FROM OUT OF 'THE ASHES' RISES

>>>> ZXir QLive Alive! <<<<

The
Timex/Sinclair NorthAmerican User Groups
Newsletter

MEMORY MAP

T/SNUG CHAIRMEN

Here is the list of 1992 T/SNUG Chairmen and how to contact them. We wish to support the following SIGS: ZX80/ZX81, Z88, SPECTRUM/TS2068/TC2068 and QL. If you have questions about any of these fine machines contact the Chairman.

POSITION	NAME	PHONE	PRIMARY FUNCTION
Chairman	Don Lambert	219-925-1372	Chief Motivator
Vice-Chairman	Dave Bennett	717-774-7531	CATS/Z88
Vice-Chairman	D.G. Smith	814-535-6998	TAPE & JLO Library
Vice-Chairman	Ed Snow	407-380-5124	ZX81 TAPE Library
Vice-Chairman	Rod Gowen	503-655-7484	CCATS
Vice-Chairman	Rod Humphreys	604-931-5509	VSUG/TS2068
Vice-Chairman	Bob Swoger	708-837-7957	Newsletter/BBS Sysop
Treasurer	Abed Kahale	708-885-4337	CATUG/Cash Tracker

T/SNUG Information

ZXir QLive Alive! is the newsletter of T/SNUG, the TIMEX/SINCLAIR NorthAmerican User Groups, providing news and software support to the T/S community in at least four newsletters per year.

It is our goal to build a Public Domain software library and develop a list of available software for all T/S machines showing the source.

Vendors have free space in this newsletter which they receive free of charge so they may see we are still out here. If you feel T/SNUG should perform other tasks, let us know your feelings.

T/SNUG wishes to have one chairman from every T/S user group who will take charge of sending us their groups newsletter and other correspondence for inclusion in this newsletter.

We encourage your group to copy this newsletter and distribute it at regular meetings to all of your members. If you can't copy this newsletter, perhaps we can provide a disk with the articles on it for use in your newsletter.

Articles appearing in this newsletter can be obtained by downloading this newsletter from our BBS.

Have you solved a problem in one of your softwares or hardwares? Please share it with the rest of us.

You can keep T/SNUG alive for an annual contribution of $10.00 made payable to Abed Kahale. Send questions, articles or check to:

ABED KAHALE
ZXir QLive Alive! Newsletter
335 W NEWPORT RD
HOFFMAN ESTATES IL 60195-3106

Tele: H708-885-4337

ZXir QLive Alive! Article Contributions

If you would like to contribute an article to the newsletter, upload a file to our BBS called TSNUG.ART . If you have an AD for the newsletter UPLOAD a file called TSNUG.ADS. If you have NEWS to POST about your group. UPLOAD a file called TSNUG.NWS.

If you need help contact the SYSOP by E-MAIL on the T/SNUG BBS, mail or by phone:

BOB SWOGER
613 PARKSIDE CIRCLE
STREAMWOOD IL 60107-1647

It is preferred you call:
H708/837-7957 or W708/576-8068

If you can only contribute hard copy, tape or disk format, send your inputs to:

DON LAMBERT
ZXir QLive Alive! Newsletter
1301 KIBLINGER PL
AUBURN IN 46706

Tele. 219/925-1372, it is not necessary to call when submitting material.

CONTRIBUTORS TO THIS ISSUE

    Marvin Johnson
    Abed Kahale
    Don Lambert
    Joseph P. Rampolla
    Bob Swoger, K9WVY

FROM THE CHAIRMANS DISK

I wish to thank Abed Kahale for building up my LarKen RAMDISK board. It works just fine although I have only a few programs on it. I have found only one program that will not work on RAMDISK, nor will it work on a regular disk unless it is LOADed from the original menu program. That is COPY12.B1 that is on TTSUC's disk #1. I thought that with it on RAMDISK I would be able to get it into operation without having to find it in the LarKen disk file box. I suspect that it is

because of a PRINT #4 in the menu that it will not work. But not being a programmer I am lost with anything that is not directly straight programming. Throw in a PRINT #4 and some VAL and I am lost.

I received LogiCall V5.1 from Abed Kahale on my RAMDISK. But there is a problem. While it works on the RAMDISK and the disks that I have generated it on it will not create itself onto another disk as an AUTOSTART program. I can SAVE LogiCall but not generate AUTOSTART. I press A for AUTOSTART and get ERROR C nonsense in BASIC, 430:1. Somewhere, somehow, it has gotten corrupted. My RAMDISK does not have a write protect switch so it could have been overwritten somehow. What I will have to do is use an earlier version of LogiCall to SAVE to disk and then SAVE just the LogiCall V5.1 portion to get the latest version on disk.

I just now tried to get a working LogiCall, that is, one that will self SAVE, by working with V4.9 and then trying to SAVE just the LogiCall V5.1 and while the program works I still cannot use that to SAVE to disk the self LOADing properties since it halts after I press the A key. So the problem is not in the AUTOSTART, maybe, but could be in V5.1. Since the only thing that I use and neither Bob or Abed don't is the LarKen/Oliger marriage that could be the problem. Will version V5.0 work? And I don't have version V5.0. Sorry about that Bob but I can't use the latest version if it will not work. [*Don fell victim to a copy of LogiCall modified for Abed's personal use with Dohany's EPROM! LogiCall has only been released as version V5.0! See editor's report. Ed.*]

In reply to "FROM THE EDITORS DESK" comments in the last issue about the problem of me supplying Editor Bob with DOUBLE SIDED disks when he has only SINGLE SIDED drives – that was an error on my part. I

3

assumed that since he had been sent a double sided disk drive that he had installed it. [*I was? By whom? Ed.*] BUT! Bob has informed me that is not the case. I will only send Bob SINGLE SIDED disks NOW THAT I KNOW THIS. And since Bob uses SSDD disks and his drives require head stepping rate not faster than 30 ms and he wants LogiCall on every disk I will set up my master disk with that on it and will then copy the disk using D.U.S. copy program "COPYII.B1".

For those that have a LarKen TS2068 disk interface and have never used COPYII.B1 it is a program that copies and FORMATs the disk at the same time. If you use COPYII.B1 to copy from a SSDD disk the result is still a SSDD disk, and if you copy a DSDD disk to a DSQD drive you end up with just the same data on the QD disk and the same number of tracks. In other words, you will be using only half the disk. The last half of the disk can't be accessed since the directory only contains the last track that was on the DSDD disk.

In reference to Bob's comment about not using LogiCall to access another drive to get an AUTOSTART to work on a drive other that drive 0, at that time, spring of 1989, I was new to disk drives and also I did not know about the existence LogiCall or Bob! It was later in the year at SMUGFest in Milwaukee that I first got LogiCall and that first version required you to type in the file name to LOAD a program.

I agree with Bob that what I need is a "quickie card". I did create one for a nice person so that there would be no problem of using the TTSUC disk MSDOS to MSCRIPT and I also sent a program to convert MSCRIPT to TASWORD II. The entire "quickie card" ended up being a page and a half of text. I included what to do as well as what to expect on the screen. Bob, how about a couple of "quickie cards" for me and or other readers? I

believe that others could benefit. so please publish it here, the ones I need are on MaxCom and LOADER V.

Don Lambert, Chairman
TIMEX/Sinclair NorthAmerican
User Groups

TREA$URY NOTE$

As of March 30, 1993, we have a balance of $486.39 from 7 groups and 39 individuals.

Abed Kahale, Treasurer
TIMEX/Sinclair NorthAmerican
User Groups

FROM THE EDITORS DESK

I contacted Chairman Don when he told me he had a copy of LogiCall V5.1. This interested me since I, the developer, only had V5.0! I found Treasurer Abed had modified a copy of V5.0 for his own personal use, called it V5.1, and sent it to Don. Abed had changed the program calls to call in his personal software names and so Don found NOTHING would work!

