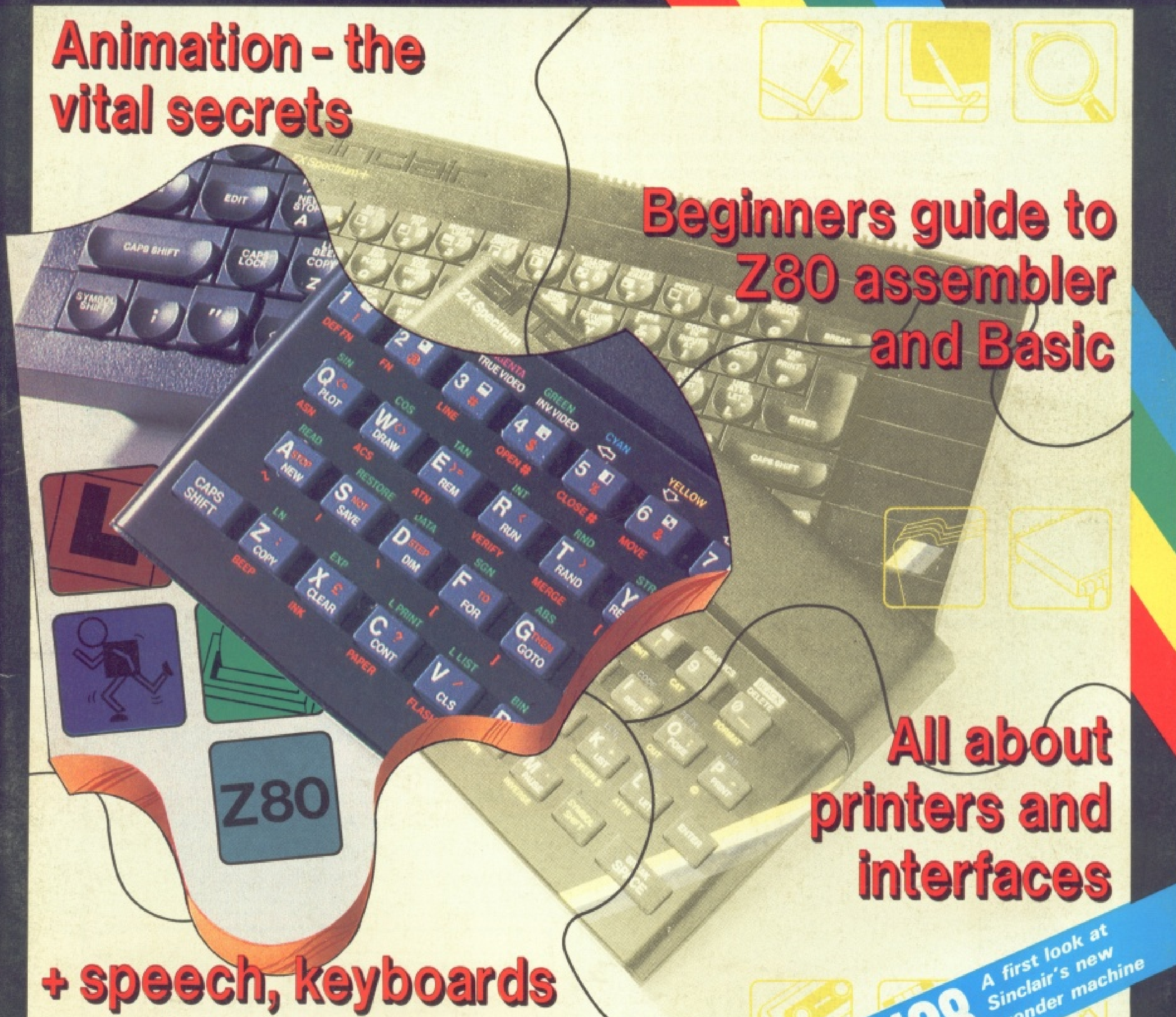
The COMPLETE

PART 3

SPECTRUM

All you want to know about the world's best-selling computer

A Database Publication



Animation - the vital secrets

Beginners guide to Z80 assembler and Basic

All about printers and interfaces

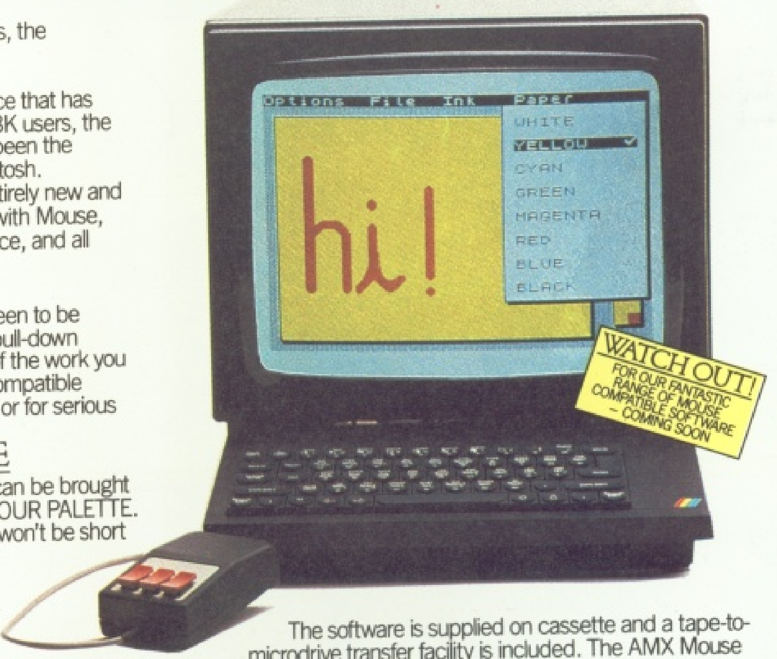
+ speech, keyboards and databases

SPECTRUM 128 A first look at Sinclair's new wonder machine

In six monthly parts

£1.50

We thought it was about time we put you in the picture.



When we introduced our AMX Mouse to micro-users, the response was phenomenal.

And no wonder!

Hailed by the press as 'probably the best input device that has arrived recently', the AMX Mouse brings to Spectrum 48K users, the same sophisticated, positive control that has, until now, been the province of more expensive computers – like the Macintosh.

The fantastic AMX Mouse Package opens up an entirely new and exciting world to Spectrum users and comes complete with Mouse, interface, which also includes a Centronics printer interface, and all these fabulous programs.

AMX ART

This computer aided, drawing program has to be seen to be believed. Making full use of on-screen windows, icons, pull-down menus and pointers, you'll be astonished at the quality of the work you can produce, save and print using either ZX or Epson compatible printers. It's a program ideal for both hours of family fun or for serious professional applications.

AMX COLOUR PALETTE

The wonderful pictures you create with AMX ART can be brought vividly to life with rich vibrant colours – using AMX COLOUR PALETTE. And with 8 Foreground and 8 Background colours you won't be short of inspiration.

AMX CONTROL

Now you can create a 'Mouse environment' in your own programs, AMX Control adds 28 commands to normal Sinclair Basic and contains three programs.

1. The machine code program to extend the Basic interpreter, which gives you full use of windows, icons, pull down menus, pointers and also supports AMX printer interface.
2. An Icon designer, an individual program for creating and storing icons for use in your own programs. The number and variety of icons you can create is limited only by your needs and imagination.
3. A demonstration program containing on screen calculator and puzzle.

This fabulous AMX Mouse Package costs only £69.95, a price as remarkable as the package itself and it includes a fully illustrated operating manual.

The software is supplied on cassette and a tape-to-microdrive transfer facility is included. The AMX Mouse package is compatible with the Spectrum 48K and Spectrum +.

Ordering couldn't be easier.

This superb package is available from all good computer dealers or direct using the freepost order form below.

So what are you waiting for?

Get into the picture now!

☎ FOR INSTANT ACCESS OR VISA ORDERS RING (0925) 602959/602690

PLEASE RUSH ME POST FREE ________ (Quantity)
AMX MOUSE SPECTRUM 48K PACKAGE/S
AT £69.95 EACH INCL. VAT AND P & P.

I ENCLOSE CHEQUE/POSTAL ORDER FOR £ ________

OR DEBIT MY ☐ ACCESS
(TICK AS APPROPRIATE) ☐ VISA

CARD NO. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

EXPIRY DATE ________

SIGNATURE ________

NAME ________
(BLOCK CAPITALS PLEASE)

ADDRESS ________

________ POSTCODE ________

AMX MOUSE

IMAGINATION AT YOUR FINGERTIPS

SEND TO: ADVANCED MEMORY SYSTEMS LIMITED, FREEPOST, WARRINGTON WA4 1BR.

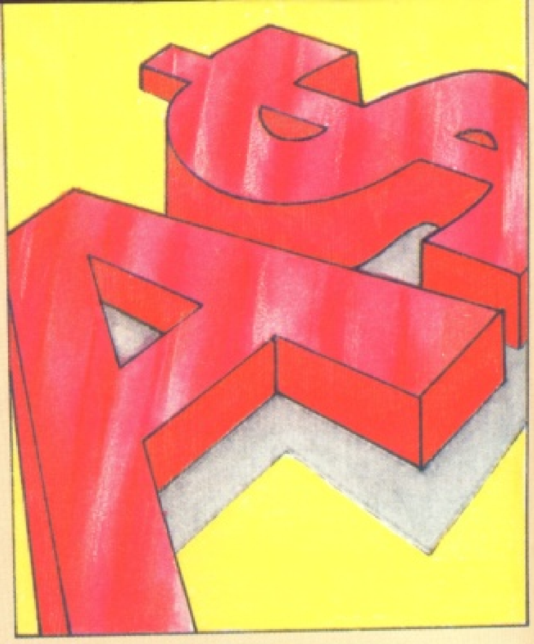
In Part Three...

