programming as one of your uses of the Spectrum in our survey so why dont you send some in? I get very few Basic programs for publication, which is a great pity because thats just what a lot of people like to type in. So why not spend some of these long winter nights at your keyboard, and then share you efforts with other readers. Go on, FORMAT readers are waiting.

Finally this month let me express a very warm seasonal greeting to all our readers.

May you have the best Christmas ever, and may your New Year bring you health, wealth, happiness, and many more issues of FORMAT.

See you next year. **Bob Brenchley. Editor.**

NEWS ON 4

SUSIE

Over two years ago, a group of teachers in Birmingham who used Spectrums in their schools got together as an informal self-help group. At that time, Brum was hooked on RML computers. Consequently little official aid was available, and the group was able to arrange discounted repair facilities, and advice both to themselves, and later to interested parents on such topics as software or problems in running machines or using Basic. Spectrums were at that time found to be advantageous for a number of reasons: software companies had over-anticipated the Spectrum Educational market in 1984, and software was very cheap; the Spectrum was familiar to the parents and children who owned it, and single key entry was an enormous help to Junior School children with spelling difficulties. This ad hoc group has now put itself on a more official footing, and SUSIE (Schools Using Spectrums In Education) recently held an inaugural meeting. Further details from John Croghan, Head, St. Francis School, Teazel Avenue, Bournville, Birmingham.

SAM GOES TO BLACKPOOL

MGT has confirmed that a pre-production SAM will be on demonstration at the NORBRECK show in Blackpool on January 29th 1989. Alan Miles of MGT says 'We are looking forward to the show, we have been looking for a show in the north for some time. We are also looking forward to seeing some new faces as not everyone can travel to London for a ZX Microfair.'

FORMAT will also have a stand at the show so see you there.

+2a VANISHING TRICK

Popular Computing Weekly have gone to town over the Amstrad +2a release (anyone see the mention FORMAT got). I did tell you back in February 1988 that the machine was on its way so dont say you werent warned.

However a new mystery has now developed, the black cased +2a machines have vanished from the shelves in places like DIXONS. "Sorry sir we are out of stock... No I dont know when we will have more in... How about a +3?" was the response in Bristol last week. Now I cant believe that they really have sold all the +2a machines, could Suger's gang have had second thoughts? I doubt it, but there is a rumour that they may be repacking machines so the poor purchaser can tell what machine he is getting for his hard earned money

FIXIT FOR THE +2a

Last month I promised you more details of the MGT 'Twister' board. Its now available and you will find a review of it in this issue.

- - - - - -

If you have any news items you want to pass on then send them in. Please mark the envelope NEWS in the top left corner.

YOUR LETTERS

Dear Editor,

Can I ask you or your other readers for help with software. I am very new to the Spectrum (I got mine secondhand in March) and there seems to be little serious software advertised. I know, from looking at a few old mags that came with my 48k, that there used to be a lot around but where is it now?

I'm looking for toolkits and other aids to programming as well as 'useful' programs.

Yours Sincerely, J.N.Harris.

Most small software producers were forced out of business due to the very high price of advertising in the glossy mags. I hope many can be encouraged back now that FORMAT is around. If readers will send in details of software companies that they know of then I will try to print a directory. ***Ed.***

Dear Editor,

I like FORMAT very much and look forward to each months issue with bated breath. But this raises the only moan I have, when can I expect each issue. Isn't it possible to tell us the publication date in the previous issue?

Yours Sincerely, Andy O'Conner.

I did try that at one time but then if I was a few days late for some reason I was flooded with telephone calls. I try to get FORMAT out around the 20th of the month but there are so many things that this depends on that it is impossible to guarantee an exact date. Still, by May of next year I intend to try to bring forward publication day to around the 10th. ***Ed.***

Dear Editor, *STAR*LETTER* *STAR*LETTER*

Is it just me? Or do other readers suffer from very poor TV pictures from a +2? My machine has been replaced twice by my local BOOTS but still the picture is bad. Surely after so many years of computer production we should expect at least a reasonably clear TV output.

Yours Sincerely, Dave Morgan.

Dear Editor,

Thank you for the interesting articles on BETTER BASIC (FORMAT Vol 1 Number 12). But when are we going to have some more from Ken Elston?

Yours Sincerely, S.T.Little.

The following month saw the start of a regular feature from Clyde Bish which has covered many of the things Ken had planned for his articles. As Ken was moving house at the time he decided to drop his plans until early 1989 when he will be back with more on Basic and another part to his Tasword 2 upgrade ***Ed.***

Letters printed may be edited for length or clarity. The writer of each months STAR LETTER wins an EXTRA 3 months subscription.

MAKING THE MOST OF THE UDG

By:Clyde Bish.

By now enthusiastic adventure writers (and basic programmers in general) will have saved hundreds of bytes in their programs, and are wondering what to do with all that free space. Why not brighten things up a little with some graphics? What immediately springs to mind is full colour, hires Hobbit-style pictures or umpteen thousand different views around the Land of Midnight. All in good time. Let's not forget the humble ascii character.

What can you do with Uncle's character set? Answer - quite a lot if you've an inventive mind. (Ex-ZX81 users might care to remember the ingenious graphics in Automata's "Can of Worms"). Let me help. Enter the following lines. (The [] mark combination keypresses, just enter whats between them.):-

```
  10 LET h$="[E-MODE CAPS-SHIFT 2]#
  20 LET b$="[E-MODE CAPS-SHIFT 1] [E-MODE CAPS-SHIFT 7][E-MODE 1][G
RAPHIC 8][GRAPHIC 8]=========[E-MODE 6][E-MODE CAPS-SHIFT 3][E-MODE
CAPS-SHIFT 9]@[E-MODE CAPS-SHIFT 7][E-MODE 1][E-MODE CAPS-SHIFT 8]::
[E-MODE CAPS-SHIFT 2][E-MODE 7]0"
  30 LET s$=CHR$22+CHR$17+CHR$22+h$+CHR$22+CHR$18+CHR$8+b$+CHR$22+CH
R$19+CHR$2+h$
```

If you've been following this series you'll know that the strange keying sequences have been incorporating colour and flash control codes into the string. (If you haven't (tut-tut!) you'll find the table on page 87 of your manual helpful).

Now RUN. When you get the O.K. message, try CLS : PRINT s$, and you'll have the mystic Sword of Power appear before you, all made with ascii characters complete with flashing magic Symbol of Gora - probably the best use for the deformed AT sign!

In a similar way you can produce quite a respectable Robot by printing an "o", "■" or "O" and "1" in a vertical line. The following line produces a bigger version.

```
  10 PRINT "O": PRINT "[GRAPHIS CAPS-SHIFT 8]": PRINT "[GRAPHIS CAPS
-SHIFT 8]": PRINT "[GRAPHIS 5]": PRINT "[GRAPHIS CAPS-SHIFT 2]"
```

How about a cache of silver rings? Try RUNning:-

```
  10 FOR f=1 TO 10: PRINT AT f,11; PAPER 0;"(12 spaces)":NEXT f
  20 FOR f=1 TO 30: PRINT AT RND *7+2, RND *9+12; PAPER 0; INK 7;"o"
: NEXT f
```

Line 10 prints a black background, whilst line 20 prints a random scattering of rings over it. Now CLS, edit line 20 to read:-

```
20 FOR f=1 TO 30: PRINT AT f,11;INK 2;"[GRAPHIS 1]"
```

and RUN 20 and you have a find of "Blood-Gold".

Jewels need a slightly different technique. For a hoard of diamonds replace line 20 with:-

```
  20 FOR f =1 TO 80: PLOT RND *80+96, RND *64+96: NEXT f
```

and RUN. (If you want rubies, set INK to 2 and RUN 20).

Before leaving the ascii codes let's look at the OVER command. This PRINTs one character on another without blanking it out as normally happens. It works in a rather strange way. INK on INK gives PAPER colour, as does PAPER on PAPER, but INK on PAPER gives INK. What this amounts to is that by over printing characters, new shapes can be produced. Sort of pseudo-user defined graphics. So overprinting * on > gives an aeroplane-like shape, c on o gives a "bullet", and * on ! gives a rocket flame. Try it out for yourself but be warned, there are tens of thousands of combinations.

Turning now to real udg's, this opens a complete new spectrum (and without invalidating the guarantee!). You can quite literally produce what you want, where you want it on screen. As the individual udg's are quite small they really need to be used in groups to get an illustration of any size. This causes a problem. You only seem to have 21 available (Graphics A - U). Even without the manipulation of the machine's memory we'll look at later, this is not so restrictive as it seems. It is quite possible, with forethought, to use some udg's for more than one purpose. We'll look at this in detail in a later article, but for the present let's rethink Uncle's chessmen udg's (page 72 in your manual). These are far too small to be of any real use, but if they were twice the size it would be a different matter. That causes a problem. There are six pieces, each of four udg's. That makes 24, and there are only 21 available. The solution to the problem is quite simple. Use two as the base, and link these with different tops for each piece. This way you will only need 6*2+2 = 14 udg's leaving some spares. (So purists could have a smaller base for their pawns).

But you're not actually restricted to just 21 udg's. You can have as many as you like (at a cost of 8 bytes each), provided you inform the machine of the fact. Amongst those mysterious System Variables, the "house-keeping" file of the machine's memory, there is one called UDG at addresses 23675 and 23676. The information these hold is the address of the first udg byte. If you:-

```
  PRINT USR "a" (ENTER)
```

you'll get the number 65386 on screen (on a 48k Spectrum). If you:-

```
  PRINT PEEK 23675, PEEK 23676 (ENTER)
```

you'll see the numbers 88 and 255. This is the way the machine holds numbers larger than 255. If you multiply 255 by 256 and add 88 you'll get that 65386 again.