I also received calls from RMG customers that some LogiCall programs didn't work. Upon checking on this I found that a bad version of VU-CALC, V1.2, is being shipped by RMG and also that some customers don't know how to convert FORMAT.B_ to run with their systems. PLEASE BE PATIENT with me and RMG, this shall soon be fixed for future RMG customers. Better to wait for good product than put up with bugs. Thanks!

Bob Swoger, Editor
TIMEX/Sinclair NorthAmerican
User Groups

LIBRARY

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! ! MAILING ADDRESSES ! !
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
It has come to my attention that there are those out there that do not want to use the telephone to find out about availability of the software

and/or hardcopy libraries. So for those I am including the addresses of the Vice-Chairmen if not given else where in ZXir QLive Alive!. Please! When writing enclose a LSASE:

Dave Bennett (Z88)
329 WALTON ST Rear
Lemoyne PA 17043

D G Smith (2068 TAPE/JLO)
R 415 STONE ST
Johnstown PA 15906

Ed Snow (ZX81 TAPE)
2136 CHURCHILL DOWNS CIR
Orlando FL 32825

Rod Gowan (CCATS)
1419 1/2 7th ST
Oregon City OR 97045

Rod Humphreys (VSUG/2068)
10984 Collins Place
Delta B.C. V4C 7E6 CANADA

INPUT/OUTPUT

Marvin Johnson writes: Hi there, I believe you are part of a Timex Sinclair user group? If so, I would appreciate being added to your EMail list for ZX81 matters if possible. I am actually in Paris France despite the EMail address, routed through Cray Inc in the States. mpj@potiron.cray.com Two of us in Cray France have a ZX81 but we haven't figured out a way of hooking it up to the Cray yet !!!

Nobody's got a Spectrum here though a couple of friends in the UK have one. One even has a QL too. :-) Regards,

Mr. Marvin Johnson
11 rue JACQUIER
75014 Paris
FRANCE

Joe Rampolla writes: I received a copy of the newsletter and was happy to see my article! You did a great job of transcribing my stuff, but the editor mistook my sample of writing text for a letter to the editor. He also cut out a few things to make a couple of strange sentences, but I am not

4

complaining. The important thing is that the correct ideas were communicated.

I received a letter from the owner of the Byte-Back company in thanks for giving new life to the modem software, and his product line. He sent the text of a version of ZCOMM, but it is different (I think) from mine, and too technical for me. But I have been making some sense of it. Perhaps I will have some new ideas in the future.

I hope you and your family are well. I am still trying to leave Baltimore city, but the real estate here has been greatly devalued because of the terrible crime problem.

Thanks for your help and all the kind work you do for people like me. Sincerely,

JOSEPH P. RAMPOLLA
2638 E. Monument Street
Baltimore MD 21205

NEWS ITEMS

ComputerFest 1993 sponsored by the Dayton Microcomputer Association, Inc. will be held on Saturday August 28 from 10 AM to 6 PM and Sunday August 29th from 10 AM to 4 PM at the HARA Conference & Exhibition Center, 1001 Shiloh Springs Rd, Dayton Ohio. Contact Chairman Don if you are going. Tickets are $5 for both days.

The ISTUG picnic will be held on June 26th 1993 at the cabin of Frank and Carol Davis. Contact Frank Davis, 513 East Main ST, Peru IN 46970 or phone him at 317-473-8031 to let him know you're coming.

N O T E   T O
M E M B E R S
If you have a question, an article or a complaint send a note or a Post Card to :-)

ABED KAHALE
335 W NEWPORT RD
HOFFMAN ESTATE IL 60195-3106
Tele: H708-885-4337

Please remember that your subscription has to be renewed every new year.......

ITEMS AVAILABLE FROM T/SNUG

It has come to our attention that some LarKen Users are using something less than Version 3 firmware. T/SNUG will supply updated EPROMs, SYSTEM DISKS, and MANUALS.

If you have a mismatch between you LarKen DOS EPROM and your Western Digital Controller chip, we will send you the correct one for free on behalf of our friends Rod Gowen of RMG and Larry Kenny of LarKen. You should be using L3 EPROMs with WD1770 controller chips or L3F EPROMs with WD1772 controller chips. Check it out! Call in requests to Bob Swoger at W708-576-8068 H708-837-7957

SPECIAL DEALS AND BUYS

NAP_Ware (Nazir A. Pashtoon's new endeavor) announces the availability of all Timex or QL PAL (Programmable Array Logic) chips. If interested, call him evenings at 708-439-1679.

If you are a LarKen LK-DOS owner and would like a SPECTRUM V2 kit for your system, we will supply an EPROM, socket and 74HCT32 for $12 which includes shipping and handling. The install instructions are in your LarKen manual. We shall not be responsible for your install job. AERCO owners need only the SPECTRUM EPROM for $10 $10 is forwarded to LarKen.

So you like to fly? The 747 Flight Simulator for Spectrum by Derek Ashton of DACC sold over 40K copies in EUROPE. Requires Spectrum Emulator. At this time supplied on LarKen SSDD disk only for $10 which goes to Derek Ashton, now working at MOTOROLA with Bob Swoger. Call in requests to Bob at W708-576-8068 H708-837-7957

ARTICLES

---

50 TIPS
compiled by Don Lambert

This is all from a little booklet that I acquired for the SPECTRUM. There was no copy right notice on the booklet nor even a publisher's by line. I thought that this was of sufficient interest to type it up for others to use. This also should work for the TS2068 with or maybe without the Spectrum ROM.

50 TIPS FOR BETTER SINCLAIR COMPUTING

0.> INTRODUCTION
Programming can be fun but often you will want to produce an effect on the computer though you are not able to find a routine to do it.

The Sinclair User team has also experienced those difficulties and we have decided to share with you a collection of 50 programming hints and tips written with reference to the letters which we receive by the sackful every week.

It is difficult to cover all the areas which make up the world of programming but we have tried to concentrate on the main aspects of writing in the BASIC language. Areas covered include graphics, sound, number base conversion, using joysticks with your own programs and efficient use of memory.

Nearly all the tips include programs or short routines which illustrate the points given, or contain step by step instructions on how each technique can be applied.

This booklet is aimed at those people who want to become more experienced in the art of programming. It is also, we hope, something which will spur those who have not tried to write their own programs to do so.

1.> STRUCTURED PROGRAMMING.
Structure is one of the all-time great snob words of programming. People with expensive machines frequently deride the Spectrum because BASIC "cannot support structured programming". Structured programming is a concept, and the principles can be used on any machine.

The idea to keep in mind is that of breaking down your program into a number of short blocks, preferably subroutines. For instance, an arcade game might have the following blocks:

    1000 set up the screen and va riables.
    2000 joystick/keyboard input.
    3000 result of move.
    4000 end of game.
    5000 set up user defined grap hics.

The rest of the program is then a simple manner of linking the routines together:

    10 GOSUB 5000: GOSUB 1000
    20 GOSUB 2000
    30 GOSUB 3000
    40 If the game is over THEN G OSUB 4000
    50 GO TO 20

Such a structure makes it much more easy to alter a program, or add extra features. You simply write another subroutine and adjust the section at the beginning which controls which subroutines are used and when.

2.> GOOD HABITS.
Always try to use names for variables which indicates the purpose of the variables. Variables such as "hor" and "ver" for horizontal and vertical coordinates are easier to follow than using "x1" and "x2" which are meaningless. BASIC was designed to follow English wherever possible, so there is every precedent for keeping the tradition.

Do not be misled into using single letter variables to save space. If you are writing a

program so long that space is important, you will need names which are easy to understand. You can always turn the variables into simple one- or two- character names at the end if it is necessary.

Use plenty of REM statements to remind you what a block of program does. You may know all about it at the time you write it but it is very easy to forget after a week away from the machine.

3.> USER DEFINED FUNCTIONS.

User defined functions are a gift to programmers, but few use them, perhaps because they are put off by the apparent complexity of the syntax.

Think of them as miniature subroutines, which do not need to be referenced by a line number, and you will appreciate their power.