The COMPLETE SPECTRUM
All you want to know about the world's best-selling computer

is published by
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.
Tel: 061-456 8383

News trade distribution by:
Europress Sales & Distribution Ltd,
11 Brighton Road, Crawley,
West Sussex RH10 6AF.
Tel: 0293 27053.

A variable solution



Third in the series that aims to explain the elements of Basic for absolute beginners

WE'VE come a long way in our programming career in the last two articles. Although you can't yet do anything spectacular with your Spectrum, you've learnt a lot of basic Basic that should stand you in good stead.

So far, we've explored how to use the PRINT command on both numbers and strings and seen the difference between both. We've seen how to let names stand for numbers as we used LET to create numeric variables and employed them in your first programs.

It's from this solid groundwork that we'll be taking off now. After seeing how strings can be held in variables we'll be exploring two of Basic's most fundamental, yet powerful, structures.

Previously our programs have been fairly fixed and rigid. Our use of INPUT to give values to variables while a program is actually running will put an end to that. And rather than having our programs go from one line to another in a relentless sequence, we'll be seeing how the FOR . . . NEXT loop structure can make our programs more efficient and powerful.

By the time we've finished your programming skills will have improved amazingly and you'll be starting to tap the full potential of your Spectrum.

By now the program:

```
10 LET count=1
20 LET count=count+1
30 LET count=count*2
40 PRINT count
```

should hold no problems for you. When you enter it into your Spectrum and set it going with a RUN command the micro finds the line with the lowest number and does what that line tells it to do.

In this case the first line is line 10 and it has the Spectrum setting up a variable *count* and giving it a value of 1. This doesn't last long, however, for the next line in the sequence adds 1 to *count*. If:

```
20 LET count=count+1
```

seems a little odd, read it as "take the value that's already in *count*, add one to it and store the new value back in *count*".

The next line doubles the number it finds in *count* and, again, stores the resulting number back in *count*.

So, as the program has progressed, *count* has varied. It had a value of 1 in line 10, this became 2 after line 20 and line 30 made it 4. The last line of the program just displays the result.

Strings can vary, too

If you've grasped the concept that variables stand for things and they can vary as a program progresses, you've made great strides. Variables are the life blood of computing. Without them most programs would be restricted to fairly trivial tasks.

So far we've only come across one kind of variable, numeric variables. This is where a variable stands for a number. So in:

```
LET cost=25
```

and:

```
LET time=12
```

both *cost* and *time* are numeric variables. Useful as they are, numeric variables have their limitations. For one thing, they can't be used to store strings of characters.

Suppose you have a long program where there are several occasions when you have to tell the user to:

Press a key when ready

Of course you can use a PRINT command to do this whenever it's needed, but it involves a lot of typing. It would be nice to be able to use something like:

```
LET message="Press a key when ready"
```

to store the string in a variable and then use:

```
PRINT message
```

whenever you want. It would save a lot of typing. The trouble is you can't do it, the Spectrum won't accept:

```
LET message="Press a key when ready"
```

The problem occurs because the micro takes *message* as the name of a numeric variable, so it expects a number to be stored in it. This expectation seems reasonable. After all, the name you've tried to give the variable, *message,* obeys all the rules of a numeric variable name.

When you try to give it a string, it objects. The Spectrum thinks, quite rightly, that if you want to store a string then you should use a string variable. And a string variable name has to obey two rules. It has to:

- Consist of a single letter
- That letter must be followed directly by the dollar sign, $.

So *A$*, *n$* and *g$* are allowed as string variable names while *ABC$*, *1$* and *message* aren't.

Now you can use variables to store strings using lines like:

A$="anystring123"

and:

m$="Press a key when ready"

and recall the strings with:

PRINT a$

and:

PRINT M$

From the last couple of lines you'll see that the Spectrum makes no distinction between upper and lower case in string variable names. So, *D$* is the same as *d$*.

All together now

You'll remember that strings consist of collections of letters, punctuations, numbers and even spaces. So as well as having:

LET f$="abcde"

it's perfectly possible to have:

LET f$=" $ ()"

and:

LET f$="1234"

Don't let the last example make you think that string variables can be used for maths. It's quite possible to have:

LET a$="2"

But try doing:

PRINT a$*a$

and see the results.

The Spectrum is quite right in not allowing you to multiply two strings. After all, while in this case the characters inside the inverted commas of the LET happen to be figures, the assignments could just as easily have been:

LET a$="cat"

or:

LET a$="$%&"

And what is:

cat*cat

or:

$%&*$%&

supposed to mean?

The rule is, if you want to do maths, use numeric variables. If you want to send messages, use string variables.

And now, having said you can't do maths, we come

Strings, variables and programs

Let's put string variables to work in a program.

```
10 LET a$="Happy"
20 LET b$="Birthday"
30 LET c$="Mum"
40 PRINT a$
50 PRINT b$
60 PRINT c$
```

The first three lines assign strings to three string variables. Once the micro has obeyed these lines *a$* will stand for Happy, *b$* for Birthday and *c$* for Mum.

The next three lines tell the micro to display whatever is in these string variables. The greeting:

```
Happy
Birthday
Mum
```

appears on the screen. While simple, the program is flexible. If we want to alter the message it's easy, we just change the string variables with a line such as:

30 LET c$="Dad"

One problem with our birthday program is that it uses three string variables. Since string variable names can only go from *a$* to *z$* then we only have 26 of them. While it doesn't matter too much in this case, in longer programs you can find yourself running out of names.

A way round this is to use a string variable more than once in a program. With this technique our birthday program becomes:

```
10 LET a$="Happy"
20 PRINT a$
30 LET a$="Birthday"
40 PRINT a$
50 LET a$="Mum"
60 PRINT a$
```

which uses only one string variable *a$*. You should be able to see how *a$* stores different strings as the program progresses.

The trouble is that we lose the first two strings. By the time the program gets to line 60, *a$* contains only Mum. Its previous contents, first Happy and then Birthday, have been overwritten. And while this makes no difference in this program, in a longer one you might need to use Happy or Birthday again.

Of course, both programs could be made considerably shorter as:

```
10 LET a$="Happy birthday, mum"
20 PRINT a$
```

shows. This only uses one LET, and one string variable *a$* to hold the whole message and prints it all on one line. However, it's not as flexible, as you'll find if you try to alter it to send a birthday greeting to your dad.

to something that looks suspiciously like adding two strings. Try entering:

```
LET a$="abcd"
LET b$="efgh"
LET c$=a$+b$
```

What is held in *c$?* A quick:

```
PRINT c$
```

gives the answer:

```
abcdefgh
```

In other words *c$* is made up of the contents of the other two strings. In formal terms the two strings *a$* and *b$* have been concatenated to form a new string which has been placed in *c$*. Notice that:

```
LET c$=a$+b$
```

and:

```
LET c$=b$+a$
```

each give a different *c$*.

So by putting a plus sign between two strings we can join them together to form another string. The point to grasp, however, is that the plus sign is not our usual addition sign. Instead, it tells the Spectrum to take the strings it finds on either side of the plus and join them together. That this is not straightforward addition is clear if you compare:

```
PRINT 1+2
```

and:

```
PRINT "1"+"2"
```

In the first case the Spectrum finds numbers on either side of the plus so it does addition, giving the answer 3. In the second case it finds a string on either side of the plus. Being a bright beast, it knows that in this case all the plus means is that it is to "glue" the two strings together. It does this with the result that 12 appears on screen. This is obviously not the answer to the sum 1+2, but the joining together of two strings 1 and 2 to produce another string, 12.

Don't worry too much if you find the difference between number addition and string concatenation a bit vague. It's one of those things that when you need to understand it, you will. In most cases your Spectrum is only too willing to tell you when you've got it wrong!

Going round loops

Suppose you wanted to send a message a number of times, such as:

```
You've won!
You've won!
You've won!
```

or imitate a policeman with:

```
Hello
Hello
Hello
```

how would you do it? One way is with a program such as:

```
10 PRINT "Hello"
20 PRINT "Hello"
30 PRINT "Hello"
```

This works, but it's fairly limited. After all, what if we wanted (for reasons best known to ourselves) to have 15 Hellos? Using this method you'd have to have a 15 line program, which seems a bit silly. After all, the Spectrum is only doing the same job,

```
PRINT "Hello"
```

over and over again. Wouldn't it be easier if there was a way to tell the Spectrum to repeat the instruction for a specified number of times?

There is such a way – it's the FOR . . . NEXT loop used in the following program:

```
10 FOR n=1 TO 15
20 PRINT "Hello"
30 NEXT n
```

You'll find the keywords FOR, TO and NEXT on the F, F and N keys respectively

The FOR . . . NEXT structure is used to get the Spectrum to obey a statement a fixed number of times. When it comes across the FOR the micro knows that it will have to repeatedly obey all the lines between the FOR and the following NEXT. It follows what is known as a loop.