If you POKE the UDG System Variable with other numbers you will cause the machine to look elsewhere for the start of the udg's, and

you can do this as often as you like. I'll prove it to you. Type in this little and run it. (The capitals in line 200 are udgs and must be entered in Graphics mode).

```
  10 FOR f=60000 TO 60007: READ a: POKE f,a: NEXT f: DATA 1,3,7,15,3
1,63,127,255
  20 POKE 23675,96: POKE 23676,234: PRINT "AAAAA"
```

The first section sets up a udg file of triangle shapes at a new address, 60000. The second section redirects the UDG systems variable to this address, before the udgs are printed as normal. You're still limited to a maximum of 21 udg's at a time, but you can have as many sets as you like, simply by redirecting UDG to the start of each set as required. You could use this idea to illustrate each of your intrepid Adventurer's finds by a 5 column by 4 row graphic. To use this system, first design your illustration on squared paper, sorting out your INK and PAPER colours as well, plus BRIGHT or FLASH if you need them.

The easiest way to translate your masterpiece into a form the computer can use is by using a Character Generator program if you have one. If not you'll have to work out and feed in the data yourself, and so construct each character. Don't forget to work across each of the rows starting with the top left and ending on the bottom right. Remember you have to use all 20 characters each time, even if you just make them blanks. You don't need the 21st character. SAVE each set with:-

```
  SAVE "title"CODE USR "a",160
```

Now you have to make all those saved udg sets - you can have up to 30 if you start at address 60000 as above - into one long length of code. You can do this using the following program:-

```
   1 REM capitals are udgs
  10 LET g$=CHR$ 22+CHR$ 10+CHR$ 0+"ABCDE"+CHR$ 13+"FGHIJ"+CHR$ 13+"
KLMNO"+CHR$ 13+"PQRST"
  15 LET a=60000: LET c=1
  20 INPUT "Title of code ";(c);"? (ENTER if end)"'t$
  30 IF t$="" THEN  GO TO 200
  35 LOAD t$CODE a,160
  40 RANDOMIZE a: POKE 23675,PEEK 23670: POKE 23676,PEEK 23671: PRIN
T g$
  50 LET k=0: FOR f=22848 TO 22944 STEP 32
  60 FOR n=0 TO 4: LET k=k+1: POKE f+n,7
  70 INPUT "Attribute value?",v
  80 POKE f+n,v: POKE a+160+k,v
  90 NEXT n: NEXT f: LET a=a+160
 100 PAUSE 100: CLS : GO TO 20
 200 INPUT "Title for code?",t$
 210 SAVE t$CODE 60000,c*180
 220 PRINT "VERIFY": VERIFY t$CODE
```

Type it in, then RUN. Answer the prompt with the name of the first set, then load it in. After loading, the program will ask for an attributes value of each udg. Work this out using the table in Fig 1. For example, for black INK on green PAPER you would enter 33. Add 64 to make it BRIGHT, and 128 for FLASH. (Remember you can only

		INK							
		0	1	2	3	4	5	6	7
	0	-	8	16	24	32	40	48	56
	1	1	-	17	25	33	41	49	57
	2	2	10	-	26	34	42	50	58
PAPER	3	3	11	19	-	35	43	51	59
	4	4	12	20	28	-	44	52	60
	5	5	13	21	29	37	-	53	61
	6	6	14	22	30	38	46	-	62
	7	7	15	23	31	39	47	55	-

Fig 1.

have one INK and one PAPER colour per character). You'll see each udg displayed change to the attribute value so you can see what the final result will be. Now repeat the process with the second udg set, and so on until the last set has been dealt with, then reply to the prompt with 0 and you'll be able to SAVE the whole collection of udg and attribute bytes from address 60000.

A subroutine to make use of this code in your adventures would look like this:-

```
 100 LET f$="001A Boot": GO SUB 1000: STOP
1000 LET c=VAL f$( TO 3)*180+59820: RANDOMIZE c: POKE 23675,PEEK 236
70: POKE 23676,PEEK 23671: PRINT g$,f$(4 TO ): LET c=c+160: FOR f=22
848 TO 22944 STEP 32: FOR n=0 TO 4: POKE f+n,PEEK c: LET c=c+1: NEXT
 n: NEXT f: RETURN
9999 LET g$=CHR$ 22+CHR$ 10+CHR$ 0+"ABCDE"+CHR$ 13+"FGHIJ"+CHR$ 13+"
KLMNO"+CHR$ 13+"PQRST": REM capitals are udgs
```

Type this in and RUN 9999. This sets up a string variable g$ as the complete illustration with the udgs in the correct places to start printing at row 10, column 0. When you get the O.K. message you could delete 9999. This information is now in the Variables area in memory and you don't need the line taking up valuable space.

You will need a statement like line 100 in your program wherever you want to call an illustration to screen. The variable f$ holds the information for the illustration number (in this case 1 - it has to be 3 figures) followed by the title - A Boot.

The subroutine starting at line 1000 uses this information to do all the hard work. Line 1000 itself is interesting. Remember you have to tell the System Variable UDG where the udg's start in that strange two-number, multiples it by 180 and adds 59820, so picture 1 begins at 60000, picture 2 at 60180, picture 3 at 60360 etc... This value resets another System Variable called SEED using RANDOMIZE in the two-number form UDG needs, so when you PEEK it, the values needed are there. Crafty! The rest of f$ is printed under the picture.

So far the program has only printed the picture on the existing screen colours for INK and PAPER. Now to get the colours right. The way the subroutine does this is to POKE the attribute values following the udg bytes into the correct addresses in the ATTRIBUTES file. This starts at 22848 with the information for the character at row 10, column 0, and moves across each row, jumping to the next when necessary. This is the function of the nested loops f and n.

Obviously you can have smaller illustrations (or larger ones, but I'll leave you to work that one out). The important point is they must all be the same size and print in the same place on screen.

Next time we'll look at ways of improving these techniques, and of economising on larger illustrations. 'Til then, happy UDG-ing!

BETA BASIC

"THE ROSE" - A PATTERN GENERATING PROGRAM.

By: ANDY WRIGHT.

The Rose is a pattern-generating method used in some computer graphics demos. The mathematician who devised it (Peter Maurer) made it public quite recently, and I have written a Beta Basic program to demonstrate it. The version below has been improved in line with suggestions from Beta Basic Newsletter readers. Something similar will probably be used as part of a SAM demo program. SAM can reproduce even complex examples of such patterns in a few seconds - the swirling effect as a pattern is drawn is really quite impressive!

The points that the program DRAWs TO (this is a DRAW TO a specified pixel, rather than the usual DRAW by a specified amount from the current position) all have x and y coordinates between -1.0 and +1.0, so it is convenient to alter the scale of the graphics coordinate system and the location of the origin. Line 10 does this by assigning values to special variables that are always present in a Beta Basic system (even after CLEAR or RUN). XOS and YOS set the x and y offsets of the origin, and XRG and YRG set the x and y range which will fill the whole screen. After line 10, PLOT 0,0 will plot near the centre of the screen, and PLOT -1,-1 will be low down on the left.

RNDM(number) is a faster (2.5 times) version of RND that gives random whole numbers between 0 and the number specified. Here it is used to set two numbers that specify each pattern. MOD(A,B) gives the remainder after A is divided by B. It is used at line 40 to avoid certain kinds of symmetrical patterns taking longer than they need to, and at line 90 to give a remainder in the range 0-359 degrees. Sines and cosines have to be calculated in lines 90 and 100. Ordinarily, this would greatly slow down the pattern generation, since SIN and COS are very slow, but Beta Basic provides SINE and COSE, which are slightly less accurate, but 6 times as fast. Once a pattern is complete, the program pauses for you to admire it. Press any key to generate the next pattern.

```
   1 REM "THE ROSE"
   5 LET k=PI/180
  10 LET xrg=3,xos=xrg/2,yrg=xrg*176/256,yos=yrg/2
  20 DO
  30   LET n=RNDM(178)+1,d=RNDM(178)+1
  40   IF MOD(n,2) AND MOD(d,8) THEN
         LET b=180
       ELSE LET b=0
  50   PRINT "N=";n'"D=";d
  60   PLOT 0,0
```

```
 70   LET a=0
 80   DO
 90     LET a=MOD(a+d,360),t=k*a,r=SINE(k*MOD(n*a,360))
100     DRAW TO r*SINE(t),r*COSE(t)
110   LOOP UNTIL a=b
120   PAUSE 0
130   CLS
140 LOOP
```

If you decide to try to write a (slower!) version of this program in standard Spectrum Basic, it may be useful to know that you can DRAW to a specified x,y point using:

```
       DRAW x-PEEK 23677,y-PEEK 23678
```

=+=+=+=+=+=+=+=+=+=+=+=

SMALL IS BEAUTIFUL

A MACHINE CODE SCREEN COPY ROUTINE

By: Carol Brooksbank.

I wrote this program because I wanted to make small bookplates, printed on peel-off labels, for a library (see Fig 1) and logos for headed notepaper. I needed to use the full width of the screen to get enough detail in the design, but the smaller of the +3's COPY routines covers threequarters of the width of a sheet of A4, and about one third of the depth. It also refuses to copy the bottom two lines of the screen.