The important thing to get right is the number of variables the function will use. When you define the function you write a general instruction with variables in it. When you use it you write the precise variables or numbers you want to use in the order in which they occur in the definition.

Here is a simple example using no variables at all. It simulates throwing a six-sided die.

```
DEF FN a()=INT(RND*6+1)
```

Later in the program the instruction LET X=FN a() will put a number from 1 to 6 into X.

Here is an example using variables. It adds two strings together with a space between them.

```
DEF FN a$(X$,Y$)=X$+""+Y$
```

Later in the program, you might have the following routine:

```
100 INPUT "First name?";C$
110 INPUT "Second name?";D$
```

```
120 LET N$=FN a$(C$,D$)
```

You could enter "John" and then "Smith". The function would combine the two to read "John Smith".

If you are writing a program using a lot of complex manipulation of numbers or text, you may well find it saves much space and trauma to use pre - defined functions.

4.> INFINITE SCROLL.

It can be frustrating, especially during a game, if the program keeps stopping and asking if you want to scroll the screen upwards.

Screen scrolling is controlled by one of the system variables, which counts the number of screen scrolls the computer will perform before it stops and asks you to press a key.

The line 10 POKE 23692,255 will disable the feature for 254 screens. If you want to make sure that you never have to press a key to scroll, then simply make sure that line is repeated every so often during the course of the program, by incorporating it into a loop, for instance.

5.> UDG STORAGE.

User defined graphics characters, or UDGs, are stored in a special part of memory which starts at address 32600. Each character is stored as eight consecutive bytes made up of eight bits each, of which one pixel, or dot, on the screen.

When displayed on the screen the eight bytes form a grid. Using standard UDGs there is only enough memory for letters A to U. Those are set up in RAM and can be changed by using instructions structured around the general formula POKE USR "graphics character" + byte number (1 to 8). For instance, the seventh byte of the letter A would be entered using the instruction:

```
POKE USR "A"+7
```

As the UDG areas of memory is protected the graphics set up within it cannot be destroy unintentionally unless you pull the plug. Not even NEW or CLEAR will do it.

6.> DEFINING UDGs.

To design and set up a UDG character within a BASIC program you need to plan the shape of the figure on a grid. Ink in the relevant squares to make up your character with a black or blue pen. Then take each horizontal line in turn and by it write the number sum of the line. That number is constructed by working from left to right. For each square that is linked in write a '1' and for every empty square put a zero. For instance, in a square with line one being white, white, black, white, white, black, white white.

When you have all eight numbers corresponding to the lines of the grid LOAD the binary/decimal converter which can be found elsewhere in the booklet. Type each of the binary numbers in, starting from the top of the grid and note the decimal values which the conversion program returns. At the end you should have eight decimal values. Those can be put straight into a data statement in the utility program below (line 50).

```
10 FOR K 0 TO 7
20 READ A:POKE USR "A"+K,A
30 NEXT K
40 PRINT"CHARACTER STORED"
50 DATA 0,0,0,0,0,0,0,0
```

Line 10 of the program sets up a loop which will go round eight times reading the decimal values of the DATA statement into the UDG area of memory. Line 20 does the reading and POKEs, or puts, those values into the UDG area. The character between quotes in line 20 is the graphics character which you want to change to the new representation.

Line 30 completes the loop and line 40 tells the programmer

that the set up procedure for that particular graphic has finished. Line 50 contains the decimal values which you obtained from the binary/decimal converter and in turn from the graphics character grid.

7.> USING UDGs.

Once set up, a UDG can be used in two ways within a BASIC program. The first way is to enclose it within a normal PRINT statement in quotation marks. When you have typed up to the point where you want to put the graphic character into a PRINT statement go into graphics mode by holding down the CAPS SHIFT key while you press the GRAPHICS key on the top row of the keyboard. Then type the key which will print the letter of the alphabet which you are using for the UDG - 'A', for instance.

When you press the key you should get a capital 'A' because the new representation is not set up until your program is run. Once you press the RUN key and then break after setting up is complete, you should see that the graphics capitals in the LISTing have changed to the spaceships, rockets or aliens which you created using the character generator routine.

The other way to display UDGs is simpler but you have to use the command CHR and the code number of the graphic which you want to appear. Those codes can be found on page 183 of your Spectrum manual. For instance, the code for the graphic 'A' is 144. To display that character all you have to do is type:

PRINT CHR$(144)

8.> ANTI-PIRACY.

Software piracy is a scrounge which is not easily eliminated. Most professional software companies incorporate anti-piracy devices within programs so that they cannot be broken into and copied, or the code in which they are written listed.

Security has always been a problem with BASIC programs as listings are easily accessible even when auto - run. There is only one simple way to combat the problem without resorting to machine code and that is to close off the avenues through which pirates can enter.

To get into a BASIC program which is running, the BREAK key is used and an error message is printed is printed at the bottom of the screen. The screen is made up of two parts - the lower half having access to a stream of information called #0. To stop a break - in, CLOSE down the stream. If information, such as an error message, tries to access it an error occurs and the system crashes.

In order to gain benefit from that knowledge all you have to do is make line 1 of any program you wish to protect:

```
1 CLOSE#0
```

9.> SOUND EFFECTS.
There is no doubt that sound is not the Spectrum's finest feature. It is very difficult to write anything remotely musical; you only have to listen to the hideous tunes produced by professional programmers to realize that.

It is possible, however, to do rather more than a few simple BEEPs at the beginning and end of the program. Experimentation is the key, but here are a few simple ideas to try out.

If your program requires single key input - for example, in choosing items from the menu - why not alter the length of the click made by the keyboard? You can do that by the instruction POKE 23609,x where x is the length of the click. Use any number from 0 to 255 until you find one you like.

Sirens are fairly easy. The secret is to slide from one note to another with very short BEEPs and then back again. Here is a short siren routine:

```
10 FOR X=0 TO 10 STEP .5
20 BEEP .02,X
30 NEXT X
40 FOR X=10 TO 0 STEP -.5
50 BEEP .02,X
60 NEXT X
```

The smaller the value after STEP, and the smaller the duration of each note, the smoother the sound.

The really revolting sounds, however, are produced by mixing two BEEPs together. The Spectrum has no facility for doing that, but you can alternate two sounds fast enough to create a nauseating tremolo which can form the basis of a variety of noises from screeches to explosions.

Here is one such piece of aural violence:

```
10 FOR X=-10 TO 10 STEP 2
20 FOR Y=1 TO 4
30 BEEP .02,.5
40 BEEP .02,X
50 NEXT Y
60 NEXT X
```

Because the two sounds are, on their own, very short, there needs to be a loop to repeat them a sufficient number of times for the noise to register in all its glory.

10.> NOVEL CRASHES.
Crashing, or bombing out, a Spectrum has become a spectators sport and many users have become fascinated, if not irritated, by the ways in which a Spectrum goes to pieces.

If you want to do it simply and deliberately one of the best ways is to type:

```
PRINT USR 1000
```

The machine should give off a buzz, the keyboard will become inoperable and a black border will appear on the screen. If you want to show how clever you are print SINCLAIR USER IS THE GREATEST on a Spectrum at a show or in a shop and then do your PRINT USR. The rest cannot be cleared without pulling the plug.

If you want something a little more colourful, but which does not give an irretrievable crash, then type:

RAND USR 5050

A multicoloured rectangle will appear on the screen followed by the Sinclair logo effectively resetting the machine.

Finally, if you have an EPSON FX-80 printer attached try:

RAND USR 600

11.> ROM BORDER.
It is possible to change the colours output to the border with a simple call to the Spectrum ROM.

The call is made to the operating system and the effect which you will get is similar to that obtained when SAVEing a computer program. First the blue and red bands, marking the header of the SAVEd program are shown and then the thinner blue and yellow bands of SAVEing. As no header information is available you will find that only one thick header signal is given. The command is:

RAND USR 1221

and the effect can be used in games to add effects during fight sequences or to show off a title page.