How many times these instructions are repeated depends on the loop control variable, the numeric variable that follows the FOR. This is given a range of values it can take. It starts at the lowest value and each time round the loop the control variable is automatically increased by one, until it is out of range and the loop stops.

In this case the loop control variable is *n* and, as it varies from 1 to 15, the loop cycles 15 times in all. What happens is that when the Spectrum encounters the line:

```
10 FOR n=1 TO 15
```

for the first time, it knows that it's about to enter a loop with control variable *n*. It gives *n* the value 1 and then goes on to obey all the following lines in order (in this case printing Hello once) until it comes to the NEXT of line 30.

Here the micro adds 1 to the value of *n*, making it 2, and the program goes back to the FOR of line 10 to see what it should do.

As line 10 has told the Spectrum that *n* is to take

LET's use INPUT instead

The programs we've used so far have all been fairly rigid. They've consisted of LET and PRINT statements and what they do is decided before they run. Take a program like:

```
10 LET first=5
20 LET result=first*2
30 PRINT result
```

This multiplies 5 by 2 to give the answer 10. And that's all it does. If we want to multiply 6 by 2 then line 10 has to be replaced by:

```
10 LET first=6
```

It would make life a lot easier if there was a way we could get the program to ask us what number we wanted to multiply when it was run. Then we could tell it 5 or 6 or whatever and the Spectrum would multiply our chosen figure by two.

There is such a statement, in the form of the keyword INPUT. Change line 10 to:

```
10 INPUT first
```

and run the program again. You'll find that nothing happens except that there's a flashing L cursor at the bottom of the screen.

What's happened is that the INPUT has told the Spectrum to wait until you enter a number at the keyboard. When you've done this it puts this value in *first* and gets on with the program. The result is that the number you typed in is multiplied by 2. You can run the program over and over again, each time multiplying a different number.

You can use INPUT to give values to string variables as you'll see if you enter:

```
INPUT a$
```

Type in the string of your choice in response to the flashing cursor and quotes at the bottom of the screen, and then use:

```
PRINT a$
```

Try using INPUT in your own programs – you'll be amazed how much better they become. However there are two things to watch out for when using it. The first is that at times you'll be wondering why the program is doing nothing when in fact it's waiting for you to enter something. You can get round this by getting the Spectrum to display a prompt to jog your memory, such as:

```
PRINT "Type in a number"
INPUT number
```

or more briefly:

```
INPUT "Type in a number", number
```

The second is that you must be careful not to get your variable types mixed up. While:

```
INPUT a$
```

will accept a number, you won't be able to do sums with a$. And if you try to give a non-numeric character such as T to:

```
INPUT number
```

the Spectrum won't like it.

values ranging from 1 to 15 and, at present, *n* is only 2, the loop goes on and another Hello appears. Only when 15 Hellos have been displayed and the NEXT has added 1 to *n* does the loop stop.

The Spectrum goes back to the FOR and finds that *n,* at 16, is greater than the limit specified. It then ignores the rest of the loop and goes on with whatever line, if any, appears after the NEXT. Adding the lines:

```
25 PRINT n
40 PRINT n
```

and running the program might make the loop process a little clearer.

There are two things to notice about this FOR . . . NEXT structure. The first is that whereas previously all our programs have consisted of one line being obeyed after another in sequence, now things are different. The FOR . . . NEXT loop doubles back on itself, repeating the same bits of program over and over.

The second thing to note is the power of the loop. See how easy it is to increase the number of Hellos just by changing the range of the control variable with lines such as:

```
10 FOR n=1 TO 100
```

or:

```
10 FOR n=1 TO 1000
```

Only one line changes but look at the difference in output.

Round and round we go

The rules for using a FOR . . . NEXT loop are fairly simple. The control variable whose range is specified after the FOR has to be a numeric variable. And unlike other numeric variables, it can only have a single letter name. So, *d* and *z* can be used as loop control variables but *a$* and *number* can't. Also the NEXT that marks the end of the lines to be repeated must be followed by that loop's control variable. So if a loop has control variable *k* then the loop ending can't be marked with a solitary:

```
NEXT
```

it has to be:

```
NEXT k
```

Bearing that in mind, you should have no problem in seeing how:

```
10 LET number=0
20 FOR p=1 TO 10
30 LET number=number+2
40 PRINT number
50 NEXT p
```

works. Now we have two lines inside the loop, sandwiched between the FOR and the NEXT. These are repeated each of the 10 times that the loop cycles, adding 2 to the value of *number* each time. Line 40 displays the result of each calculation.

The fact that the loop control variable is automatically increased by 1 each time round the loop can be very useful. We can actually use its value inside the lines that are to be repeated as:

```
10 FOR a=1 TO 5
20 PRINT a,a*a,
30 NEXT a
```

shows by giving the squares of the numbers from 1 to 5.

The next program shows how powerfully a FOR . . . NEXT loop can combine with INPUT. Only five lines long, it will print out the multiplication table for any number you want.

```
10 PRINT "Give me a number"
20 INPUT number
30 FOR b=1 TO 12
40 PRINT b*number
50 NEXT b
```

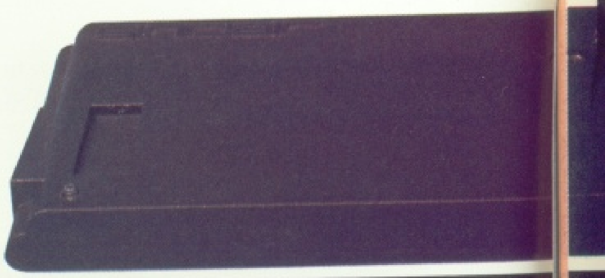
Here the loop control variable *b* goes from 1 to 12. Can you use another INPUT to make the program ask you to select the range it will vary over?

Interfaces...

The bare Spectrum can only drive dedicated printers, like the ZX printer, which work straight off the edge connector. So for full-size quality printing you'll need a suitable interface . . . read on!

Sinclair Interface I

Besides microdrive capability, Interface I gives you a serial RS232 interface suitable for use with almost any serial printer. It is easy to direct print via Interface I with Extended Basic commands, and a wide range of standard baud rates is available at the computer end to match up with the baud rate of the printer. Some software, like the Tasman range, will drive this interface



Tasman Printer Interface

This parallel interface comes from the people who supply the popular Tasword word processor and is of course compatible with that program. It comes complete with the connecting cable and driving software on cassette, and is suitable for use with a large number of inexpensive parallel printers. Extra software is available for printing in different type styles.

Problems
Driving software on cassette has to be loaded before use which can be a nuisance.

Wafadrive

The Rotronics wafadrive has both RS232 and Centronics parallel interfaces built in, so any printer can be connected. The printer can be accessed by shadow ROM extended Basic commands.

Problems
Again, non-standard connectors may mean trouble fitting a printer cable.

...and printers

Full size printers

Cheap dot matrix printers were once very likely to have a parallel interface, as it requires less electronics to build than a serial one. But electronics have now become very cheap and you are often given a choice, or may even find both fitted as standard. Daisywheel printers used to be more likely to come with a serial interface, but there are no rules anymore.

Before you buy make sure that the printer you want will work with the interface you have chosen, or which you already have as part of your system. This includes the all important connecting cable and driving software.

Printer prices have been dropping steadily for years, and will continue to do so. Don't buy the first machine you see – there are bargains to be had by shopping around. Decide on your priorities. It may be that you do not require speed so much as print quality, or vice versa. You are unlikely to get both cheaply. Dot matrix is faster than daisywheel, but even cheap daisywheel gives better quality.

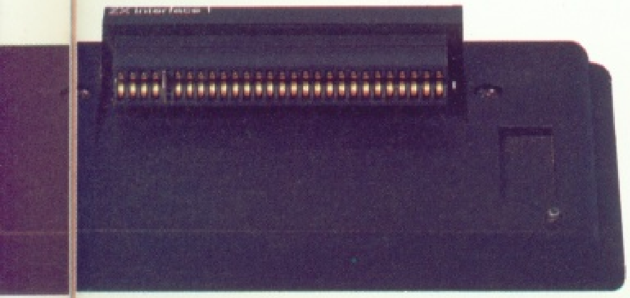
Find out how much ribbons cost for the printer you are considering – you might get a shock. Paper costs are also important if a special thermal paper is required. Cheap printers sometimes rely on expensive consumables, some even require batteries.

RS232 serial printers (various makes)

Three wires – for data, ground, and handshake – are enough to drive serial printers, so a longish run of cable is possible. If you are happy to stick with microdrive, then no extra interface need be bought. Otherwise, you had better go parallel.

Problems
The RS232 standard is seldom adhered to strictly by manufacturers, particularly in the home computer area, so connection can be tricky. Sinclair's implementation isn't helpful, with backwards pin labelling in the Interface I documentation and a non-standard plug. You also have to remember to match baud rates between computer and printer.

directly, for easy word processing or fancy print.
Problems
Interface I does not have a standard RS232 plug, so you could have trouble getting a cable to connect to the printer. However, Tasman and others do sell suitable cables. You won't be able to connect both Interface I and a disc drive to the Spectrum, so upgrading to disc will leave you without a printer interface. Most other add-on printer interfaces are parallel.



Opus Discovery

This disc drive comes with a parallel printer port suitable for use with a large number of inexpensive printers with Centronics parallel input. The Opus shadow ROM provides easy extended Basic commands to use the printer.
Problems
An edge connector instead of a standard amphenol plug at the computer end of the printer cable may be a problem, unless you can get the right cable from Opus.