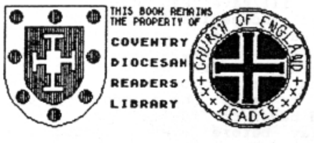


Fig. 1. SCREEN DUMP FOR LABELS

A small screen dump was needed, and this is the little routine that produces it. It was written for the +3, but it will work on any Spectrum model, provided it is hitched up to a dot matrix printer with Epson compatible bit-image graphics.

The machine code only handles the printing of one horizontal row of character squares. The repeat loop which prints all the rows is handled from BASIC, so that if you don't want to print all the screen you can change the number of times the loop repeats. If you want to match peel-off labels, for instance, they are only 14 screen rows deep in the size of screen dump I use.

There is a problem to be overcome before you can print a screen dump using bit-image graphics. Each character square consists of 64 pixels (8 bytes). The bytes held in the Spectrum memory are the HORIZONTAL bytes - most significant bit (MSB) 7 on the left, least significant bit (LSB) 0 on the right. The printer requires the VERTICAL bytes, MSB 7 at the top, LSB 0 at the bottom (see Fig 2).

We have to convert the horizontal bytes in the memory to the required vertical ones. The key is the RL instruction. The diagram shows the effect of that instruction on the byte held in a register (see Fig 3). Bit 7 moves to the carry flag, all the other bits move one step to the left, and the original value of carry goes to bit 0.

So, if we point HL to the top byte in the square, and execute RL (HL), the carry flag will hold bit 7 of the top byte. Now execute RL D, and that same bit will be transferred to bit 0 of D. Point HL to the next byte down in our square, repeat the operation, and that first bit will take one step to the left and be now bit 1 of D. Repeat the whole

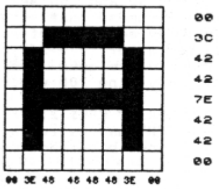


Fig. 2. HEX VALUES OF VERTICAL AND HORIZONTAL BYTES OF CHARACTER SQUARE

thing a total of 8 times, and D will be holding a byte made up of the number 7 bits from all of the horizontal bytes, in the right order. Since that is what we wanted, we can print it. As a by-product, all the original bits 6 of the horizontal bytes will have stepped to the left, and be waiting in the bits 7 positions for us to repeat the operation to get the next vertical byte.

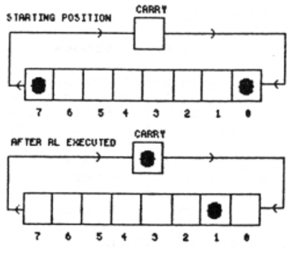


Fig. 3. EFFECT OF THE RL INSTRUCTION

Unfortunately, at the same time. all the bits 0 of our horizontal bytes will have been corrupted, because the contents of the carry flag will have been copied to them. As each successive vertical byte is printed, the character square becomes more corrupted. That means we cannot work directly on the screen display. You would only be able to make one screen dump, and would have to re-load the screen display before every printout for multiple copies.

So, before each character square is printed, its horizontal bytes are copied to a workspace where we can manipulate and corrupt them without affecting the screen display.

And that is really all there is to the machine code. An outer loop repeats the character square dump 32 times, to copy a complete screen row. The notes in the listing explain in detail what each instruction does. The little routine labelled NXDOWN is a separate routine which finds the top byte of the lefthand square of the next row down, and is called from BASIC each time the program moves on to the next row of squares.

The BASIC controls the bit-image mode being used, and this in turn, affects the appearance of the finished screen dump. Line 30, which sets up the interface to receive control codes, would have to be modified if you are not using the +3 interface (see table 1). Consult your interface instruction book to find the correct method.

Lines 40 and 200 select the mode in use, and the proportion. Line 200 assumes that the printer supports the useful ESC "*" control, which selects one of several available bit-image modes by entering a number after "*". If you have ESC "*", the value given to "mode" in line 40 selects the bit-image mode. If your printer does not have ESC "*", line 200 would have to be changed to:

```
LPRINT CHR$ 27;"K"; CHR$ 0; CHR$ dots;
("K" may be replaced by "L","Y" or "Z" to select
other bit-image modes.)
```

The value of "dots" governs the number of times each vertical byte is printed, and the value of "margin" governs the distance of the printout from the lefthand side of the page. Line 70 sets the margin width and selects 22/216 inch line spacing (remember to reset your printer after using this dump routine). Line 90 loads the screen, and line 110 stores the address of the top lefthand byte of the screen in the program variable ACRNPOS for the start of the routine. Line 120 governs the number of lines copied, so change this line if you want to print less than a full screen dump. Line 150 points SCRNPOS to the next row down. The subroutine at 200 selects the bit-image mode, prints the line, and performs a carriage return. Line 9999 saves the BASIC and machine code.

Changing the mode and dots values gives a whole range of screen dumps in varying sizes and proportions. I put together a demo screen with a couple of type faces, some geometric shapes including a true circle, and some small sprites, and the illustrations give you an idea of the effects of the changing proportions on the different features. Mode 3, dots 3, comes closest to a perfect copy. There are dozens of combinations of values available, giving a wide variety of results, but I will leave you to play around and investigate them for yourself!

The source code was written using the Laser Gènius Assembler which allows long labels and comments which I like. You may need to make slight alterations when using different assemlers

The Source Code

```
 1 *SCREEN    ON
   *LIST      ON
 2 *PRINTER   ON
   *LLIST     ON
10          ORG  60000
20          LD   A,3 ; Output to printer
            CALL #1601
            LD   HL,(SCRNPOS) ; Fetch address of top byte of first
                                square
            LD   B,32 ; Number of squares in a row
30 LOOP5:PUSH BC ; Save number of squares to do
            PUSH HL ; Save square we are on
            LD   B,8 ; Number of bytes in square
40          LD   DE,BYTES ; Address of list of bytes we are converting
                               to vertical bytes
   LOOP1:LD   A,(HL) ; Fetch value of byte in square
            LD   (DE),A ; Store it in list
            INC  H ; Point to next byte down in square
            INC  DE ; Point to next address in list
            DJNZ LOOP1 ; Jump back if more bytes to list
50          LD   B,8 ; Number of vertical bytes to print
   LOOP4:PUSH BC ; Save number of bytes to do
            LD   HL,BYTES ; Address of list of horizontal bytes
            XOR  A ; Clear the D register
            LD   D,A
            LD   B,8 ; Number of bits in a byte
60 LOOP2:RL   (HL) ; Move bit 7 of listed byte to carry, and move
                        bit for next vertical byte to bit 7 position
            RL   D ; Transfer carry to bit 0 of D register, and move
                        all other bits in D to left.
            INC  HL ; Point to next address in list of bytes to do
            DJNZ LOOP2 ; Jump back unless all 8 bits copied to D
70          LD   A,(DOTS) ; Fetch number of times each vertical byte
                              is to be printed
```

```
        LD   B,A ; Use this figure as counter
   LOOP3:LD   A,D ; Transfer vertical byte to A register
        RST  16 ; Print it
        DJNZ LOOP3 ; Jump back if it is to be printed again
        POP  BC ; Fetch number of bytes left to do in this square
        DJNZ LOOP4 ; Jump back unless square finished
 80     POP  HL ; Fetch address of screen square we have just done
        POP  BC ; Fetch number of screen squares left to do in
                  this row
        INC  HL ; Point to next screen square
        DJNZ LOOP5 ; Jump back unless row completed
        RET   ; Return to BASIC
 90 BYTES:DEFS 8 ; List of bytes in a screen square. These will be
                 rotated and  so values changed as each vertical
                 byte is printed.
   SCRNPOS:
        DEFS 2 ; Address of first byte of screen row  we are
                 printing. Poked from BASIC as 16384 at start.
   DOTS :DEFS 1 ; Number of times each vertical byte is to be
                 printed. Poked from BASIC
100 ; Subroutine moves SCRNPOS to the top byte of the next row down
   NXDOWN:
        LD   HL,(SCRNPOS)
        RR   H
        RR   H
        RR   H
        LD   BC,32
        ADD  HL,BC
        RL   H
        RL   H
        RL   H
        LD   (SCRNPOS),HL
        RET
110 *PRINTER  OFF
```

THE BASIC PROGRAM.

```
  10 CLEAR 59999
  20 LOAD "dumpcode"CODE 60000
  30 FORMAT LPRINT "U"
  40 INPUT "mode";mode: INPUT "dots";dots
  50 INPUT "margin";margin: INPUT "screen to print";s$
  60 POKE 60067,dots
  70 INPUT "number of copies";n
  80 LOAD d1;s$ SCREEN$
  90 LPRINT CHR$ 27;"l";CHR$ margin;CHR$ 27;"3";CHR$ 22
 100 FOR r= 1 TO n
 110 POKE 60065,0: POKE 60066,64
 120 FOR Q=1 TO 24
 130 GO SUB 200
 150 RANDOMIZE USR 60068
 160 NEXT Q
 170 LPRINT CHR$ 12;
 180 NEXT r
 190 STOP
 200 LPRINT CHR$ 27;"*";CHR$ mode;CHR$ 0;CHR$ dots;
 210 RANDOMIZE USR 60000
```

```
 220 LPRINT
 230 RETURN
9999 SAVE "scrdmp" LINE 10: SAVE "dumpcode"CODE 60000,91
```

Table 1 - Other interfaces.

Replace line 30 of the basic program to match your interface.