12.> OUT BORDER.
The colour of the screen border can also be changed using the OUT instruction which normally sends out a specified signal to a peripheral or device which is clipped onto the end of the buffer at the back of the machine.

It can also effect the screen on which its signals are represented as colours. In order to make use of the border in this way it is necessary to develop a short loop routine such as the one below.

```
10 FOR K = 1 TO 100
20 PAUSE 5
```

10

```
30 OUT 254,K
40 NEXT K
```

Try the routine and you will find that colours flash on and off the border. If you make the PAUSE number greater the colours will run through a slower sequence. A smaller PAUSE number, except of course zero, will speed up the colour cycle.

The advantage of using the routine rather than a ROM call is that you have more control over the timing of the effect.

13.> FOOLPROOF INPUTS.
Writing the perfect program is not much use if imperfect human beings are capable of crashing it. When users are called upon to enter instructions it is normal to use the INPUT command, but that can cause problems. Efficient programmers ensure that even if the user makes a mistake the program will not crash.

The best thing to do is use INKEY$ wherever possible. That can only be done simply with single characters as input, but it does prevent problems with incorrect input.

The line 10 INPUT X$ will fail if the inverted commas at the bottom of the screen are deleted accidentally. Replace with:

```
10 LET X$=INKEY$
20 IF X$=""THEN GO TO 10
```

Line 20 is necessary to make sure that the program will not continue unless a key is pressed.

[In the above example the machine is doing unnecessary processing, I prefer:

```
10 PAUSE 0
20 LET X$=INKEY$
```

Line 10 is necessary to make sure that the program will not continue unless a key is pressed. - Ed.]

But what if a number is required and not just any old character? Just add the line:

```
30 IF((CODEX$)<48)OR((CODEX$)>
57 THEN GO TO 10.
```

This will check that a number key has been pressed by examining the code of the character in question.

If you are using a menu the above method is by far the best, and as long as the relevant options are indicated by a consecutive block of letters or numbers the limits for the codes can be changed as appropriate.

If the program requires more than one character to be entered at a time, the INKEY$ routine could be made a subroutine and each individual character checked in turn to make sure it is of the right type. In that case you will have to write some lines to store all the different key presses in a single string, and then use that for text or the VAL command to turn the string into its numerical equivalent.

That will not entirely eliminate the possibility of error. It is therefore a good idea with very important data to PRINT it up to screen to be visually checked, with an opportunity for the user to press 'y' or 'n' - again using the INKEY$ routine for safety - to confirm the accuracy of the information.

Remember, do not just be kind to the computer, be kind to the user as well - even if it is only just yourself.

14.> LOW PRINT.
You may have noticed that the Spectrum screen display is made up of two parts. The first and main part controls the upper three-quarters of the screen and it is here that listings and program output are displayed.

The bottom quarter, usually made up of two lines, is the 11 part of the screen upon which commands can be typed and which the Spectrum uses to ask for INPUT during a program. Information is output to this and the other part of the screen using streams. The bottom part of the screen can be controlled using the reference number #0. Just as you could close down the bottom part of the screen using:

CLOSE #0

so you can print on it using a print instruction suffixed with #0. For instance PRINT #0; "USING BOTTOM SCREEN" would put the message between the quote marks at the bottom of the screen.

Other commands can also be prefixed with #0. These include PAPER, INK and CLS.

15.> EXPLOSIONS.
There are several ways in which explosions can be created on the screen. The simplest is make the main part of the screen flash with colour in a loop using a routine such as that below.

```
10 FOR K = 1 TO 50
20 FOR M = 1 TO 9
30 PAPER M:CLS:PAUSE 1
40 NEXT M
50 NEXT K
```

Unfortunately you will have to re-draw whatever was on the screen before the effect when it has finished as CLS has been used.

Alternately, you could flash the border using the ROM routine or loop described in the sections on border colour change elsewhere in the booklet. There is also another ROM routine which adds an extra dimension and that is:

RAND USR 1300

It is a cross between a quick SAVE signal and a colour.

16.> PUTTING IN ZERO.
You may have seen some programs which start with a line

numbered 0. That is a device used to protect programs against piracy. If a copyright statement is put into a line 0, it is difficult to remove unless you know something about how programs are stored in memory.

To create your own line 0, you must first write an ordinary line such as:

1 REM PIRATES WILL BE PROSECU TED!

Now you must find the beginning of the program area of memory, which is given by PRINT PEEK 23635+256*PEEK 23636. The number printed will be the address of the start of the program.

The first two bytes of any line in a program contains the line number, so if you POKE those two bytes with 0 your line 1 will turn into a line 0 which cannot be deleted or edited in the normal way.

Thus, if the start address was 23755 you would enter:

POKE 23755,0:POKE 233756,0

and the job would be done.

17.> REMOVING ZERO.
To remove a line 0 you must reverse the procedure outlined in hint 16. Again, find the start of the program area with PRINT PEEK 23635+256*PEEK 23636.

Now you must POKE the relevant line number into the two bytes at the beginning of the program.

Those bytes act as a pair, so that numbers bigger than 255, the limit for any one byte can be held. If you are familiar with Z80 machine code at all, you might expect the two bytes to hold the number back to front, with the 'least significant byte' first, like writing 12 as 21. But that odd convention is not used for line numbers on the Spectrum.

Because you can hold up to 255 in the second byte, the first byte stands for 256s. To find out how your line number should be POKEd in, divide it by 256 and note the remainder. Then POKE the answer to the division into the first byte, and the remainder into the second byte.

For example, if the start address was 23755 and the desired line number was 1, you would enter: POKE 23755,0:POKE 23756:1. If you wanted to have a line number 4368, you would enter: POKE 23755,17:POKE 23756,16 because 256 times 17 equals 4352, plus 16 equals 4368.

Remember always to use a line number which does not exist elsewhere in the program, or you may get into trouble.

If you wish to remove a line 0 from a commercial program, in order to rewrite it or hack it for your own purposes, that is legal. But if it is your intention to remove the line in order to copy the program in some way or pass it off as your won work, that is illegal and may even result in criminal proceedings. So be warned - you may think piracy sounds romantic but in reality it is just another name for thieving.

18.> RENUMBER.
A renumber routine is very useful for serious programming in BASIC. Often the line numbers get so crowded there is simply no room to squeeze in any more statements. The renumber routine below asks for the new start number and the step for each new line, in other words increasing the numbers in ones, five's or 100s as desired.

It works by POKEing the two bytes which carry the line number with a new number and then searching for the next line number. It will stop at 9900 rather then change its own line numbers, which would be disastrous.

One thing it does not do is to renumber GO TO or GOSUB statements. That would require a much more complicated routine and take far longer in BASIC. You will have to make a note of those yourself; REM statements are always helpful in identifying the beginning of a new program section.

```
      9900 LET START=PEEK 23635+
256*(PEEK 23636)
      9905 PRINT "ENTER SIZE OF
STEP":INPUT STEP
      9910  PRINT "ENTER FIRST N
UMBER":INPUT NUM
      9915 IF(PEEK(START+1)+256*
(PEEK START))>=8999 THEN STOP
      9920 POKE START,INT(NUM/25
6):POKE START+1,NUM INT((NUM/25
6)*256)
      9925 LET START=START+PEEK(
START+2)+PEEK(START+3)*256+4
      9930 LET NUM=STEP+NUM
      9935 GO TO 9915
```

19.> WHICH MODEL?
Sinclair research has so far brought out four versions of the Spectrum. You may not think that matters as all the models are bound to be the same.

Unfortunately, there are some differences and it helps to know which model you have in case special instructions are given in hardware manuals and the explanations of some BASIC listings provided in magazines.

To find out your model, be it 16K or 48K, type in:

10 PRINT IN 57342

There are several numeric answers to this request for information to be printed on the screen. If you own an issue one or two you should get the number 255. Issue three owners should get the number 191. That is not a hard and fast rule, however. Some users may get 161. That should not happen but if it does your machine is probably an issue three. They do say that nothing in the Sinclair world is certain.