Kempston E

A parallel interface with the software in eprom, this one comes with the cable to connect to the printer. Automatically responds to LPRINT, LLIST, and COPY.
Problems
There is an older model, the Kempston S, still around. This needs the driving software to be loaded from cassette. Make sure which one you are getting – the S model should be cheaper.

Euroelectonics ZX LPRINT III

Both Centronics parallel and RS232 serial printers can be driven by this interface. The software is in ROM and responds to LPRINT, LLIST and COPY commands. COPY has extra features on Epson compatibles, and an optional extra will even allow you to do four colour screen dumps on suitable printers.
Problems
Compatible with most printers, but specify which one you intend to use when ordering.

Centronics parallel printers (various makes)

The word "Centronics" is the brand name of a printer that first used this interface and established the standard. The brand names you are most likely to encounter now are Epson, Citizen, Star, Shinwa, Canon, Seikosha, Juki and Brother. Most Spectrum interfaces are parallel, so this type of printer is likely to be easier to interface than serial. Parallel interfacing does not require the matching of baud rates.

"Epson Compatibility" means that a printer will accept Epson control codes, and this is a good thing to have as many software packages are set up to drive an Epson.
Problems
Requires many more wires – usually a ribbon cable – to make the connection to the computer, so the printer and the computer cannot be very far apart.

Dedicated Spectrum printers

These need no interface, but plug straight into the Spectrum edge connector and respond to LPRINT, LLIST, and COPY commands. Narrow paper width (around four inches) and minimal print quality is OK for listings, but no good for word processing or letter writing.

Sinclair ZX printer

This was originally designed for the ZX81 and it gave the Sinclair computers the first affordable hard copy in the hobby micro world. They are no longer being made, but there are still a lot about.

Problems

Print quality is poor, with a stylus burning through the special aluminium coated paper to expose a black undercoating. It is also slow and noisy. The edge connector does not carry through the whole of the Spectrum's edge connector, as it was made for the narrower connector on the ZX81.

Floyd 40

A thermal printer with built in firmware to give extra printing options. It will print in 32 or 40 columns, double height and/or width, and inverse, and it can tidy up your listings so that words are not split across the lines.

Problems

Needs thermal paper. Costs more than ZX or Alphacom.

Alphacom 32

A ZX act-alike thermal printer giving better quality than the ZX item. Recently it has been selling for as little as £30 – a real bargain.

Problems

Thermal printers need special paper, but at least you won't have to buy ribbons for it.

UK Suppliers

Floyd 40	Floyd PO Box 76 Plymouth Devon PL1 1SQ (No phone).
Tasman Interface	Tasman Software Springfield House Hyde Terrace Leeds LS2 9LN (0532) 438301
Kempston Interface	Kempston Micro Electronics Singer Way Kempston Bedford MK42 7AW (0234) 856633
Alphacom 32	Dean Electronics (0344) 885661
Transform	Transform 24 West Oak Beckenham Kent BR3 2EZ 01-658 6350

RAMBO ™

FIRST BLOOD PART II ™

OFFICIAL COMPUTER GAME OF THE BOX-OFFICE No.1 SMASH HIT. TAKE THE ROLE OF SYLVESTER STALLONE IN ALL THE ACTION-PACKED BATTLE SCENES.

RELEASED NOW

STALLONE

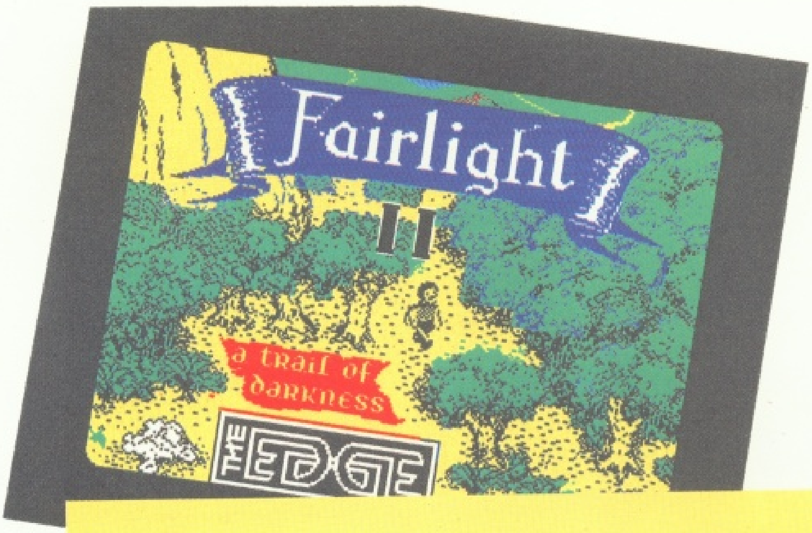
SPECTRUM 48k
7·95

ocean

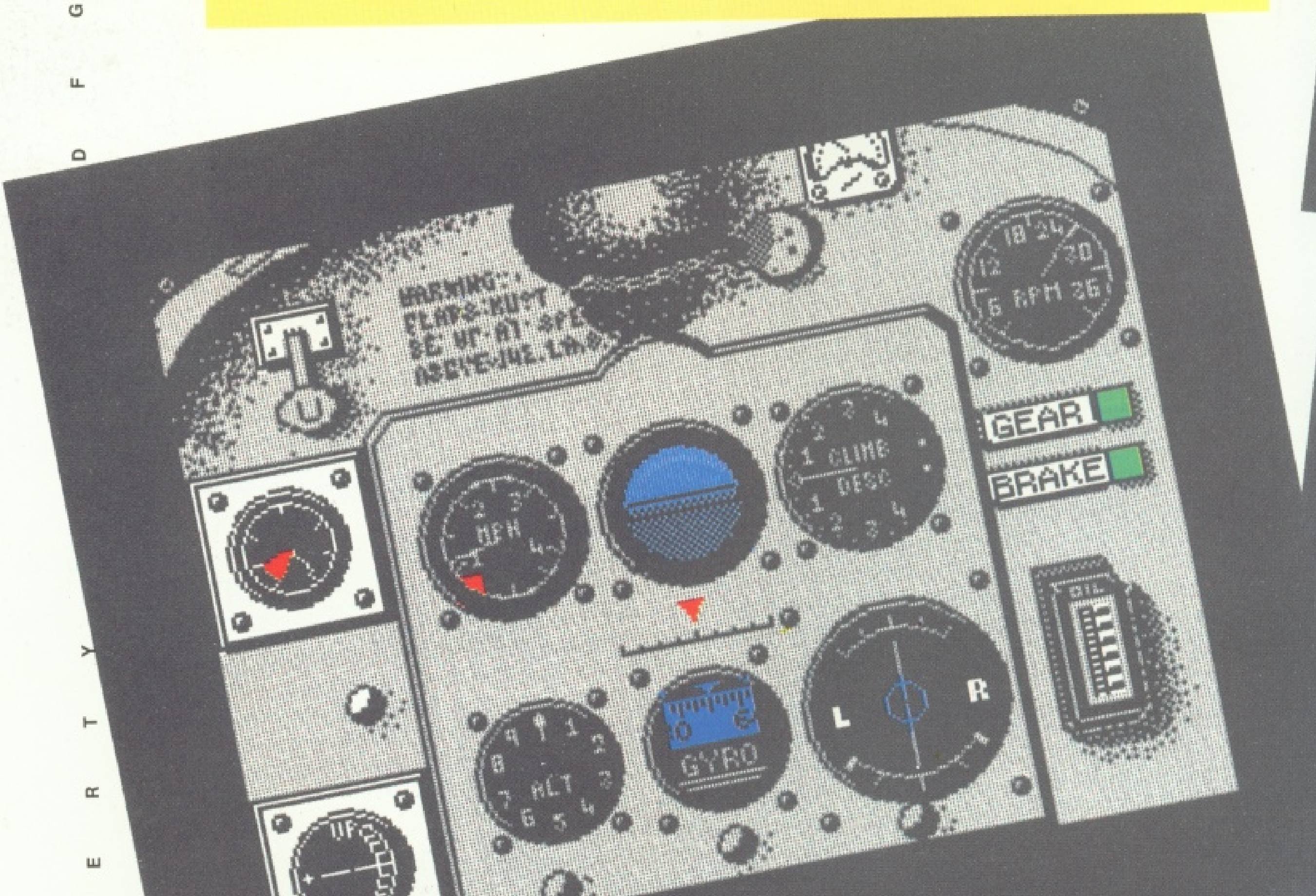
COMMODORE 64 & AMSTRAD
8·95

Ocean House · 6 Central Street · Manchester · M2 5NS · Telephone 061 832 6633 · Telex 669977
Ocean Software is available from selected branches of: Boots, WHSMITH, John Menzies, WOOLWORTH, LASKYS, Rumbelows, Greens, Spectrum Shops and all good software dealers. Trade enquiries welcome.