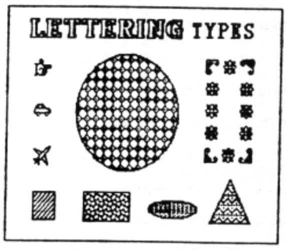
DISCiPLE / PLUS D = 30 POKE @6,1

INTERFACE 1 = 30 FORMAT "b";baud rate: OPEN #3;"b"

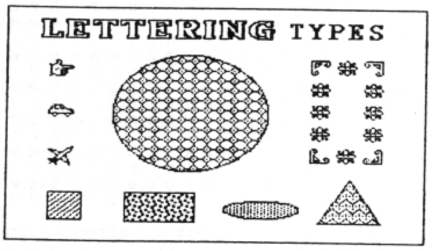
CODE POKER

```
  10 FOR I=1 TO 91: READ N: POKE I,N: NEXT I
  20 SAVE "dumpcode"CODE 60000,91
  30 DATA 62,3,205,1,22,42,161,234,6,32,197,229,6,8,17,153,234,126
  40 DATA 18,36,19,16,250,6,8,197,33,153,234,175,87,6,8,203,22,203
  50 DATA 18,35,16,249,58,163,234,71,122,215,16;252,193,16,230,225
  60 DATA 193,35,16,210,201,0,0,0,0,0,0,0,0,88,1,42,161,234,203
  70 DATA 28,203,28,203,28,1,32,0,9,203,20,203,20,203,20,34,161
  80 DATA 234,201,31
```

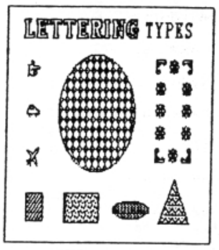
SOME EXAMPLES



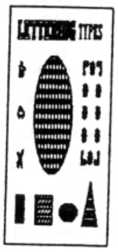
MODE 6, DOTS 1.



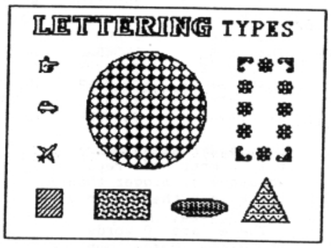
MODE 2, DOTS 2.



MODE 2, DOTS 1.



MODE 3, DOTS 1.



MODE 3, DOTS 3.

ALL EXAMPLES SHOWN HALF ACTUAL SIZE

COMPETITION WORDSQUARE

```
J R J X I O D W R E I N D E E R C N M F R Q R T C C G Z
P A N T O M I M E S V R V Y D A T E S H T S O C R H N V
K T C T F A I R Y I E A Y T P T Q S V H Y E B C E U I G
S T U N T S E H C K E Z S U Z J A X M L W L I A T R P N
L E S N I T I H C S W V N Y A Z X X L V I B N R T C P W
Y K E K A C Y A W F P C E A M W Z O M Y N U J D I H O J
V H C M N D R T D F H Y E N M I H C G M T A N G L H H K
I D H S U C O S N O W B A L L A N H F S E B Q O G U S N
G R E T T E B P T I N O D S S R E D A E R R T A M R O F
N Q L J C S E S Z Z Y G C M E T A L O C O H C A T N A S
O I G L M A N G L F L R M N R N A P I Z R A M F N O S D
I L N I B U N O A I I R D E F F V A G E P M R T L M N J
S D A G C E B D I B I N E M T E A W R R C O P K V C L X
I L P U D G G A L T Y U L E T I D E E K R C S O U G Y R
V R E F W P O V L E A P M C T Z W S E S U A T I S G R S
E R O J J C L M I L N R P S D V E V T G D L U S Y O T T
L Z L G Q J N U G N O L O I N N M M I V O E F X K J B O
E Y O V C D A Y H L Y O N C T O A A N Q L N F P U E P C
T S R M G W E E T J C N N S E A W B G R F D I P A P L K
E Z A R W G H K S K E Y C S P D X M S H R A N Y Q F R I
C S C M E P Q R G R Y Q W X V T G D A F E R G Z I B X N
Q Z O Z T M N U K T A M J H B O S X X N S L L E B Z A G
O R J O K S I T S D H G R O T T O T N Y A A Y A Y P L M
Q E G M G T I O Q G I T T H A F I F U O U M Q J T A D O
C P E R A G R R J S L E D G E M W O Q N B G K W P P P R
B M K A L F N A H U E O T E L T S I M N C B I Y G O W J
Y A N T J D G F O C U E J O S S V N M O I L I I C T U E
P H Y S I A D V E N T Y D N A C O I V Y I X J R K M A J
```

FORMAT's CHRISTMAS PRIZE WORDSQUARE

CHRISTMAS
TELEVISION
DECORATIONS
CHESTNUTS
CALENDAR
PRESENTS
MISTLETOE
REINDEER
PANTOMIME
GREETINGS

SHOPPING
STOCKING
YULETIDE
STUFFING
BALLOONS
SNOWBALL
CHIMNEY
CRACKER
BAUBLES
GLITTER

ADVENT
DINNER
WINTER
FROSTY
SLEDGE
SANTA
HOLLY
GROTTO
RUDOLF
CANDLE

ANGLE
MERRY
BELLS
EVE
FAIRY
RIBBON
HAMPER
CHURCH
LIGHTS
PUNCH

IVY
TOYS
CANDY
DATES
GOOSE
CRIB
TREE
CAKE
CARD
STAR

MARZIPAN
CHOCOLATE
SNOWMAN
TURKEY
ROBIN
CAROL
TINSEL
LOG
PUD
NUTS

Last Christmas we ran a competition wordsquare which, judging from the number of replies, proved very popular. So here's another wordsquare, bigger than last years so it should give you something to do when there's only repeats on the telly.

There are 60 words (all related to the festive season) to find in the grid. Words run FORWARD and BACKWARD, UP and DOWN, even DIAGONALLY, in fact in all 8 directions. **SNOWBALL** is marked to give you a start so there's just 59 left to find and circle.

Geoff Bobker of ZX GUARANTEED has very kindly donated 30 copies of his book 'TRADE SECRETS' as prizes. Send your completed entry (photo copy accepted, but only 1 entry per member allowed) in an envelope marked 'WORDSQUARE', to our usual address To arrive no later than first post on Monday 30th January 1989. To make it fair for our overseas members 6 of the books have been set aside for them, Overseas entries should arrive by 31st March 1989.

All correct entries will go into the hat on the closing date and the draw will be made. Winners will be notified by post, as usual in these things the Editor's word is law.

- - - - - - - - - -

And now a little quiz for you. No prizes, just a bit of fun. See if you can get all of them right

1. What was the name of Sir Clive Sinclairs first company?

   a) Sinclair Research b) Sinclair Radionics c) Uncle Clive's Ltd

2. What was the name of his first computer?

   a) ZX Spectrum b) ZX80 c) MK14 d) Oric

3. What was the first product from Sinclair Research?

   a) Hi-Fi equipment b) Pocket calculator c) Black watch d) ZX80

4. In what year was the ZX Spectrum launched?

   a) 1980 b) 1981 c) 1982 d) 1983

5. How much RAM did the original ZX81 have?

   a) 16k b) 4k c)48k d) 1k

6. What is Sinclairs latest computer called?

   a)ZX88 b) Superbrain c) Z88 d) QL+3

7. Who invented the DISCiPLE and PLUS D?

   a) Bruce Gordon b) Noel Gordon c) Flash Gordon

8. Which software company had early links with Sinclair?

   a) Psion b)Python c) Nylon d) Crayon

9. Including the Dec'88 show How many ZX Microfairs have there been?

   a) 6 b) 17 c) 38 d) 28

10. Which is the best magazine available for Spectrum users?

    a) Atari User b) Crash Bang Wallop c) FORMAT d) The Beano

For answers turn to page 27.

THE SAM SPOT

By: Bob Brenchley.

SAM is the super, Z80 based, computer that MGT will be launching next year. As I am working very closely with MGT on the project, FORMAT will usually be first with news on SAM. As I am pledged to secrecy I can not answer questions (by letter or phone) regarding the inner workings of SAM. However, after clearing the subject matter of articles with MGT, I will be giving you the full details on SAM over the coming months. Please remember that some fine details are still subject to change.

First bit of news this month is that SAM has been given its full name at last. It will be know as the **SAM COUPE.** It will be the first of a family of SAM computers which are being designed to grow with the user. You will be able to upgrade your existing SAM as each new member of the family is launched. This will, once and forever, remove that feeling we have all had at some time in the past - Oh if only I had waited.

Two versions of the SAM Coupe will be available from its launch in April next year. The base model, which we have talked about before, will be a cassette based with 256k of RAM. It will sell (provided RAM chips don't soar in price again) for £149.95. A disc based version will also be released at more or less the same time this will have a single 1/3rd hight drive (3.5") fitted in the front edge of the machine. At around £220 this will be fantastic value.

DISCiPLE & PLUS D users will still be able to use their existing disc systems with SAM using a small adaptor supplied by MGT. Discs created by a DISCiPLE or PLUS D will work with SAM's enhanced DOS. However the reverse will not always be true because SAM will support several new file types including sub-directories. Having said that INDUG will be looking at ways to make some of these advanced disc features available to members if at all possible.

I've also seen the artists impressions of the case and keyboard which have been done by the Nick Holland Design Group. This company was recommended by the UK Design Council when they were aproached by MGT for guidance on the external design of SAM. I hope to have drawings of the computer for you next month, I think you will like it (I do). The keyboard has 71 keys (including 10 function keys) and is quite PC like in both feel and looks. Production of SAM will, at least for the first few months, be done in India. Bruce Gordon has already made several visits and high level negotiations with the Indian authorities are nearly complete. This will also open the vast Indian market to MGT, something no other home computer manufacturer has been able to do.

Next month I hope to bring you more details of SAM so keep reading. Remember, if you want to be the first with news of SAM, FORMAT is for you.

HACK-ZONE

By: Hugh J. McLenaghan.