20.> 48K TO 16K.
If you own a 48K Spectrum there may be times when you want it

to behave in a fashion similar to the 16K model.

The main reason is usually that you want to write a program for a 16K machine but cannot tell when you overstep the memory boundary on the 48K model. The other reason is that you want to see if a professional game or a program LISTing from a magazine will work on a 16K as your friend or relative is unlucky enough to have won one.

The method of performing the change is slightly complicated to explain but the use or it simple. Just type:

CLEAR 32600

The Spectrum will then only accept 16K programs. The reason is that the CLEAR instruction moves a barrier in memory about. If you type CLEAR without a number following it, that barrier will move to a pre - defined place within the RAM giving you access to the maximum amount of memory available.

If you move the barrier down the amount of memory available shrinks as more and more RAM is left in its wake. Moving it to address 32600 gives access to approximately 16K of memory.

21.> MEMORY REMAINING.
To discover how much memory is left, you must find out where the free space in RAM begins, after taking account of the length of your program, and the amount of space eaten up by variables. A short program which uses an array of dimensions (20,20,10) will take up an enormous chunk of memory for the array.

The Spectrum memory is arranged so that the area for variables and calculations is after the area holding the program. That location will inevitably vary with the length of the program. It is therefore given a system variable called STKEND which contains the address of the location. Another system variable, RAMTOP, gives the

address of the last byte of RAM available in BASIC; there may be memory available above RAMTOP, but that will either be occupied by information about user-defined graphics or machine code which you have stored there yourself by deliberately changing RAMTOP, so that you should already know how long that area is if you intend to use it.

The appropriate instructions to read those addresses and subtract one from the other as follows:

```
    10 LET STK=PEEK 23653+256*(
PEEK 23654)
    20 LET RMT=PEEK 23730+256*(
PEEK 23731)
    30 LET MEM=RMT-STK
    40 PRINT "MEMORY REMAINING
= ";MEM;"BYTES"
```

22.> PROGRAM LENGTH.
To find the length of a program written in BASIC, it is necessary to discover the addresses of the beginning and end of the section in RAM which holds it.

Luckily, the computer knows those addresses or it would not be able to find the program itself. The beginning is indicated by a system variable, START. The end is indicated by another variable, VARS, which is in fact the start of the variables area, which is space reserved for the variables set up by the program. That area is located directly after the program itself.

The following instructions will print the number of bytes occupied by the BASIC program, by PEEKing the contents of both system variables and subtracting the one from the other.

```
    10 LET PRG=PEEK 23635+256*
(PEEK 23636)
    20 LET VRS=PEEK 23627+256*
(PEEK 23628)
    30 LET LEN=VRS-PRG
    40 PRINT"PROGRAM IS";LEN;"
BYTES LONG"
```

Remember that the length of the

program is not the same thing as the space used when the program is running and all the variables have been set up.

23.> MONETARY FORMAT.
Many business programs require some form of monetary input showing figures in pounds and pence. Unfortunately some business users are not as adept at figures as they should be and press extra keys giving the computer ridiculous numbers. You would look silly if your balance showed a total of $50.03233.

The program below will look at each input number and chop off any extra figures entered. It will also give the number a pound sign as a prefix.

```
10 INPUT A
20 LET A$=STR$(A)
30 FOR K=1 TO LEN A$
40 IF A$(K TO K) ="." THEN LET
B=K
50 NEXT K
60 LET B$=A$(B TO)
70 LET B$=B$(1 TO 3)
80 LET A$(1 TO B-1)+B$
90 PRINT VAL(A$)
```

Use the routine as a subroutine in one of your programs and any value input to variable 'A' will be formatted.

24.> JOYSTICK COMPATIBLE.
Using a joystick to control action within one of your own BASIC programs may seem difficult but nothing could be easier. It is only the Kempston 'standard' with which you may have difficulty.

Most of the other interfaces use the character codes generated by the cursor keys. For instance, if you pushed the joystick to the left the Spectrum would register the code 08.

The Kempston interface is not so simple to use. To read it you can use the program routine below.

```
10 LET JOY = IN 30
20 IF JOY = 3 THEN PRINT "UP"
30 IF JOY = 4 THEN PRINT "DOWN"
```

```
40 IF JOY = 1 THEN PRINT "RIGHT
"
50 IF JOY = 2 THEN PRINT "LEFT"
60 GO TO 10
```

As soon as you push the joystick in a new direction the IF...THEN statements will check the direction in which the change has been made. You can, of course, turn the program into a subroutine and put your own actions after THEN. If you want the "fire" option active within your game you will have to add an extra line to the program.

```
55 IF JOY = 16 THEN PRINT "BANG
"
```

25.> COLOUR CODES.

There are two ways to use colour on the Spectrum. One is to write INK and PAPER statements to set the colours you want, the other is to use control codes.

The colour control codes are the easiest to use because the keys you use have the colours written over them.

To use a colour control code, go into Extended Mode - CAPS SHIFT and SYMBOL SHIFT -. Then hold down CAPS SHIFT and press the key with the appropriate colour. The cursor will change colour, and any character typed will now be in the appropriate colour.

If you delete the line, you will find the cursor deletes a hitherto unseen question - mark at the point where the colour was changed. That is the control code, invisible in the program LISTing but stored in the computer memory all the same.

You must remember to use another colour code at the end of the colour section to return to the original colour, usually black ink on white paper.

The colour code only affects the ink colour. If you want to change the paper colour you could either use PAPER or perhaps an additional code for

inverse Video, thus swapping the paper and ink colours around. You can write codes in a row to get the required effect.

The Spectrum should usually handle control codes perfectly, but sometimes produces the 'dreaded buzz' at which point the computer issues a low growl and refuses to accept any input, including delete.

To deal with the buzz, just EDIT a line down from the main program to wipe out the current line and start again.

The precise causes of the buzz are beyond the scope of this booklet. Think of it as one of those idiosyncratic Clivebugs which we must all learn to love.

26.> CHARACTER CODES.

Character control codes can be used in place of functions such as SPACE, ENTER, CAPS LOCK and can also be used to print characters on the screen.

If you want to print a character without having all the bother of enclosing them within quotation marks you can use the principal control code instruction which is CHR$. For instance if you wanted to type the message "HELLO" on the screen you could either print:

```
PRINT "HELLO"
```

or you could use the command:

```
PRINT CHR$ (72) +CHR$ (69) +CHR$76+
CHR$ (76) +CHR$ (79)
```

Those codes can be found on page 183 of the Spectrum manual and each letter is put in a separate CHR$ statement. The '+' operator adds one character to another so that they appear on one line together. You could force a new line just by putting the control code in between the letters. The code of ENTER is 13 so the line

```
PRINT CHR$ (65) +CHR$ (13) +CHR$ (66
)
```

will produce 'A' followed by 'B' on the next line.

27.> INVERSE VIDEO.
Using inverse video is as easy as using the colour codes, but people get confused because there are several ways of doing it, notably two INVERSE keys.

The one at the bottom of the keyboard (E-mode key M) is a command just like PAPER and INK. You follow it with a 1 for inverse and a 0 to cancel the command.

Key 4 has Inverse video printed above it, and in conjunction with CAPS SHIFT produces the control code which will turn any further characters into inverse form.

Key 3 is the control code to reverse the operation. Remember that, unlike the colour control codes, you do not go into Extended Mode to get the codes. Just hold down the CAPS SHIFT.

One set of codes Sir Clive did not print on the keyboard are those for making the characters flash. They can be found by going into Extended Mode, holding down CAPS SHIFT, and pressing keys 8 and 9. Key 9 turns the flash on and key 8 turns it off.

To make matters even more complicated, a completely different method is used to obtain the inverse of the block graphics on keys 1 to 8. To get those, all you need to do is hold down CAPS SHIFT while in graphics mode.

When the ZX-81 was produced it had all the block graphics possible each on its own key. But there were so many extra functions to include on the Spectrum that the set got chopped in half, with one half the inverse of the other half.