Q W E R T Y D F G H J K L Z X C V B N M



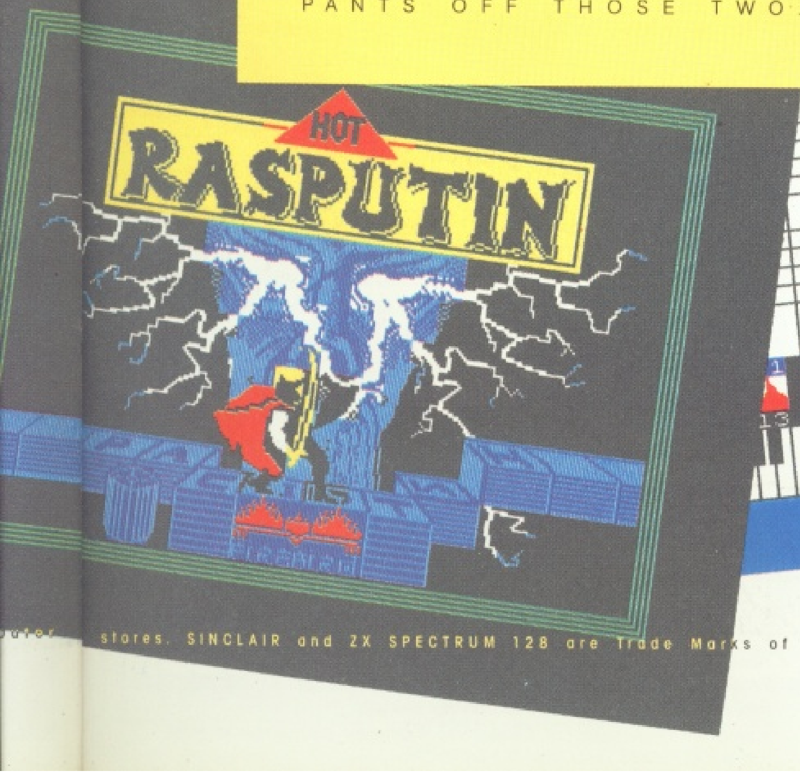
THE NEW ZX SPECTRUM 128 CAN HANDLE HARDER GAMES. WHETHER YOU CAN IS A DIFFERENT MATTER.



From Boots, Currys, Dixons, Laskys, John Lewis, Power City, W H Smith and other good compu

SLOT A CASSETTE IN AND GET READY FOR A REAL PASTING. THE NEW SPECTRUM'S 128K MEMORY WILL SHOW YOU JUST HOW TOUGH IT IS TO THRASH DALEY THOMPSON AT HIS OWN GAME. ROUGHLY THREE TIMES AS TOUGH.

OR IT'LL KEEP YOU PINNED TO THE EDGE OF YOUR SEAT PLAYING A GAME CALLED NEVER ENDING STORY. IT TAKES THREE TIMES AS LONG. AND BECAUSE WE WANT YOU TO SEE JUST HOW BRILL' ITS GRAPHICS ARE, AND HOW ITS NEW SOUND SYNTHESISER PLAYS MUSIC AND SPINE CHILLING SOUND EFFECTS, WE'RE GIVING BOTH TITLES AWAY FREE WHEN YOU BUY IT. BY THE WAY, THERE ARE THOUSANDS MORE IF EVER YOU DO BEAT THE PANTS OFF THOSE TWO.



Q W E R J K L Z X C V B N M

uter stores. SINCLAIR and ZX SPECTRUM 128 are Trade Marks of Sinclair Research Ltd.

Available directly from Kempston Micro Sales Ltd Fast Mail order service. Please send me the following

Name

Address

Postcode

1	☐	Kempston Joystick Interface	£9.95
2	☐	Centronics Interface E	£39.95
3	☐	Formula 2 Joystick	£11.95
4	☐	Kempston Pro Joystick Interface	£16.95
5	☐	Formula 1 Joystick	£16.95
6	☐	Junior Pro-Joystick	£5.95

SPECIAL COMBINATION OFFER

☐ Formula One Joystick + Joystick Interface £19.95

☐ Formula Two Joystick + Joystick Interface £17.50

Access/B'card ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

All Prices include VAT P&P. Overseas add £4 P&P

Kempston Micro Sales Ltd, Singer Way, Kempston, Bedford MK42 7AW.
Tel (0234) 856633 Telex 826078 KEMPMI G

Printers: The fine details

Now you're familiar with the general idea of printing, we look more closely at what's involved

THE ability to produce program listings or print out text must come very high on the list of priorities of those Spectrum owners who wish to progress beyond just playing other people's games.

With many micros standard interfaces are provided to allow a wide choice of printing systems to be attached. An interface, in computer jargon, is simply a means of connecting the computer to the world outside its box. In this instance we are concerned with connecting printers – but it could equally be a cassette recorder, joystick or even another computer.

With the ZX Spectrum, as with earlier Sinclair models, no such standard interface is supplied, reducing the available choice to a handful of units. Although the printer interface provided on the Spectrum is a parallel one it doesn't conform to the industry standard for such connections – often called Centronics after the firm who invented it – just as the internal character codes used by the Spectrum fail to conform to the internationally accepted Ascii codes.

These problems can be resolved in several ways, however, and certainly shouldn't present a barrier to any user determined to get letter quality output from his word processing package.

Interfacing it

The simplest solution to the problem is to obtain a special interface unit which converts the Spectrum edge connector signals into ones that a conventional printer can understand. There are many commercially available interface units that allow the ZX Spectrum to be connected to printers of one type or another. In the main these convert the signals provided on the expansion connector into the industry standard Centronics parallel interface although some do offer serial as an alternative.

There are two stages to this conversion – that of altering the electrical characteristics of the ouput from the ZX Spectrum and that of converting the character codes from Sinclair's own internal format to true Ascii.

Almost all the interfaces will be supplied with software drivers of one sort or another and in general they all work by intercepting the ZX Spectrum's printer routine vector. This is the location in memory that stores the address of the routine that controls data being sent to the printer. By changing this address to that of the new driver all the data is re-routed through the interface. On its way each of the characters passes through a code conversion routine. While the normal alphanumeric codes closely follow the standard Ascii set all the control codes, pre-defined graphics and Basic tokens must be intercepted. (See Part I, Page 28, for an example of the Spectrum's character set.)

The Basic tokens are converted into strings of Ascii letters before being listed. Because many of the currently available printers use sequences of control codes to change character fonts or to select graphics, provision must also be made to allow these to be sent without being trapped. Basic interfaces will provide the driver software on cassette which must be loaded into the Spectrum before it is used while the more sophisticated offerings contain all the necessary software in ROM inside the interface itself.

For the Spectrum owner who is either using or considering the purchase of a microdrive system an interface unit may be unnecessary as the microdrive interface itself is supplied with a serial interface which can be used to drive a printer. The alternative to some form of conversion system is to buy a printer which has been modified to operate directly with the Spectrum, just as Sinclair's own ZX Printer does. These are generally either thermal or electrostatic units which

Each character generated by the Spectrum requires one byte of storage space, each byte being made up of eight bits. In order to transfer each character all eight bits must be sent from the computer to the printer. With a parallel interface all eight can be sent together with one bit going down each of the eight wires. A serial interface, on the other hand, must send the bits down its single wire one after the other. In both the serial and parallel interface some extra wires are needed to provide control over the sending of the information. These extra wires are generally known as the handshake connections and the method with which they control the flow of information is called the protocol.

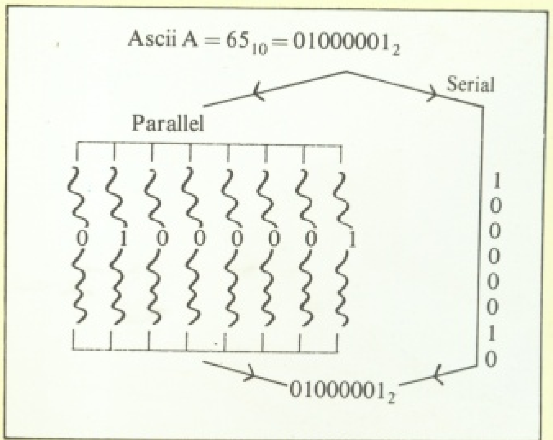


Figure 1: Parallel versus serial transmission

tend to offer a cheaper solution than buying an interface and a good quality matrix printer but the results obtained tend to be less professional.

Hitting the page

Printers are generally divided into two main categories – impact and non-impact. Each group is broken down into many different variants depending on the particular way in which the image is transferred on to the paper, such as daisy wheel, ink jet, thermal transfer and so on. Impact printers are the largest group and consist of three main sub categories – dot matrix, daisy wheel and ink jet – and, apart from the latter which actually fires its ink, these operate by pressing something hard through a ribbon in order to leave an impression on the sheet of paper.

Type	Impact	Price range	Quality
Daisywheel	Y	£400 and up	Excellent
Dot Matrix	Y	£150 and up	Good to excellent
Thermal	N	£75 to £250	Average
Electrostatic	N	£30 to £75	Poor to average
Ink-jet	Y	£300 and up	Average to good
Laser	N	£3,000 plus	Excellent

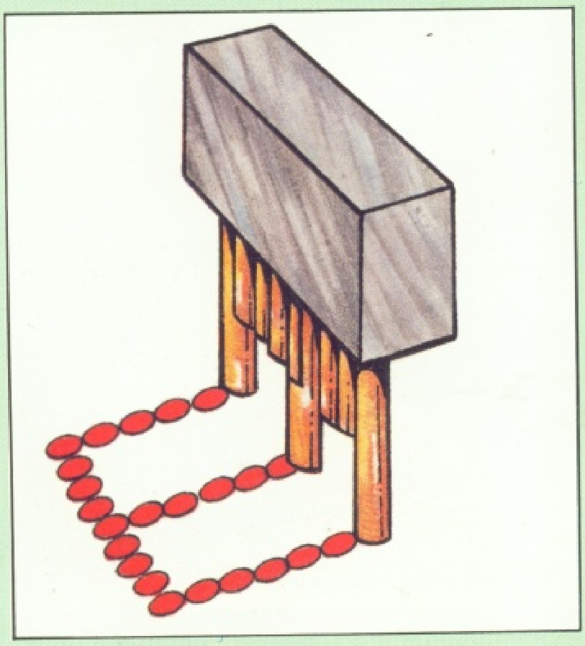
Table I

The characters to be printed are sent from the computer along either a parallel (Centronics) or serial (RS232/V24) interface. Because the characters can be transmitted much faster than they can be printed they are stored, in the printer, in a block of memory known as a buffer. This will hold at least one or more complete lines of text and sometimes as much as 16k.