This month I have written a Cat-Sort routine. This is taken from an idea by James Willsher in Vol 2 No 2. Although this routine does the same thing, it is written in machine-code and has some extra features.

Here is the program, you enter it using the Hexloader which was printed in Vol. 2 - No. 2.

```
65000: CDF8FDCD7EFFCDDF7        65264: 065BFE02D821017FC
65008: FECDEFFECDAAFFC99        65272: 3EFF22085B32075B1
65016: CD02FE3AA9FFB720A        65280: 3A075B3C32075B479
65024: F7C9110100AF32A95        65288: 3A065BB8C82A085B0
65032: FFD5CD84FE28183A9        65296: 2422085BCD19FF184
65040: D61CB72815D3BBD19        65304: E7E5D1220B5B3A079
65048: 1C7BFE0B20EB1E01E        65312: 5B320A5B2A0B5B24E
65056: 147AFE0420E3C9AF5        65320: 220B5B3A0A5B3C329
65064: 18023E01D3BB32007        65328: 0A5B473A065B3DB85
65072: 5BED53015BFE01288        65336: D82A085BED5B0B5B8
65080: 0ADBBB3AD61CD3BB3        65344: 1ACDD7FF46CDEBFFA
65088: B7202FD11C7BFE0BB        65352: B828133802l8D52BD
65096: 20071E01147AFE04C        65360: 1B06001A4F7E12717
65104: C8D5CD84FE20183AC        65368: 231310F718C606092
65112: D61CB72015D3BBD19        65376: 23131ACDD7FF46CD0
65120: 1C7BFE0B20EB1E01E        65384: EBFFB838082004235
65128: 147AFE0420E3C9AF5        65392: 1310EF18AF2A085BA
65136: 18023E01D3BB32031        65400: ED5B0B5B18D11101F
65144: 5BED53045BCD8DFEA        65408: 00210080D5E5CF3F0
65152: D1C309FECF3FDBBB1        65416: D121D61BD50100026
65160: 3AD61BB7C921D61BC        65424: DBBBEDB0D3BBE1245
65168: 84671100D0CDD5FEE        65432: 24D11C7BFE0B20E40
65176: ED5B015BD5CF3F3A9        65440: 1E01147AFE0420DCB
65184: 005B21D61B8467E5B        65448: C9001101002100805
65192: 1100D1CDD5FED121D        65456: D5E511D61B010002C
65200: 00D0CDD5FED1CF3EF        65464: DBBBEDB0D3BBE1D1A
65208: ED5B045BD5CF3F3A5        65472: D5E5CF3EE1D12424F
65216: 035B11D61B8257215        65480: 1C7BFE0B20E21E014
65224: 00D1CDD5FED1CF3E1        65488: 147AFE0420DAC9474
65232: 21A9FF34C9010001E        65496: 3A815CFE5B280A785
65240: DBBBEDB0D3BBC9061        65504: FE61D8FE7BD0CBAF9
65248: FF21007F04247EB72        65512: C978C9F578CDD7FF3
65256: 20FA7832065BC93AF        65520: 47F1C910101010005
```

The first feature is that it compacts the directory, this just makes the directory numbers go up in sequence, ie 1,2,3,4,etc. instead of maybe 1,4,9,23,etc.. This is useful if you have a lot of files and want to know how many files you do have on the disc. Another feature is that it can either treat capitals as the same or different from lower case, ie TEST=test or TEST<>test. The default is TEST<>test, but if you do want them treated as the same all you have to do is POKE 23681,0 or any number except 91.

Using this program will corrupt a large area of memory, 32768-53759. The program also uses 23296-23310. To save the routine off to disc use SAVE d*"CatSortCd" CODE 65000,523.

Here is a BASIC program to be used with the Machine-Code. You do not need it, but it does help.

```
   10 REM BASIC program for Cat-Sort Routine.
   20 REM Written By Hugh J. McLenaghan
   30 REM On 2nd Nov 1988.
   40 REM For DISCiPLE ONLY.
   50 REM
   60 CLEAR 64999
   70 LOAD d*"CatSortCd" CODE
   80 CLS
   90 PRINT "Insert Disc, Then Press ENTER."
  100 IF INKEY$<>CHR$13 THEN GOTO 100
  110 CLS
  120 CAT 1
  130 PRINT '"Sort This Disc? (Y/N)"
  140 GOSUB 9e3
  150 IF a$="N" THEN GOTO 80
  160 PRINT ''"Do you wish Upper Case to be    treated as Lower Case
? (Y/N)"
  170 GOSUB 9e3
  180 LET case=(91 AND a$="N")+(0 AND a$="Y"): POKE 23681,case
  190 CLS
  200 PRINT AT 10,10; FLASH 1;"PLEASE WAIT"
  210 RANDOMIZE USR 65e3
  220 CLS
  230 PRINT "Do you wish to Sort another     Disc? (Y/N)"
  240 GOSUB 9e3
  250 IF a$="Y" THEN GOTO 80
  260 PRINT USR 0
 9000 POKE 23658,8: LET a$=INKEY$: IF a$<>"Y" AND a$<>"N" THEN GOTO
9e3
 9010 RETURN
```

You now save it as SAVE d*"Cat-Sort" LINE 10. All you have to do now if you wish to sort a disc is insert the disc you have saved these programs on and type LOAD d*"Cat-Sort" and the program does the rest for you."

I hope you have fun with this utility. If you have any comments or suggestions, then do not hesitate in contacting me c/o FORMAT. I would value your POKEs, Alterations and Hacks, I will publish the best but I need you to send them in first.

All letters will be answered when I have time to do so as I am now at University.

Thank you for reading and see you next month.

ADVENTURE CORNER

By: Paul Rigby.

This month I would like to introduce the world of maps and mapping to you. I consider the importance of mapping with in adventures high enough to warrant an article all of its own. Hence the reason for mentioning it now rather than during the two previous articles for beginners.

There are experienced adventurers out there who never, or seldom, make maps of the adventures they play. They do actually finish adventures this way but I will bet that they incure frustration, that could easily be avoided, solely as a result of not building a map as they play. So what are maps? How do you build them? Are they really necessary?

A map is a pictorial representation of all of the locations that you have visited. It can show many things but it will, at the very least, give a representative view of all of the positions of the locations relative to each other plus the directional paths that lead to and from them. A map that shows the latter will give the player an instant overview of the area that has been searched. It also allows the rapid movement, from one side of the map to the other, of the character without wasting time being lost in a myrad of locations. Where you reach the point of - " Now is it west to the large oak tree or east? ". Lets face it, adventures contain enough puzzles to tease and frustrate without you creating a whole new set of your own!

To begin with, let us assume that each location takes the form of a simple rectangular box. It does not matter if the location represents a lone point in a large field of in the middle of an ocean think of the place you are in as a box. Why? Well, because that is exactly how it looks on the majority of the original maps in the possession of the adventure author in question. So, if it takes three moves to cross a large tract of empty space then think of those three moves as three rectangular boxes. The boxes serve as a vehicle to contain valuable information about the location. To illustrate the method of map making imagine a room in a house, where the adventure partly takes place. The loacation description describes the room as; " A cold, bare room with no furniture and broken windows. The mice, you notice, have played merry hell with the skirting boards. Exits lay to the east and west. " This is our first location (see Fig 1).

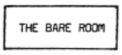


Fig 1.

In this example I have called the room the " Bare Room ". This is to avoid any confusion between any other room which may be present in the adventure. Of course, if you re-read the above location description you can see that Fig 1 could easily have read " The Cold Room ", or whatever takes your fancy. Notice the two sets of dotted lines protruding out of the box. They represent the two possible exits out of the room. I would recommend adding them now rather than later. Some players add the directional indicators as they are about to go in that particular direction. However, this means that it is possible to forget that an exit exists. For example, if, from the " Bare Room ", you travelled west into another room and then west again and then north, etc, etc you would, possibly become embroiled in the puzzles in that area. The exit east of the " Bare Room " could easily be forgotten. Therefore, mark it down straight away and it will serve as a visual reminder. Other visual reminders that could be included with the words " Bare Room " are any hazards that you encoured there, such as a man-eating dog, or any objects that have been found or temporarily dropped. All that is needed is, for example, the letter "H", for hazard, and then the word DOG. While a dropped object may be represented by "DO" and then an abbrieviated name for the object dropped. Similarly, an object found in that location could have the letters "OF" followed by the name of the object.

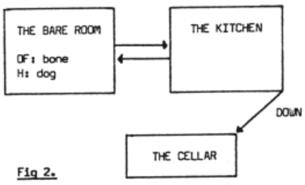
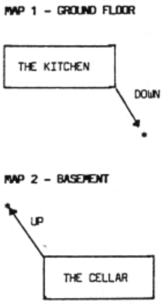


Fig 2.

There are occasions, such as in buildings, when the player may need to go upstairs to a second floor or down to a cellar. In this case the method of indicating exits may vary, according to the player's taste. The indicator may protrude from the corner at a diagonal with an arrow at the end of it and the word up or down marked along it. The first location the player sees on the next floor may be marked on the map (see Fig 2). An alternative may be to place an asterisk, or similar, at the end of the arrow. A completely new map could then be made on a different area of the piece of paper or on a totally seperate piece of paper. The corresponding exit to the original floor should be made on the new map, however, to make sure that you remember where to go. This should have the same symbol as on the original map (see Fig 3).


MAP 1 - GROUND FLOOR

MAP 2 - BASEMENT

Fig 3.