The change is an excellent example of what computer manu-facturers call 'enhancement', usually a new way of doing things which makes life twice as difficult for the user.

28.> SOFT-RESET.
When programming in BASIC it is annoying to have to pull the plug every time you want to remove some user defined graphics quickly from the graphics area of memory or nullify the effects of a program so that the system is reset.

It is possible, however, to reset the system:

    Rand USR 0

When you type the command in and execute it by pressing ENTER the screen should display a large black rectangle which, after a few seconds will disappear to be replaced by the Sinclair logo.

The explanation is simple. Typing RAND USR will invoke machine code at a specified address. The number zero, after the command, sets the Spectrum running at address 0000 which is the start of the BASIC interpreter and operating system.

29.> SHORTHAND LOGIC.
Although BASIC was designed to be as like English as possible, the logic operations: AND, OR, =, < and so on are often confusing unless used in very simple forms. In fact, used properly, they can increase the speed of a program.

In Sinclair BASIC, the expression IF A=B THEN GO TO 100 has an obvious meaning. However, the computer works the meaning out in a rather odd way. The expression A=B is treated as a number, just like SIN X has a numerical value. The value of A=B will be 1 if true and 0 if false. The same applies to other operations.

That allows you to rewrite complicated IF statements in a shorthand logic which relies on the numerical values.

Take the line:

    10 IF A=B THEN LET X=X-1

That could also be written as follows:

```
10 LET X=X-(A=B)
```

If A does not equal B then X does not change. If it does then 1, the value of A=B, is subtracted from X.

There will be already a small saving in speed but because IF is no longer used, a number of similar conditions can be added together.

```
10 IF A=B THEN LET X=X+1
20 IF A>B THEN LET X=X-1
30 If A<B THEN LET X=X+7
```

can be written:

```
10 LET X=X+(A=B)-(A>B)+((A<B)*
7)
```

Good programmers may use a dozen or more such conditions in a single line to save both time and space.

The concept can be expanded using the following further facts: (A OR B) is 1 if either A or B is not equal to zero. IF A THEN LET X=X+1 means exactly the same as IF A<>0 THEN LET X=X+1. (A AND B) has the value of A if B is not equal to zero. (NOT A) is 0 if A is not zero. The full list can be found on page 64 of the original Spectrum manual.

Logic operations are not easy to follow without practice, but when mastered you will find your programs are far more efficient and less likely to carry mistakes than if you use dozens of IF statements.

30.> BETTER DRAW.
Drawing line figures on the screen is irritating because it usually means that you have to write lines containing long lists of coordinates mixed up with DRAW and POINT commands.

A simple way around that is to write a subroutine consisting of one DRAW command. Then set up your coordinates in an array and feed them through two simple variables called x and y. In that way you will be better able to keep track of what you draw on the screen. The routine will look something like the program outlined below.

```
10 DIM A(6): REM EVEN NUMBERED
ARRAY FOR CO-ORDINATES
 20 SET UP...
 30 AS MANY...
 40 VARIABLES...
 50 AS YOU NEED...
 60 LET X=NEW X CO-ORD
 70 LET Y=NEW Y CO-ORD
 80 GOSUB DRAW
 90 REM SUBROUTINE DRAW
100 DRAW x,y
110 RETURN
```

Take above outline as your guide and you will find writing graphics routines easier.

31.> NUMERIC SORT.
Sorting information into ascending order is something with which business people are familiar. The program below is a simple sort routine. It takes numbers from one array and transfers them in order of ascendancy into another.

```
 10 LET N=0
 20 DIM A(5)
 30 FOR K = 1 TO 5
 40 INPUT A(K)
 50 NEXT K
 60 LET B= A(1)
 70 DIM C(5)
 80 FOR M = 1 TO 5
 90 FOR K = 1 TO 5
100 IF A(K)B THEN LET B=K:LET
D=K
110 NEXT K
120 LET C(N)=A(D)
130 LET N = N+1
140 LET AD = 9999
150 FOR K = 1 TO 5
160 PRINT C(K)
170 NEXT K
```

32.> SORTING CHARACTERS.
Sorting characters into alphabetical order is no more difficult than sorting numbers. The only problem is that you have to find the numeric value of each character using the CODE instruction.

In the alphabet the letter A comes before the letter B. Unfortunately the computer has

no such knowledge but as each character has a number representation you can compare the values the IF...Then statement.

The simplest way to sort a string of characters, making sure that they are all capital letters, is to dimension three variables. The first is empty but will hold the result, the second holds the string to be sorted and the third is the letter under consideration or the last lowest value character.

Look at the first character in the string and store its value. Then look at the second character. If the value of the second character is lower than that of the first then put that value into the variable, replacing the first one. Then go to the next character on the string and compare it with the last value.

When you get to the end of the string the first time around put the character whose value is in the variable into the empty string. Then check to see if the two strings are of the same length. If they are not then the sort has not been finished and a further round of comparisons must take place.

After the first round of comparisons has been made, however, a new variable must be defined. That contains the value of the last letter put into the string which you are building out of nothing. Then every comparison must satisfy two conditions. Is the value lower than the preceding one and is it higher than the last letter entered into the string which is being built? If the answer is yes to both conditions than the value replaces the old lowest value and the comparison continues. Comparisons will continue to be made until the two strings are equal, that is, they have the same number of characters in each, the latter being sorted into alphabetical order.

33.> AUTO CAPS LOCK.
When you are writing a program which needs to recognize pieces of text entered via the keyboard by the user, the last thing you want to do is check every input to see if the CAPS LOCK is on or lower case letters are being used.

The best way to avoid that is to put the CAPS LOCK on automatically within the program before asking for information.

    POKE 23658,8

The address 23658 tells the computer which cursor mode it is in. K means command, L means line and C means Capital letters. The above command will give you a 'C' cursor and typing INPUT will provide you with capitals. Type in the program below and you will see for yourself.

```
10 POKE 23658,8
20 INPUT A$
30 PRINT A$;
40 GOTO 10
```

34.> BINARY/DECIMAL.
The following program converts Binary numbers into decimal. You may find it useful for working out the correct decimal numbers for UDGs - see tip 6 - as well as other uses.

```
10 REM BINARY TO DECIMAL
15 LET DEC = 0
20 PRINT "ENTER BINARY NUMBER"
30 INPUT A$
40 LET X=LEN A$
50 FOR N=X TO 1 STEP -1
60 LET DEC=DEC+VAL(A$(N))*21(X
-N)
70 NEXT N
80 PRINT "DECIMAL VALUE OF";A$
;":"'';DEC
90 STOP
```

35.> DECIMAL/BINARY.
The following program will convert decimal numbers into binary.

```
10 REM DECIMAL TO BINARY
20 PRINT "ENTER DECIMAL NUMBER
"
30 INPUT X:LET DEC=X
40 LET A$=""
```

```
 50 IF INT(DEC/2)=(DEC/2) THEN
GO TO 100
 60 LET A$="1"+A$
 70 LET DEC=DEC-1
 80 IF DEC=0 THEN GO TO 150
 90 GO TO 110
100 LET A$="0"+A$
110 LET DEC=DEC/2
120 GO TO 50
150 PRINT "BINARY FOR";X;"IS"''
;A$
160 STOP
```

36.> HEX TO DECIMAL.

It is often necessary to convert a hexadecimal value into denary, or base 10, in order to poke those values straight into memory. As you may realize most magazines list machine code programs in hexadecimal code. The program below will convert hex to denary.

```
 10 LET SUB=1:LET DEC=0
 20 INPUT "HEX"; X$
 30 LET N$=X$(LEN X$)
 40 LET X$=X$( TO LEN X$-1)
 50 GO TO 120
 60 IF LEN X$ THEN GO TO 30
 70 PRINT "DENARY ";DEC
110 STOP
120 LET F=CODE N$
130 IF F<=57 THEN LET L=F-48
140 IF F=>65 THEN LET L=F-55
150 IF F=>97 THEN LET L=F-87
160 LET DEC=DEC+L*SUB
170 LET SUB=SUB*16
180 GO TO 60
```

37.> DECIMAL TO HEX.