When the printer is ready to output the text stored in the buffer it is read out, one character at a time, and its Ascii code used as the address of a location in a character generator ROM. This contains the pattern of dots that, when printed on paper, will make up the actual character as shown in Figure II. It makes no difference at all, electronically speaking, whether the line is printed from left to right or from right to left and most modern printers are capable of bi-directional printing.

With a daisy wheel printer the process is slightly different in that the ROM holds information relating to the position of the character on the spokes of the wheel. The character itself is made from either metal – the best quality – or plastic and is hammered through the ribbon leaving a complete impression on the paper.

Printers which create the whole character in one strike rather than from individual dots are sometimes called "character printers". Although the daisy wheel system is fairly universal there are variations such as the thimble as used in the NEC Spinwriter series. Here the wheel is mounted horizontally rather than vertically and the spokes are bent up at the ends to form a cup shape. Regardless of the methods used, however, the quality of print is superb and is referred to as "letter" or "correspondence" quality, indicating that it is indistinguishable from that produced by a professional office typewriter.



Spray it again

The largest potential upset to the dominance of the matrix printer as a low-cost output device is, however, likely to come from an even more esoteric system. Imagine microscopic drops of ink being fired towards a sheet of paper in a defined and very carefully controlled pattern.

Already well established in the industrial and commercial marketplaces these devices are beginning to make an appearance in the home market. The system works by pumping liquid ink from a reservoir to the tip of a very fine jet. Here minute droplets are charged to very high voltage before being ejected. The jet is commonly made of a piezo-electric material and this shapes the droplets by very high frequency vibrations.

As the droplet leaves the jet it is held in position by an electric field which further propels it towards the paper. The sheet of paper is not stretched over a rubber roller or platen – as it would be with an impact printer – but a sheet of metal.

This is charged to the opposite potential of that held by the droplet and attracts the ink into the paper. Very little mess occurs, about the worst that can happen is that the jet gets clogged causing the ink drops to get oversized or the reservoir is removed without being capped. The quality of printout depends mainly on the type of paper used – the more absorbent it is the more the ink soaks in and the image blurs. At their best ink-jet printers can produce an output quality several times better than that of the average matrix printer.

The latest application of the ink-jet principle is that of colour printing. As well as printing in black these devices also contain reservoirs of red, blue and yellow inks. These colours may be unfamiliar to those used to working with graphics on a TV screen but they are the painter's equivalent to red, green and blue as all other colours can be made by mixing them together.

Unlike the colour printing achieved from multi-ribbon matrix printers the results are really excellent. The penalty for this is not, somewhat surprisingly, the price but the fact that the paper used has to be slightly absorbent so the printout is of average quality.

Gently does it

An interesting variation on the liquid ink-jet printer, is the "Dry Ink" system based on the principle of spark erosion. Normally printers of this type are classified as non-impact but this version actually deposits carbon rather than burning away a layer of silvering.

The printer has several advantages over conventional printers – it is almost silent, the printhead is very

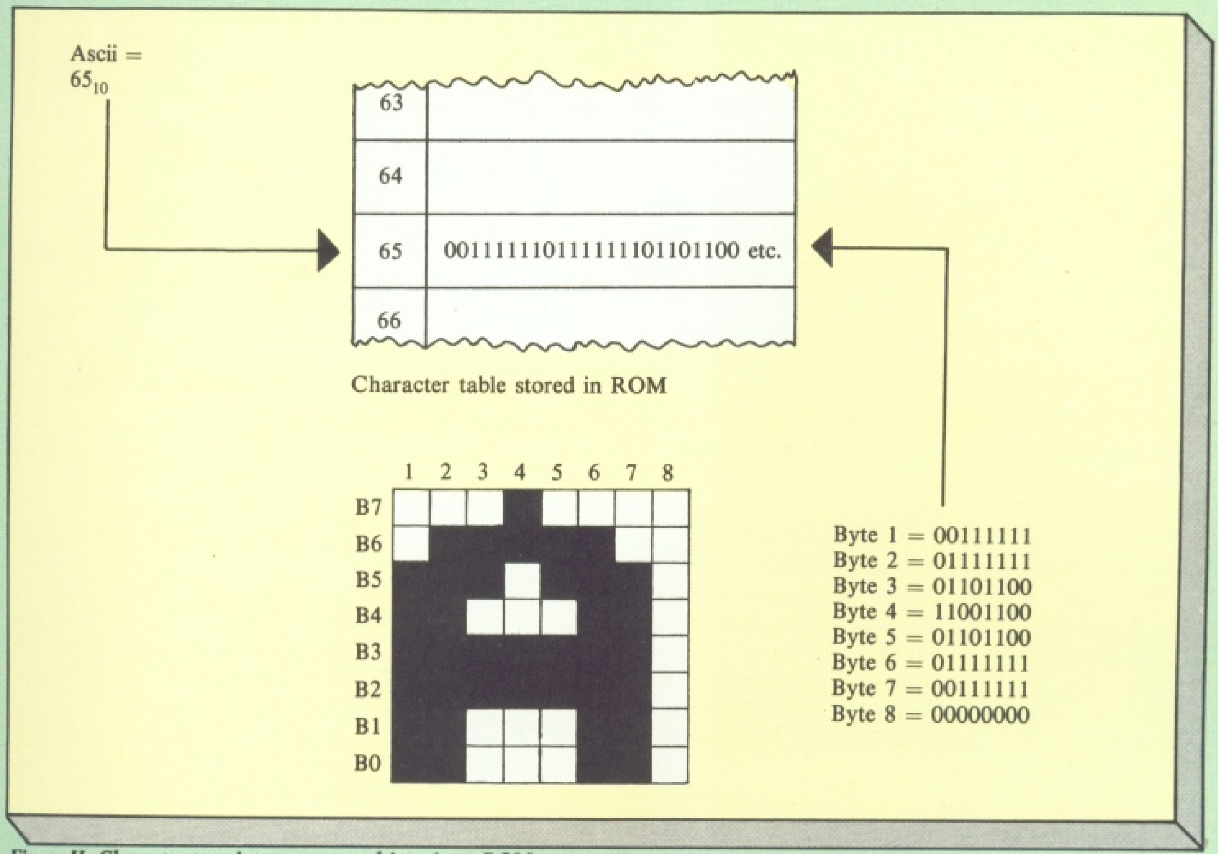


Figure II: Character to printout, as stored in printer ROM

light so powerful motors are not needed and almost any kind of paper can be used. Its drawbacks are that the printing speed is low, the head only prints one dot of each character as it passes across the paper, and the ink does tend to smudge.

Non-impact printers get their name from the fact that they don't hammer needles or shaped pieces of metal through a ribbon in order to leave their mark on a piece of paper. There are two main types – thermal and electrostatic – although there are other categories such as the laser, but this can hardly be classified as low-cost!

The thermal printer works in much the same way as an impact matrix printer except that instead of a column of needles in the printing head it has a column of heating elements. As each dot is required the respective element is rapidly heated and the minute area of paper under the element changes colour. The contrast between the dots and the background is good enough for most uses but hardly rates as high quality. The most endearing feature about these devices is their almost total silence.

In contrast, the electrostatic mechanism is moderately noisy and initially failed to gain any real support. That was until Sir Clive Sinclair adopted the system for the ZX Printer, a dedicated peripheral for the ZX 81 and ZX Spectrum. The principle is that a single-wire printing head is dragged across a specially coated paper. For every dot needed to build up a character a spark is generated by the printer which burns away the thin silver coating revealing the black backing paper. Sinclair improved the system by using two heads on a continuous belt but it still takes four passes of each head to create a row of characters.

Electronically both types of printer operate in much the same way as a conventional impact matrix printer with the pattern of dots being taken from a ROM. The quality of the printing depends on the number of dots used to create each character – typical values range from eight rows of eight dots to the so called, Near Letter Quality of 24 rows of 17 dots. The low cost thermal and electrostatic printing systems generally produce an 8 × 8 matrix printout which, while clearly legible, doesn't give true descenders on lower case letters like 'y' and 'g' and cannot support high resolution graphics.

The main drawback with both types is that they use special paper which can be expensive and is only available in rolls. This can make storage difficult as the printer doesn't produce separate pages. In the case of the thermal printer it is essential to get the correct grade of paper otherwise the image does not develop properly. Electrostatic paper is even more delicate and if handled with damp hands the image will blur as the silver coating dissolves. In both cases the best way to ensure a good, long-lasting image is to take a photocopy.