Other suggestions, which are by no means essential, but do help, are the personalisation of different locations and areas. For example, in a marshy area the group of locations within the marsh could have tufts of grass with water symbols drawn between them with a felt-tip pen. Other areas could be similarly decorated to ease identifacation of an area. Finally some locations may be drawn in a characteristic shape. For example, a room may be circular or hexagonal in shape. The corresponding shape on a map would, again, help to identify a location.

If you have any ideas or views on maps, or any other facet of adventuring then please write to me care of Format. So until next month - Happy Adventuring!

NETWORK BATTLE

By: Nev Young.

This is the game of battleships that I knocked up one afternoon to amuse my son. It is played on two Spectrums connected via a network. All the network commands I have used are the Interface 1 format as I have one DISCiPLE and one Interface 1.

To play, the user without the program on disc should enter **LOAD *"n";1** as a direct command. The other should load battleships from the disc. If saved with the command LINE 1 it should auto run. The master Spectrum then copies the program over the net and the game begins. Each player enters their fleet and then their first salvo and when both are complete the shots are transferred over the net. Neither Spectrum has any knowledge of the whereabouts of the other players ships and so cheating is avoided. (My son and I play in different rooms). The tally of hits and firepower is maintained by the computer. As it is only a simple program, however, no check is made that the placing of your fleet is legal. Anyway the rules probably change from place to place so I will leave you to police your own set of rules.

None the less it may be of interest to anybody that has the capability of running a network.

```
  10 DEF FN t(n)=1+1*(n=66)+2*(n=68)+3*(n=67)+4*(n=83)
  20 CLS : PRINT "Waiting to send"
  30 SAVE * "n";1 LINE 50
  40 LET mas=1: GOTO 60
  50 LET mas=0
  60 REM set up my sea
  80 CLS #
  90 PAPER 1: BORDER 1: INK 9: CLS
 100 PRINT " 102030405060708090"
 110 FOR n=1 TO 9: PRINT 'n': NEXT n
 120 FOR n= 8 TO 152 STEP 16: PLOT n,23: DRAW 0,152: NEXT n
 130 FOR n= 167 TO 23  STEP -16: PLOT 0,n: DRAW 152,0: NEXT n
 140 DIM o(9,9)
 150 DATA "miss","bat","des","cru","sub": DIM d$(5,4): FOR n=1 TO 5:
 READ d$(n): NEXT n
 160 FOR n= 1 TO 18: PRINT  OVER 1;AT n,1; PAPER 5;"
  ";: NEXT n
 170 DATA "Enter Battleship ","B",5,"Enter Destroyer ","D",4,"Enter
Cruser ","C",3,"Enter Submarine ","S",2
 180 FOR z=1 TO 4: READ p$,q$,l
 190 FOR n=1 TO l
 200 INPUT (p$);ship
 210 LET q=ship: GOSUB 290: IF NOT q THEN  GOTO 200
 220 IF q THEN  IF o(x,y)<>0 THEN  GOTO 200
 230 LET o(x,y)=CODE q$
 240 LET co=ship: GOSUB 340: PRINT AT r,c; OVER 1;q$;
 250 NEXT n
 260 NEXT z
 270 GOTO 400
 280 STOP
 290 REM validate q = good location
 300 LET x=INT (q/10): LET y=q-10*x
 310 IF x<1 OR x>9 THEN  LET q=0
 320 IF y<1 OR y>9 THEN  LET q=0
 330 RETURN
 340 REM get row & col from co
 350 LET c=INT (co/10)
 360 LET r=co-10*c
 370 LET r=2*r
 380 LET c=2*c
 390 RETURN
 400 REM start game
 410 PRINT AT 0,20; "Your  salvo";AT 10,20;"Enemy salvo";
 420 LET shots =5: LET ba=5: LET de=4: LET cr=3: LET su=2
 430 DATA "Enter 1st shot ","Enter 2nd shot ","Enter 3rd shot ","Ent
er 4th shot ","Enter 5th shot "
 440 DIM t(5): RESTORE 430: FOR n= 1 TO shots
 450 READ p$
 460 INPUT (p$);q: GOSUB 290: IF NOT q THEN  GOTO 460
 470 PRINT AT 2+n,20;q,
 480 LET co=q: GOSUB 340: PRINT AT r-1,c-1; OVER 1; PAPER 2; FLASH 1
;" ";
 490 LET t(n)=q
 500 NEXT n
 510 FOR n=n TO 5: PRINT AT 2+n,20;" ",: NEXT n
 520 PRINT AT 21,0;"Waiting to send": REM send my salvo
 get his salvo
 530 IF mas THEN  SAVE *"n";1 DATA t(): LOAD *"n";1 DATA r()
 540 IF NOT mas THEN  LOAD *"n";1 DATA r(): SAVE *"n";1 DATA t()
 550 PRINT AT 21,0;" ",,
 560 REM mark his shots on my sea
 570 FOR n=1 TO 5
 580 PRINT AT 12+n,20;" ",
 590 IF NOT r(n) THEN  GOTO 720
 600 LET q=r(n): GOSUB 290
 610 LET co=q: GOSUB 340
 620 PRINT  PAPER 4;AT r,c; OVER 1;" ";
 630 PRINT AT 12+n,20;q,
 640 LET r(n)=o(x,y)
 650 PRINT AT 12+n,25;d$(FN t(r(n)))
 660 IF NOT r(n) THEN  GOTO 720
 670 REM BEEN HIT !!!
 680 IF r(n)=66 THEN  LET ba=ba-1: IF ba=0 THEN  LET shots=shots-2
 690 IF r(n)=68 THEN  LET de=de-1: IF de=0 THEN  LET shots=shots-1
 700 IF r(n)=67 THEN  LET cr=cr-1: IF cr=0 THEN  LET shots=shots-1
 710 IF r(n)=83 THEN  LET su=su-1: IF su=0 THEN  LET shots=shots-1
 720 NEXT n
 730 REM tel him wot he's hit
 740 PRINT AT 21,0;"Waiting to send"
 750 IF mas THEN  SAVE * "n";1 DATA r(): LOAD * "n";1 DATA e()
 760 IF NOT mas THEN  LOAD * "n";1 DATA e(): SAVE *"n";1 DATA r()
 770 PRINT AT 21,0;" ",,
 780 REM show wot i've hit
```

```
 790 FOR n= 1 TO 5
 800 LET q=t(n): GOSUB 290
 810 LET co=q: GOSUB 340
 820 PRINT AT r-1,c-1; OVER 1; PAPER 2+NOT NOT e(n);CHR$ e(n) AND e(
n);" " AND NOT e(n);
 830 PRINT AT 2+n,25;d$(FN t(e(n))),
 840 NEXT n
 841 DIM m(1): LET m(1) = shots
 842 IF mas THEN  SAVE *"n";1 DATA m(): LOAD *"n";1 DATA l()
 843 IF NOT mas THEN  LOAD *"n";1 DATA l(): SAVE *"n";1 DATA m()
 850 IF NOT l(1) THEN  GOTO 1000
 860 IF shots THEN  GOTO 440
 870 PRINT AT 20,0;"You have lost"'"press any key to continue": PAUS
E 0: GOTO 2000
1000 IF shots THEN  PRINT AT 20,0;"You have won"'"press any key to c
ontinue": PAUSE 0: GOTO 2000
1010 PRINT AT 20,0;"You have scored a draw"'"press any key to contin
ue": PAUSE 0: GOTO 2000
2000 IF mas THEN RUN
9999 CLS #: LOAD *"n";1
```

- + - + - + - + - + - + -

ANSWERS TO FUN QUIZ ON PAGE 19.

1) b, 2) c, 3) d, 4) c. 5) d. 6) c. 7) a, 8) a. 9) d. 10) d.

If you disagree with me on any answers - HARD LUCK - I'm the editor. By the way, modesty forbids me to give 'c' as the correct answer to question 10.

MIDI

MIDI FOR BEGINNERS PART 4 - BY RAY ELDER.

A useful utility for MIDI users.

This program was born out of the need to change instrument sounds quickly so that I could play several combinations of patches on different instruments without having to stop and fiddle with lots of little buttons. It is intended for use by those of you who have two or more MIDI instruments perhaps linked together with a sequencer or computer.

By introducing a Spectrum with a MIDI interface into the system using a MIDI combine box, a box that will allow two sets of signals to be combined into one out socket and then passed onto the setup, we can overcome an awkward problem for those who like to change their sound combinations quickly, perhaps in the middle of a song as you are playing. The problem is that if two or more instruments are linked directly, then when you change the sound on the master keyboard the patches on the other keyboards are changed to the same numbered patch.

For example, if I have the master keyboard set to play an organ patch and the sound module set to play a string patch this is fine until I decide to change the master voice to perhaps a brass sound - the sound module also changes to whatever patch number the master is changed to, and this is most likely to be completely the wrong sound for that section. Manually it means pressing a key on the master instrument and then changing the other modules etc. This can often take several minutes and completely ruin the chorus of 'Tie a Yellow Ribbon'!.

As I mentioned you will need an combiner box but these are available reasonably cheaply from a variety of suppliers.

The program is set up to work on all the main MIDI interfaces that are available and provides three operational functions or modes. It is a fully menu driven program leading you into each mode, and from each of these modes you have the option to return to the main menu. The operating modes are as follows:-

1. SINGLE PATCH, this is really included because it was easy to add! Basically you enter the number of the patch you want to send and press the ENTER key. The patch change is sent instantly and the information is then lost. It is not stored in memory. This is probably a more awkward way of changing a patch than by using the keyboard selector....

2. PRESET PATCHES, up to ten individual patch changes are stored in memory and then are sent in a block when a key is pressed. This is useful for sending multiple patch changes on several MIDI channels. When the program first starts all the ten patches are set to 0 and

you must set them up as you require.