As many monitor programs display machine code in the hexadecimal number notation, where one to nine is followed by A to F, representing numbers 10 to 15, a conversion program for decimal to hexadecimal is required. In the program below all you have to so is type in the denary, base ten number, and it is automatically converted to hexadecimal, base 16.

```
 10 INPUT "DENARY:";DEN
 20 LET BAS=4096
 30 LET X$=""
 40 LET SUB=INT(DEN/BASE)
 50 GO TO 130
 60 LET DEN=DEN-SUB*BASE
 70 LET BASE=INT(BASE/16)
 80 IF BASE<1 THEN GO TO 100
 90 GO TO 40
```

```
100 FOR K=1 TO 3:IF X$(1)="0" T
HEN LET X$=X$(2 TO):NEXT K
110 PRINT "HEXADECIMAL:";X$
120 STOP
130 IF SUB<=9 THEN LET X$=X$+CH
R$(SUB+48)
140 IF SUB>9 THEN LET X$=X$+CHR
$(SUB+55)
150 GO TO 60
```

38.> STRING SEARCHING.

To search for a string within another string you will need two string variables. The first contains the string upon which the search is to take place and the second contains the string for which we are searching. The program below demonstrates the procedure.

```
 10 INPUT "MAIN STRING";M$
 20 INPUT "SEARCH KEY";S$
 30 FOR K = 1 TO LEN(M$)
 40 IF K LEN(S$)>LEN(M$) THEN G
O TO 60
 50 IF S$ = M$(K TO (LENS(S$))
THEN PRINT "MATCH AT ";K
 60 NEXT K
```

Line 10 asks for the main string to be entered and line 20 accepts the string for which you are searching within M$. Line 30 sets up a string which will loop until all of M$ has been searched. Line 40 will make sure that M$ is long enough to cope with a search using the S$ key. If S$ was bigger than M$ an error could occur. If that is the case the comparison is skipped and the loop is executed harmlessly until it has finished.

Line 50 performs the search. It compares S$ to the part of the M$ starting at character 1, then continues with character 2 and so on. If a match is made the placing at which it is found within M$ is printed. If no match is found no message printed.

39.> CENTERED TEXT.

The following routine examines a string of text and centers it on the screen so that the left and right margins are equal. Although the routines requires you to enter a line of text, which should not contain more than 32 characters, it is best

used in the form of a subroutine in part of a longer program to deal with headings and the like.

```
10 PRINT "ENTER YOUR LINE OF T
EXT"
20 INPUT A$:CLS
30 LET X=LEN A$
40 LET X=INT(32-X)/2
50 PRINT AT 0,X;A$
60 STOP
```

If you want to center more than one line of text, change the 0 in line fifty to a variety such as Y. Then arrange to have Y increase by one each time the routine is used, so that the lines will be printed one under the other.

40.> RANGING RIGHT.
Sometimes, when printing text on the screen, it is desirable to have a right hand margin straight and the left hand margin ragged, the reverse of normal procedure.

The following routine takes a line of text, which must be no more than 32 characters long, and prints it to the right hand margin.

```
10 PRINT "ENTER LINE OF TEXT"
20 INPUT A$: CLS
30 LET X=32-LEN A$
40 PRINT AT 0,X;A$
50 STOP
```

The routine is obviously of greater use as a subroutine to be used when required. In that case you would probably want to adjust the value 0 in line 40 to indicate how far down the screen the text should be printed, by using a variable which could be altered as desired.

41.> 255 COLOURS.
Few people realize that the Spectrum can display more than its allotted nine colours. It is a simple method which combines UDGs and the little - used OVER instruction.

First construct two UDGs in the chessboard fashion shown in the two diagrams above (not given). Then you are ready to print new

colours. The program below will demonstrate the technique and uses the UDGs "A" and "B" which have already been set up.

```
30 PRINT AT 10,10;INK;"A"
40 OVER 1
50 PRINT AT 10,10;INK "B"
60 OVER 0
```

What you should have on the screen after running the routine is a character square containing a new colour. Experiment. By controlling the INK colour with a simple FOR NEXT counting loop you can cycle through all the 255 colours available.

42.> SLOW PRINT.
To add a science fiction effect to your armory of game programming ploys you could try the sub-routine shown below.

```
10 LET A$="THIS IS A MESSAGE"
20 FOR K=1 TO LEN(A$)
30 PRINT A$(K TO K);:PAUSE 5
40 NEXT K
50 PRINT
```

You will find the printing of the message which has been put into the variable A$ is slowed down by the PAUSE in line 30. Your computer gives you a display similar to those in the big science fiction movies where printing is slowed down on the screen to give a dramatic effect. Any self - respecting programmer would die rather than have a program acting as slowly in real life but a game display can be enhanced especially if you redefine the character set to produce square, futuristic, lettering.

43.> TEXT SEARCH.
When writing programs such as adventure games it is often convenient to have a string which contains a whole series of words and end - to - end, in order to save space instead of using huge arrays. The problem is to search for a particular word in the string.

The following program does that, using as an example a list of directions which, in

the context of an adventure
game, might refer to the
various movement commands.

```
10 LET B$="NORTHSOUTHEASTWESTU
PDOWN"
20 INPUT "ENTER YOUR WORD";A$:
IF A$=""THEN GOT TO 20
30 LET X+LEN B$-LEN A$+1
40 LET N=1
50 IF B$(N  TO  LEN  A$)1))=A4
THEN GO TO 100
60 IF N=X THEN GO TO 200
70 LET N=N+1:GO TO 50
100 PRINT "YOUR WORD IS IN THE
LIST"
110 STOP
200 PRINT "YOUR WORD IS NOT IN
THE LIST"
210 STOP
```

Of course, in a real adventure
game you might also want to
check the validity of the word
found, to avoid the program
accepting something like "UP
DOWN" which is in the string.
That might be done by having
spaces between the words, and
checking the characters on each
side of the word found. There
might also be a set of numbers
or characters after each word
which the computer would use to
implement the instruction: for
instance, the number of a
subroutine to perform the
action "GO NORTH".

44.> HEX LOADER.
The following program will
allow you to enter a block of
machine code in hexadecimal
form. You may input as many
pairs of hex code characters as
you wish, and the letter S will
stop the program when you have
finished.

The program first asks for the
starting address to which the
code is to be sent.

```
10 INPUT "STARTING ADDRESS";ST
20 INPUT "ENTER STRING OF
HEX";A$
30 IF A$=""THEN GO TO 20
40 IF A$(1)="$" THEN GO TO 130
50 LET A=CODE A$(1):LET B=CODE
A$(2)
60 LET A=A=(A>64)*55-(A<58)*48
70 LET B=B-(B>64)*64)*55-(A<58
)*48
80 LET C=16*A+B
90 POKE ST,C;LET ST=ST+1
```

```
100 IF LEN A$=2 THEN GO TO 20
110 LET A$=A$(3 TO)
120 GO TO 40
130 STOP
```

Do remember to enter all the
letters in the hex code as
capital letters.

45.> FILL.
You can draw lines and circles
on the Spectrum, and it is easy
to link them to form all sorts
of shapes. But unfortunately
there is no command you can use
to colour them in.

There are several ways of
filling in shapes. One is to
draw a series of lines one on
top of the other, instead of
the outline, but that requires
a formula for the shape. It is
easy with triangles or squares
but more complex with curves or
irregular shapes.

The following program is one
solution to the problem. It
examines each pixel in turn,
scanning from left to right
across the screen. When it
finds a dot, it starts drawing
a row of dots until it comes
across another one. The program
is slow, but it does the job; a
really good fill routine would
require machine-code, which is
beyond the scope of this
booklet.