Printer type	Market leaders
Daisywheel	Brother, Juki, Qume
Dot Matrix	Epson, Star, Seikosha
Thermal	Brother, Star, Epson
Electrostatic	Alphacom
Ink jet	Epson, Hewlett Packard
Laser	Hewlett Packard, Canon

Table II

A sequence of events

Part II of this easy-to-follow series on animation techniques

Seeing is believing

BEFORE you can experiment with animation you need a method of playing back a sequence of frames so that the animation can be seen.

While there are any number of programs on the market that provide this they tend to be difficult to come to grips with or they limit the size of the frames – normally 16 × 16 pixels.

The program listed here is an aid I use extensively to check out my animation. Its main limitation is lack of speed, but it has features that more than make up for this:

- It's in Basic so it is easily customised.
- Its size is limited to 32 characters long (the entire screen width) by 10 characters high. The smaller the size of frame used, the faster the program runs, with correspondingly less flicker.
- The program makes up frames from a SCREEN$, so you can use any screen designer program on the market to create your frames. I use Grafpad and Melbourne Draw.

Listing I

```
   1 REM CORRECT CHARS
   2 POKE 23606,0
   3 POKE 23607,60
   4 STOP
   5 REM
  10 REM
  11 REM FRAME MAKER
  12 REM
  13 REM
  14 REM
  15 REM
  16 REM INITIALISE
  17 REM
  18 REM
  20 CLEAR 39999: LET ADDRESS=40000
  25 LET VERT=0: LET HOR=0: LET FRAME
=1
  30 LET FRAMES=8: REM NUMBER OF FRAM
ES TO BE CONSTRUCTED
  35 DIM F(FRAMES): DIM L(FRAMES): DI
M I(FRAMES)
  40 DIM V(FRAMES): DIM H(FRAMES)
  50 LET LENGTH=0: LET HIGHT=0
  60 LOAD "walk1"SCREEN$
  70 GO TO 2000
  80 REM
  90 REM
 100 REM FIND A FRAME
 101 REM
 110 LET A=PEEK (22528+(VERT*32)+HOR)
 117 IF (22528+(VERT*32)+HOR)>22528+(
24*32) THEN  STOP
 120 IF A=40 OR A=104 THEN  LET f(fra
me)=ADDRESS: LET V(FRAME)=VERT: LET H
(FRAME)=HOR: RETURN
 130 LET HOR=HOR+1
 140 IF HOR>31 THEN  LET HOR=0: LET V
ERT=VERT+1
 150 GO TO 110
 190 REM
 195 REM
 200 REM FIND LENGTH
 201 REM
 205 LET LENGTH=0
 210 LET B=PEEK (22528+(VERT*32)+HOR+
LENGTH)
 220 IF HOR+LENGTH=31 THEN  RETURN
 230 IF B<>A THEN  RETURN
 240 LET LENGTH=LENGTH+1
 250 GO TO 210
 290 REM
 295 REM
 300 REM FIND HIEGHT
 301 REM
 305 LET HIGHT=0
 310 LET B=PEEK (22528+((VERT+HIGHT)*
32)+HOR)
 330 IF B<>A THEN  RETURN
 340 LET HIGHT=HIGHT+1
 350 GO TO 310
 390 REM
 395 REM
 400 REM "CHECK FRAME HAS NOT
          BEEN DONE BEFORE"
 401 REM
 410 LET DONE=0: IF VERT=0 THEN  RETU
RN
 425 IF PEEK (22528+((VERT-1)*32)+HOR
)=PEEK (22528+(VERT*32)+HOR) THEN  LE
T DONE=1
 430 RETURN
 490 REM
 495 REM
 500 REM FIND RELEVANT BIT
         MAPPED ADDRESS FOR
         COORDINATES
 501 REM
 510 IF M<8 THEN  LET START=(16384+(M
*32)+N): RETURN
 520 IF M<16 THEN  LET START=((16384+
(8*32*8))+((M-8)*32)+N): RETURN
 530 LET START=((16384+(16*32*8))+((M
-16)*32)+N)
 540 RETURN
 990 REM
 995 REM
1000 REM READ CHARACTER
    DEFINITIONS
1001 REM
1010 FOR M=VERT TO (VERT+(HIGHT-1))
1020 FOR N=HOR TO (HOR+(LENGTH-1))
1030 GO SUB 500
1040 FOR X=0 TO 7
1050 POKE ADDRESS,PEEK (START+(32*X*8
))
1055 POKE 65368+X,PEEK ADDRESS
1060 LET ADDRESS=ADDRESS+1
1070 NEXT X
1075 PAPER 2: INK 8: OVER 1: PRINT AT
 M,N;"?";
1076 PAPER 5: INK 8: OVER 0: PRINT AT
 M,N;"?";
1078 PAPER 7: INK 0
1080 NEXT N
1090 NEXT M
1100 RETURN
2000 REM
```

The program

Key in Listing I. Once it is entered and saved, you should make up a SCREEN$ to test the program.

The walk illustrated in this article is a good test (see section on using the walking figure). Almost anything that varies over several frames – a flickering fire, a bouncing ball and so on – will do.

Whatever you use the frames should be saved as a SCREEN$ and the following criteria observed during its preparation:

1. The attribute setting for the whole screen should be set to:

INK 0:PAPER 7:BRIGHT 0

2. Frames should be arranged in order left to right. If there is not enough room they can form a second or even third row down. Do not use the bottom two rows of the screen, which are reserved for error messages.
3. All frames should have their colour attributes set to:

INK 0:PAPER 5

4. The frames should not touch each other and should have the same dimensions.

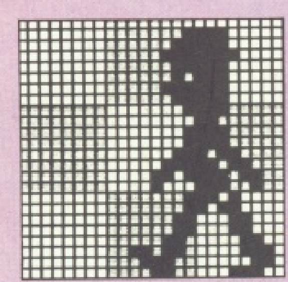
If all these criteria are satisfied, then all the frames will be copied, in order, into an area of memory – for use in the second half of the program – when the program is run.

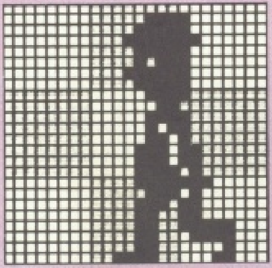
Before running the program however it is necessary to enter some variables into the listing.

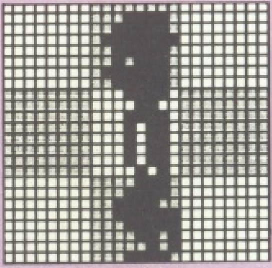
Line 30	Number of frames to be constructed.
Line 60	Name of SCREEN$ to be loaded in (Note that if you're using a tape system line 60 will need the microdrive specifiers removing).
Line 3015	Make B$= the sequence in which frames are to appear. The sequence is repeated.
Lines 3020 to 3027	More variables to be set (see REMs in the listing for details).

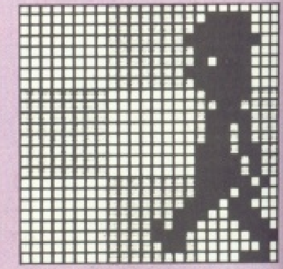
The program is run by entering RUN 10

```
2001 REM
2002 REM MAIN LOOP
2003 REM
2004 REM
2005 GO SUB 100
2010 GO SUB 400
2020 IF DONE=1 THEN  GO SUB 130: GO T
O 2010
2030 GO SUB 200
2040 GO SUB 300
2050 LET H(FRAME)=HOR+LENGTH
2060 GO SUB 1000
2065 LET I(FRAME)=HIGHT: LET L(FRAME)
=LENGTH
2070 LET HOR=HOR+LENGTH: LET VERT=V(F
RAME): LET FRAME=FRAME+1
2075 IF FRAME=(FRAMES+1) THEN  CLS :
GO TO 3000
2080 GO TO 2000
2095 REM
3000 REM
3001 REM
3005 REM SECOND HALF OF PROG
3006 REM
3010 LET A$="": FOR N=32 TO 127: LET
A$=A$+CHR$ N: NEXT N
3015 LET B$="12345678": REM SEQUENCE
CODES
3020 LET SEQ=8: REM NUMBER OF FRAMES
IN SEQUENCE
3025 LET test=3: REM HIEGHT OF FRAME
3026 LET l=3: REM WIDTH OF FRAME IN C
HARACTERS
3027 LET VERT=0: LET HOR=10: REM POSI
TION ON SCREEN
3030 GO TO 3300
3190 REM
3191 REM
3192 REM MAIN PRINTING ROUTINE
3193 REM
3194 REM
3200 REM  POKE 23606,LO: POKE 23607,H
I
3205 PRINT AT VERT,HOR;a$(1 TO l)
3206 IF test=1 THEN  RETURN
3210 PRINT AT VERT+1,HOR;a$(l+1 TO 2*
l)
3211 IF test=2 THEN  RETURN
3215 PRINT AT VERT+2,HOR;a$(2*l+1 TO
3*l)
3216 IF test=3 THEN  RETURN
3220 PRINT AT VERT+3,HOR;a$(3*l+1 TO
4*l)
3221 IF test=4 THEN  RETURN
3225 PRINT AT VERT+4,HOR;a$(4*l+1 TO
5*l)
3226 IF test=5 THEN  RETURN
3230 PRINT AT VERT+5,HOR;a$(5*l+1 TO
6*l)
3231 IF test=6 THEN  RETURN
3235 PRINT AT VERT+6,HOR;a$(6*l+1 TO
7*l)
3236 IF test=7 THEN  RETURN
3240 PRINT AT VERT+7,HOR;a$(7*l+1 TO
8*l)
3241 IF test=8 THEN  RETURN
3245 PRINT AT VERT+8,HOR;a$(8*l+1 TO
9*l)
3246 IF test=9 THEN  RETURN
3250 PRINT AT VERT+9,HOR;a$(9*l+1 TO
10*l)
3251 IF test=10 THEN  RETURN
3260 RETURN
3291 REM
3300 REM
3301 REM
3305 REM MAIN LOOP
3306 REM
3307 REM
3310 GO SUB 6000
3315 FOR n=1 TO SEQ: POKE 23606,x(n):
 POKE 23607,w(n)
3320 GO SUB 3200: NEXT n
3325 IF INKEY$<>"" THEN  GO TO 1
3330 GO TO 3315
6000 REM
6001 REM
6002 REM WORK OUT ARRAY CONTENTS
         FOR CHARS
6003 REM
6004 REM
6005 DIM w(frame): DIM x(frame): DIM
y(frame): DIM z(frame)
6010 FOR m=1 TO SEQ
6020 LET n=VAL b$(m)
6030 GO SUB 9000: LET w(m)=HI: LET x(
m)=LO: NEXT m
6040 RETURN
9000 REM WORK OUT HI/LO POKES
9001 REM
9010 LET ADD=F(N)-256
9020 LET HI=INT (ADD/256)
9030 LET LO=ADD-(HI*256)
9040 RETURN
9900 STOP
9990 REM
9995 SAVE *"M";1;"FRAMEMAKR"
9996 STOP
9998 SAVE "F.MAKER.1"
```