3. SEQUENCE PATCHES, again up to ten patches are sent, but this mode they are sent one at a time each time a key is pressed. Once the end of a sequence is reached then the program cycles back to the start and begins sending the sequence all over again. You need not enter all ten patch changes; for instance, setting two patches, say 10 & 32, will change from one to the other and back again on subsequent presses of a key

There are two keys which must not be pressed when sending patches, P & R, these have special functions. 'P' key allows you to reset all your patch sequence data and 'R' will return you to the main menu.

There are two entry modes in the program, a single key press from the menus and by entering numerical data followed by pressing the ENTER key. Whenever you are at a point where the ENTER key needs to be pressed you can type C and press ENTER and you will be offered the option of altering the channel output number.

```
  20 CLS : PRINT AT 1,10;"PATCH SEND.";AT 4,3;"1. MICON (XRI system
s)";AT 6,3;"2. EMR system";AT 8,3;"3. SEIL, JMS";AT 10,3;"4.UPSTREA
M, E+MM"
  30 PRINT AT 14,6;"Press a key 1 to 4"
  40 LET g$=INKEY$: IF g$<"1" OR g$>"4" THEN GOTO 40
  50 LET stat=63+(128 AND g$>"1")-(32 AND g$>"2"): LET trans=191+(6
4 AND g$>"2")
  60 GOSUB 9000: GOTO 500
 100 INPUT (m$); LINE p$
 110 IF (p$="c" OR p$="C") AND NOT flag THEN GOTO 200
 115 IF (p$="r" OR p$="R") AND NOT ret THEN LET ret=1: LET p=0: RET
URN
 120 IF p$="" OR LEN p$>2 THEN GOTO 100
 125 IF p$="e" OR p$="E" THEN LET end=1: LET p=1: RETURN
 130 FOR i=1 TO LEN p$: IF p$(i)<"0" OR p$(i)>"9" THEN LET p$="XX"
 140 NEXT i: IF p$="XX" THEN GOTO 100
 150 IF VAL p$<0 OR VAL p$>max THEN GOTO 100
 160 LET p=VAL p$: RETURN
 200 CLS : PRINT AT 4,8;"Change channel number.";AT 6,9;"Present ch
annel = ";chan
 210 LET flag=1: LET m$="Enter channel number 1 - 16 ": LET max=16
 220 GOSUB inp: IF p<1 THEN GOTO 220
 230 LET chan=p: LET ctrl=191+chan
 240 RETURN
 500 CLS : PRINT AT 1,8;"MIDI PATCH SEND";AT 6,6;"1. Single input";
AT 8,6;"2. Preset Patches";AT 10,6;"3. Sequence of patches"
 510 PRINT AT 16,6;"Press a key 1,2 or 3"
 520 LET g$=INKEY$: IF g$<"1" OR g$>"3" THEN GOTO 520
 530 GOSUB 1000*VAL g$
 540 GOTO 500
1000 CLS : PRINT AT 6,10;"Patch change ";AT 8,10;"Last patch=";pat;
AT 10,10;"Channel No.=";chan;AT 16,0;"Enter C to change channel num
beror    R to return to the menu."
1010 LET ret=0: LET flag=0: LET m$="ENTER PATCH NUMBER 1 TO 99 ": L
ET MAX=99
1015 IF ret THEN RETURN
1020 GOSUB inp: IF flag THEN LET flag=0: GOTO 1000
```

```
1025 IF ret THEN RETURN
1030 LET pat=p
1035 OUT trans,ctrl: OUT trans,p
1040 GOTO 1000
2000 CLS : PRINT AT 0,8;"PATCH CHANGE PRESETS";AT 2,11;"CHANNEL No.
";chan;AT 5,0;"KEY No.  ";AT 9,0;"PATCH No."
2010 FOR i=1 TO 10: PRINT AT 6,i*3-1;i-1;AT 8,i*3-1;p(i): NEXT i
2020 PRINT AT 12,0;"Press Key";AT 14,5;"0 to 9 to send patch";AT 16
,5;"P to redefine patch preset";AT 18,5;"R to return to the menu"
2025 LET flag=0
2030 LET g$=INKEY$: IF g$<>"p" AND g$<>"P" AND g$<>"r" AND g$<>"R"
AND (g$<"0" OR g$>"9") THEN GOTO 2030
2040 IF g$="P" OR g$="p" THEN GOTO 2500
2050 IF g$="R" OR g$="r" THEN RETURN
2060 OUT trans,ctrl: OUT trans,p(VAL g$+1)
2070 GOTO 2030
2500 LET m$="Enter preset number 0 to 9 ": LET max=9
2510 GOSUB inp: IF flag THEN GOTO 2000
2520 LET x=p+1: LET m$="Enter value of preset "+STR$ p+" (1 to 99)
": LET max=99
2530 GOSUB inp: IF p<1 THEN GOTO 2530
2535 IF flag THEN GOTO 2000
2540 LET p(x)=p: PRINT AT 8,x*3-1;p
2550 GOTO 2030
3000 CLS : LET x=1: LET end=0: DIM q(10): LET flag=0
3010 LET max=99: LET m$="Enter patch no."+STR$ x+" (E to End) "
3020 GOSUB inp: IF end AND x=1 THEN LET end=0: GOTO 3020
3025 IF flag THEN LET flag=0: CLS : FOR i=1 TO x-1: PRINT AT i*2,8;
"PATCH ";i;"=";q(i): NEXT i:GOTO 3010
3030 IF NOT end THEN PRINT AT x*2,8;"PATCH ";x;"=";p: LET q(x)=p: L
ET x=x+1: IF x<11 THEN GOTO 3010
3035 LET x=x-1
3040 CLS : PRINT AT 0,10;"PATCH SEQUENCES": FOR i=1 TO x: PRINT AT
4,i*3-1;i;AT 6,i*3-1;q(i): NEXT i
3050 PRINT AT 8,0;"Press";AT 10,6;"Any key for next patch";AT 12,6;
"R to return to menu";AT 14,6;"P to set up a new sequence"
3060 FOR i=1 TO x
3065 PRINT AT 7,i*3-1; FLASH 1;"^"; FLASH 0: LET y=i-1+(x AND i=1):
 PRINT AT 7,y*3-1;" "
3067 IF INKEY$<>"" THEN GOTO 3067
3070 LET g$=INKEY$: IF g$="" THEN GOTO 3070
3080 IF g$="R" OR g$="r" THEN RETURN
3090 IF g$="P" OR g$="p" THEN GOTO 3000
3100 OUT trans,ctrl: OUT trans,q(i)
3120 NEXT i
3130 GOTO 3060
9000 DIM p(10): LET chan=1: LET ctrl=192: LET pnt=1: LET pat=1
9010 OUT stat,3: OUT stat,86: OUT trans,176: OUT trans,124: OUT tra
ns,176: OUT trans,127
9020 LET ret=1: LET flag=0: LET inp=100: RETURN
```

Well that's all for this month, for the next two months I will be looking at practical ways of using MIDI sequencers and editor programs and presenting some of the ideas and tips that have been sent to me. If you have any suggestions or ideas or problems or comments or in fact anything related to MIDI music please drop me a line at 1 Periton Court, Parkhouse Rd. Minehead, Somerset. TA24 8AE. Please write to me I look forward to hearing from you.

ADDING COMMANDS TO BASIC

By: Nev Young.

This month I will attempt to show how new functions can be added to the spectrum. Although the first part of this article is intended for DISCiPLE or PLUS D owners the last section will interest most Spectrum owners.

Perhaps I should first say what the difference is between a COMMAND and a FUNCTION as the terms are often mixed up. A COMMAND is the word that starts a BASIC statement and the whole statement is often named after it eg PRINT, SAVE. A function is quite another thing. Functions may or may not take parameters but they will always return something. Functions can be nested (ie one inside the other) and can get very complex. The main feature of a function is that it can always be resolved into either a string or a number. eg VAL (STR$ ((SIN x)^2+(COS x)^2))=1 where VAL, STR$, SIN and COS are functions. The whole thing being an expression, and 1 is the value of the expression.

As well as the problems we have when adding a new command we have a few new problems to contend with if we want to add new functions. These are caused by the DOS code that has tidied thing up for us. The calculator stack has been cleared so any partial results of an expression are lost and the workspace is cleared. (Thank you Bruce). This can be very nasty as some functions have parts copied into the workspace and are evaluated there, VAL "2+3" for example has the "2+3" copied to workspace and then it is syntaxed and then evaluated. This means we are very limited as to what we can do with our new functions. The example I will give is PEEK @ (It is the opposite of POKE @).

By way of an example of the problems mentioned above if PEEK @ 1 gives 208 then what does 7+PEEK @ 1 give : 1 of course ?? This is because the 7 has been cleared from the calculator stack but the + has not been removed from the opcode list. (PEEK @ 1)+7 will work OK. You will find quite a number of places where new functions will not work or cause strange things to happen. For safety always use them as a single statement eg LET x = PEEK @ 1: PRINT x;TAB 12; Rather than PRINT PEEK @ 1;TAB 12;. The most annoying thing is when you can enter a line of BASIC but get error 12 (Nonsense in BASIC) when you run it.

This example is written in three parts. The first (lines 330-460) is the auto run code that installs the new function. The second (lines 540-1060) test the syntax of the function and resets the machine stack and error to just before the error happened. The third (lines 1070-1270) does the work. The last few lines return into the expression evaluation routine.