As an example, the program
starts by drawing a square.

```
10 CLS
20 PLOT 165,120: DRAW 0,-81:DR
AW -81,0: DRAW 0,81: DRAW 81,0
30 LET KON = 1
40 FOR Y=175 TO 0 STEP -1
50 FOR X=0 TO 255
60 IF POINT(X,Y)=1 THEN LET KO
N=1-KON
70 IF KON<1 THEN PLOT X,Y
80 NEXT X
90 NEXT Y
```

The program is very slow. In
can be improved by altering the
limits of the variables x and y
in lines 40 and 50. At present
they cover the whole screen,
but if you know the area in
which the shape it be filed is
located, then you could have

the program scan only that narrow range.

46.> CHARACTER CHANGE.

User defined graphics are all very well, but there are only 21 of them. If you want more you can completely rewrite the character set. To do that you must construct your own characters, store them in memory, and then convince the Spectrum to use yours instead of those in ROM.

It is not as difficult as it sounds, Let us assume you want to alter the alphabet and numbers only, which will leave the keywords and the like unchanged thus avoiding too many problems reading your program LISTing. It will nevertheless give you 82 possible UDGs, which should be more than enough.

The following program will do the job. Numbers underlined and in brackets are equivalents for the 16K Spectrum.

```
10 CLEAR 63999(30999)
20 FOR X=0 TO 767
30 LET Y=PEEK(15616+X)
40 POKE 64000(31000)+X,Y
50 NEXT X
```

The program simply transfers the existing character set to an area above RAMTOP. You are now set to alter those characters you wish to change.

Prepare your character set as if you were using user define graphics. To load them in, use the following routines:

For the numbers 0-9

```
60 FOR X=64128(31128) TO 64207
(31207)
70 READ NUM
80 POKE X, NUM
90 NEXT X
100 DATA......
```

The data statements contain the new design for the character as in creating UDGs, i.e. a series of eight numbers for each character, either as decimal or binary numbers using BIN. For

capital letters change line 60 to:

```
FOR X=64264(31471) TO 64471
(31471)
```

For lower case letters change line 60 to:

```
FOR X=64520(31520) TO 64727
(31727)
```

If you want to change the entire set, use the values for X given in the first program in line 60 of the second.

Having completed the hard work, you are now set to fool the computer into preferring your set over its own. When the computer wants to print a character, it looks at a system variable called CHARS, which contains the address in ROM of the character set. You can change that address to make it refer to your own set.

The following instructions make the appropriate change:

```
10 POKE 23606,0(24)
20 POKE 23607,249(12)
```

You will of course want to save the new character set and load it in to one of your programs later. Save it using SAVE "name" CODE 64000(3100),768 and LOAD it in using something like:

```
10 CLEAR 36999(30999):LOAD""CO
DE
```

at the beginning of the program.

47.> LIST FILES.

The following programs lists the names of programs on a tape and loads the one selected. It stores the names of up to ten programs, and each name can be up to ten letters long. Machine - code programs should be given a name starting with the letter "X".

```
10 REM LISTING PROGRAMS
20 DIM A$(10,10)
30 PRINT"PRESS'L'FOR A LIST OF
PROGRAMS"'"PRESS'N'TO NAME PROG
RAMS"
```

```
40 GO TO 50+(INKEY$="L")*50+(I
NKEY$="N")*100
 50 GO TO 40
100 CLS
110 PRINT # 0;"SELECT A TO J TO
LOAD PROGRAM"
120 FOR X=0 TO 9
130 PRINT AT X,0;CHR$(X+65),A$(
X,1 TO)
140 LET B$=INKEY$
150 IF B$=""THEN GO TO 140
160 GO TO 170-(((CODE B$)<65)OR
((CODE B$>74))))*30
170 LET NUM=(CODE B$)-64
180 CLS:PRINT"START TAPE AND PR
ESS ANY KEY":PAUSE 0
190 IF A$(NUM,1)="X"THEN GO TO
300
195 LOAD A$(NUM,1 TO):STOP
200 CLS
210 FOR X=0 TO 9
220 PRINT AT X,0;CHR$(X+65),A$(
1 TO)
230 NEXT X
240 PRINT"""TO CHANGE OR ADD NA
ME,"'"PRESS LETTER AND ENTER NA
ME"
245 PRINT "PRESS $ TO STOP"
250 PRINT "START M/C NAMES WITH
THE LETTER X"
260 LET B$=INKEY$;IF B$="" THEN
GO TO 260
265 IF B$="$" THEN GO TO 30
270 IF(((CODE B$<65)OR (CODE B$
> 75) THEN GO TO 260
280 INPUT "ENTER NAME";C$:IF LE
N C$>10 THEN GO TO 280
290 LET A$(76-CODEB$)=C$:GO TO
200
300 LOAD A$(NUM,1 TO)CODE:STOP
400 SAVE "FILE" LINE 30
```

Two notes: Always write your
program name in capital
letters, and always save the
program by typing GO TO 400.

48.> ANIMATION.

Animation of a character could
not be simpler but it is the
most effective technique which
can be used on the screen.

Firstly you must design the
character which you want to
animate and put it in a
character square. Then make it
into a UDG using the techniques
outlined elsewhere in this
booklet.

Then using character squares
again, show the positions into
which your characters can get.
For instance a stick man or

woman might move arms or legs
up and down. Design UDGs which
illustrate that the final
positions into which the limbs
will move.

Now to the animation. Print the
first character which you
designed onto a position on the
screen. From here you will have
to make a choice. If your
character is to stand still,
use the sequence of movements
which you have created with
UDGs in the same position. As
you print one, then the next
and then the next, so the
impression of movement will be
achieved. The effect is rather
like sketching many pictures if
a pin man hundreds of times on
different pages in an exercise
book but in different poses. As
the pages are flicked so the
pin man appears to move.

If you want the man to move
about the screen you can use
the PRINT AT instruction to
move him up, down or across the
screen. The movement effect
will be the same and the eyes
will be deceived.

49.> CONTROL KEYPAD.

A control keypad is simply a
set of keys to manipulate what
is displayed on the screen,
usually to move an object
around such as the hero of an
arcade game.

The following routine uses keys
Q, A, O and P for Up, Down,
Left and Right to move a
character one space at a time.
It also checks to ensure the
character does not move off the
edge of the screen. The
character used is "?", and
starts at coordinates—ordinates
10,15:

```
 10 REM CONTROL PAD
 20 LET X=10:LET Y=15
 30 LET X=X+(INKEY$="P")-(INKEY
$="0")
 40 LET Y=Y(INKEY$="Q")+(INKEY$
="A")
 45 GO TO 50+(INKEY$="S")*50
 50 LET X=X-(X>31)+(X<0):LET Y=
Y-(Y>21)+(Y<0)
 70 PRINT AT Y,X;"?"
 80 GO TO 30
100 STOP
```

23

Lines 30 to 50 use logic operations instead of IF statements. A full explanation is given in tip 29. Line 45 makes the program stop if "S" is pressed, and go to line 50 if not.

50.> RANDOM NUMBERS.

If you use the RND function on its own you will get a result which falls between 0 and 1. It is, of course, real and contains a decimal point. Try PRINT RND and you will see the effect.

In order to get a range of integers you must use the INT function. For instance, you may want to find a random number in a dice throw, one to six. To do that use the formula:

INT(RND end-of-range) start - of - range

The dice would, therefore, be simulated using:

PRINT INT(RND*6)+1

If, however, you wanted a range between three and seven you would type:

PRINT INT(RND*5)+3

The random number generator can also be set by using RAND, followed by a number or not. Random numbers are not calculated: they are looked up in a table of figures. If you type RAND 3 the Spectrum will start by giving the third number in the table followed by the fourth and so on.

If RAND is used without a number the computer will start at a point determined by the number of separate television frames generated since the Spectrum has been switched on. That should be random enough for you.

ADS

To put an AD in the BBS and newsletter, upload a file with the filename.filetype:
        TSNUGxxx.ADS
  where xxx is your initials.
! ! ! Our ADS are free   ! ! !

Your ADS appear in FOUR different newsletters!