Type the following routine into your assembler and assemble it to a convenient address, I used 65000.

```
0010  ; ADD A FUNCTION TO BASIC
0020  ;
0030  ; PEEK @n
0040  ;
0050  ;
0060          ORG  65000
0070  ;
0080  ; MAIN ROM ADDRESSES
0090  ;
0100  CHADD   EQU  23645
0110  X_PTR   EQU  23647
0120  ;
0130  ; DISCiPLE ADDRESSES
0140  ;
0150  CMR     EQU  16
0160  RTHL    EQU  79
0170  RTBC    EQU  70
0180  GTNC    EQU  40
0190  CFSO    EQU  48
0200  RS18    EQU  44
0210  ;
0220  ; DISCiPLE only ADDRESSES
0230  ;
0240  ONERR   EQU  678
0250  RESP    EQU  187
0260  OFFSET EQU  664
0265  CADR    EQU  1735
0270  ;
0280  ; PLUS D only ADDRESSES
0290  ;
0300  ONERR   EQU  8206
0310  RESP    EQU  231
0320  OFFSET EQU  8192
0325  CADR    EQU  8463
0330  ;
0340  ; Begin by setting ONERR
0345  ; & altering RST CMR
0350  ;
0360  ; page in the DISCiPLE
0370          RST  8
0380          DEFB 71
0390  ; load ONERR
0400          LD   HL,START
0410          LD   (ONERR),HL
0420  ; alter RST to CALL
0430          LD   HL,CADR
0440          LD   (HL),205
0450  ; page out the DISCiPLE
0460          OUT  (RESP),A
0470          RET
0480  ;
0490  ; Jump to here if DISCiPLE
0500  ; syntax fails
0510  ;
0520  ; reset CHADD to just
0530  ; before syntax failed
0540
0550  START  LD   HL,(X-PTR)
0560          DEC  HL
0570          LD   (CHADD),HL
0580  ;
0590  ; see if last token='PEEK'
0600          CALL RS18
0610          CP   190      ; PEEK
0620  ; pre-load error address
0630          LD   BC,88
0640          JP   NZ,RTBC
0650  ;
0660  ; test if a '@'
0670  ;
0680          RST  GTNC
0690          CP   '@'
0700  ; pre-load error address
0710          LD   BC,88
0720          JP   NZ,RTBC
0730  ;
0740  ; OK it was a PEEK @
0750  ; clear machine stack by
0760  ; searching for PEEK op code
0770  ; see table #1 for ops
0780  ;
0790          LD   DE,4331 ; = PEEK
0800  L1      POP  HL
0810          XOR  A
0820          SBC  HL,DE
0830          JR   NZ,L1
0840  ;
0850  ; We have removed the last
0860  ; function op code from the
0870  ; function stack.
0880  ;
0890  ; Now clear the error
0900          LD   A,255
0910          LD   (IY+0),A
0920  ;
0930  ; We now need to put a last
0940  ; value on the calc stack
0950  ;
0960  ; move on to the next char
0970          rst  gtnc
0980  ;
0990  ; read next number onto
1000  ; calculator stack
1010          RST  CMR
1020          DEFW 7298    ; (1C82H)
1030  ;
1040  ; test for syntax
1050          RST  CFSO
1060          JP   Z,FIN
1070  ;
1080  ; get last value in BC
1090          RST  CMR
1100          DEFW 7833    ; (1E99H)
1110  ;
1120  ; fiddle the base value
1130          LD   HL,OFFSET
```

```
1140          ADD  HL,BC
1150  ;
1160  ; do the peek
1170          LD   A,(HL)
1180  ;
1190  ; put it on the calc stack
1200          RST  CMR
1210          DEFW 11560
1220  ;
1230  ; We now have the value we
1240  ; require on the calc stack
1250  ; so jump back into the
1260  ; middle of SCANNING routine
1270  ;
1280  ; load A with (chadd)
1290  FIN     CALL RS18
1300  ;
1310  ; load B with final calc op
1320          LD   B,0
1330  ;
1340  ; and return
1350          LD   HL,10036
1360          JP   RTHL
1370  ;
```

Now a few words (pages) on how the expression evaluation is done. It always starts with a call to SCANNING at 24FBH and what is more it can call itself (A very clever piece of recursive programming). Two stacks are involved in the process, one - the op code stack - is held on the machine stack and holds the operation codes (+ - VAL COS etc). The other is the calculator stack that holds a pile (Is that the correct term for items on a stack?) of values representing numbers or string parameters. The top most of which is the LAST value.

When SCANNING is first entered the opcode stack and the calc stack are both empty. An opcode of 00 is pushed onto the op code stack to mark the end of the stack. The BASIC line is then scanned. Each operation that is found is added to the op code stack. Eventually an operand will be found, this is either a number, a string or a variable holding a number or a string. When this happens the operand is placed onto the calculator stack.

	OP CODE STACK	CALC STACK
enter	empty	empty
mark end	00	empty
find 2	00	2
find +	06CF 00	2
find 3	06CF 00	3 2
find *	08C4 06CF 00	3 2
find 4	08C4 06CF 00	4 3 2
find end	00 08C4 06CF 00	4 3 2
call FP	00 06CF 00	12 2
call FP	00 00	14
finish	empty	14

EXAMPLE 1 - 2+3*4

What happens next depends on what follows the operand. If it is the end of the statement then the expression is evaluated by applying the operations from the op code stack one after the other to the last value on the calc stack.This is done by repeatedly calling the floating point calculator. When the op code is 00 then the last value on the calc stack is the required one. However if the end of the statement is not reached and we find instead another operator a decision has to be made. Either add it to the op code list, or recurse and evaluate the next part of the expression first.

In a simple example what is 2+3*4 24 or 14 ? The arithmetic priority rules state the answer must be 14. If we step through that example to show how it is done for the two expressions 2+3*4 (Fig 1) and 4*3+2 (Fig 2).

Easy isn't it! But, I hear you cry, where do these magic op codes come from and what do they mean? The op code is built of two parts the first byte is the priority, High priority things get done first.

(See chapter 30 of your spectrum manual "The BASIC", there is a table of operations and priorities). The lower 6 bits of the second byte is used to identify the operation 0 through 65 and are used by the Floating point calc to find the address of the routine that does the work for that operation. They are generated in two ways by looking up in a table for binary operators (+ - * / ^ = > < <= >= <> OR AND) and by fiddling about (CODE [keyword] - 0AFH + 0DCH) with the value of the keyword (VAL$ through NOT). The last two bits of the lower byte show the type (string or number) of the last value required (bit 6) and the type of last value returned (bit 7) eg VAL needs a string and returns a number. If the bit is reset it signifies a string. As a final note there are 3 functions that are not handled by the FP calc. These are SCREEN$, ATTR, and POINT. These are all performed by their own special subroutines that are called from SCANNING.

	OP CODE STACK	CALC STACK
enter	empty	empty
mark end	00	empty
find 4	00	4
find *	08C4 00	4
find 3	08C4 00	3 4
find +	06CF 08C4 00	3 4
06 less than 08 so call FP NOW		
call FP	06CF 00	12
find 2	06CF 00	2 12
find end	00 06CF 00	2 12
call FP	00 00	14
finish	empty	14

EXAMPLE 2 - 4*3+2

Well thats all I've got to say for the moment on expanding basic, its now up to you. In the early days of Interface 1 there were lots of articles on new routines to extend basic. Most should be convertible to the DISCiPLE & PLUS D and when you've done the conversion why not send it in for publication in FORMAT. It would only be right to credit the original author in your article and to mention the magazine you got it from. I think I can speak for everyone when I say we look forward to seeing the results.

FIXIT REVIEW

MGT's FIXIT - FIXES THE SPECTRUM +2a.

By: John Wase.

The trouble with the +2a is that the old ROM-CS line on output port pin 24 (lower) is no longer there: there are now two ROM switch lines in previously unused locations, (pins 4 upper and 14 lower). Equally thoughtfully, Alan Sugar's mates have omitted the 9 volt output (pin 4 lower). The DISCiPLE and PLUS D won't work. MGT gallop in to the rescue with their "Fixit".

This gizmo is a well-made little connector which joins pins 4 and 14 to 24 through two diodes (this prevents any cock-up pushing current back into the chips), and the 12 volt line on pin 21 (upper) to pin 24. This compromise is not perfect, but enables many things to work.

The +D/Disciple both work splendidly, except that the DOS fix letting you work with the +2 in 128k full screen will not fix the 2a: beware. Interface 1 runs microdrives a treat. Whatever I tried with the +3 resulted in a reset: MGT are working on this. The Discovery was more of a problem. For those who don't know, a Discovery disc system contains a power supply which also powers the Spectrum back through the edge connector. This won't work this way round with the "Fixit", but if you plug in the power pack with its great big plug and switch on everything at once to prevent a crash, it all works. Beware, though, if (as I did), you subsequently leave the power unit switched off, but the plug still in: 12 volts will find its way round the power pack and back through the 9 volt line, eventually killing your Spectrum 2a dead. Embarassing, as I had borrowed it from the main Dixons in Birmingham city centre. They were very nice about it too (in fact, in spite of all the adverse press comment, I've always found that branch very helpful). And that's why I could test nothing else, although I suspect most modems would now be O.K.

If you are going to run a device like the Discovery, with its own power supply back into the Spectrum, let MGT know, and they will supply one with the 12 volt track cut. It then won't be of general use, but can't ruin your Spectrum. At the price (£7.95), you could afford the luxury of a dedicated one. It's neat, cheap, and does far more than MGT at first intended.

* * COMING SOON * *
The YOUR REVIEW Section
see next month for details