Frame 1


Frame 2


Frame 3


Frame 4

Using the walking character

There are eight frames each measuring three by three characters – 24 × 24 pixels. The head design is not important so you may customise that part of the figure, but everything below is critical to give the finished illusion.

The figure's placing in the 24 × 24 grid is very important. You will see that it moves leftwards two pixels between each frame and that on frames one and four it is moved a whole character square to the right.

If you make up a SCREEN$ containing these called "walk1", and run the program, the figure will appear to jerk backwards each few frames when it gets to the second half of the program. To get round this just add the following lines:

```
3316 IF n=4 THEN LET HOR=HOR-1
3317 IF n=4 AND HOR=0 THEN LET HOR=28
3321 LET HOR=HOR-1
```

The figure will now repeatedly march from right to left.

Method of printing used

The frames are printed to the screen using the method described last time as self matting. This is the easiest and quickest way in Basic, but allows no use of background.

You might like to try adapting the program so that it prints using OVER 1. To do this you must remember to blot out the old frame before printing the new.

Animated SCREEN$

One approach to experimenting with animation which is ideally suited to this program is that of an animated SCREEN$. A good example of this is Durrel Software's Combat Lynx.

The loading screen shows a picture very similar to the cassette inlay, a helicopter in flight. When the game has loaded the screen suddenly comes to life, the rotor blades spinning realistically. The effect is spectacular and can make a stunning demonstration.

It is of little importance whether you start with a complete SCREEN$ and then animate a small section, or do it the other way round, that is make up your animated frames and then compose a picture around them. Whichever way you decide to go start with something small, a bird singing in a tree for example. You will be surprised at how little needs to be animated in order to bring a whole picture to life.

Walking theory

Looking carefully at the walking frames you will notice that the silhouette of frames 1 and 5, frames 2 and 6 or frames 4 and 8 would be identical. The only variation is the highlighting on the arm or leg to show which is in front of which. You may also notice that frames 3 and 7 are identical.

At this resolution and in this style it's virtually impossible to show highlighting on this frame. So if computer memory is limited you could save the space used by frame 7 by using frame 3 whenever frame 7 was needed. Frames 3 and 7 are known as a crossover, the leg bent and crossing the other in motion. This position is extremely important in giving character to a walk.

If you make the figure much smaller than this it becomes impossible to show highlights on the other frames and you can get away with an eight-phase walk using only four frames.

Similarly if the figure is enlarged it becomes increasingly easy to show detail on limbs which cover others and eight frames per cycle will always be needed.

● ***Next time we'll look at the actual art of animation with illustrations from waterfalls to flying saucers to facial expressions.***

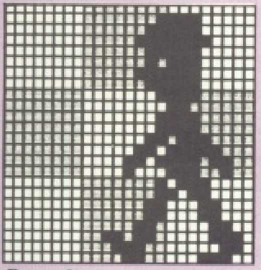
Using BIN to input graphics

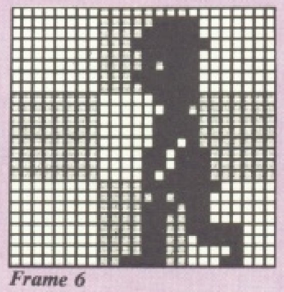
It is possible to use the BIN function on the Spectrum to input data direct to memory.

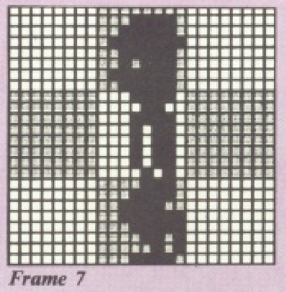
If you choose this method for lack of a screen drawing package, remember to take the top left hand character of the frame first and then from that character poke into successive addresses each of the eight bytes of that character in binary format. Then do the same for each character, working from left to right and from top to bottom of the frame.

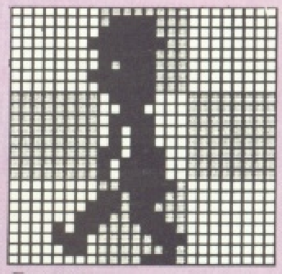
The frames are stored in memory starting at Address 40000. All the frame addresses are to be found in the array F(n) – where n is the frame number.

The frames can of course be saved separately but you will need to make a note of their addresses if you wish to use them in a different program.


Frame 5


Frame 6


Frame 7


Frame 8

Finding things to animate

Eyes opening, closing or looking from side to side. Mouths talking, smiling, frowning. Water, birds, or the ubiquitous throbbing of a space invader.

These are all grist to your mill.

Your sequence can contain up to 10 different frames. And since B$ (line 3015) actually codes the order in which they are printed and can be of any length, you can use B$ to print a complicated frame sequence. For instance suppose you wanted to animate a man's face. You could have:

1=frame of man with eyes open, mouth closed.
2=frame of man with eyes closed, mouth closed.
3=frame of man with mouth open, eyes open.

And if B$="11121113131313131111112111" what would the result be?

In particular try to program sequences with several frames – anybody can do animation between two frames, so don't run with the rest. Experiment with whatever you like, but I would recommend that you don't move too much at a time.

There are three small differences between the Mexican face shown in Figure I and Figure II. Any one of these will give the illusion of animation but if two or all three happen at exactly the same time the effect is nowhere near as good.

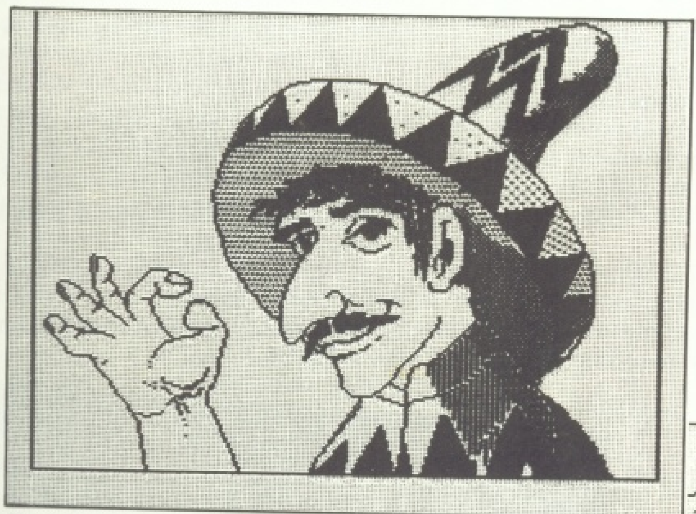


Figure I

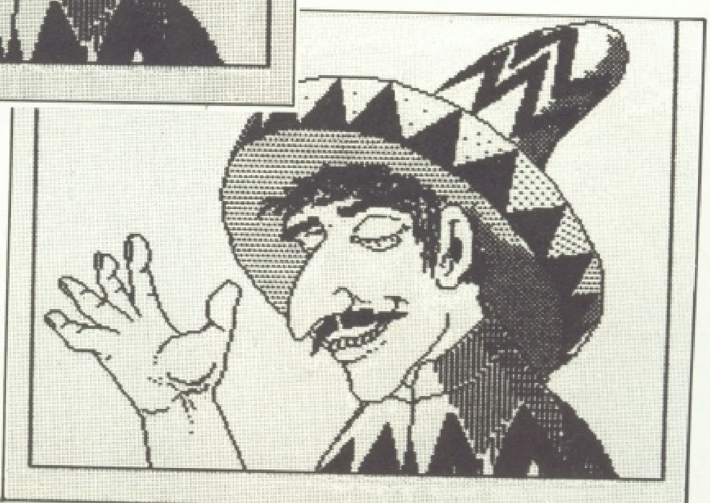


Figure II

Upgrading the keyboard

We outline the qualities to look for in a replacement keyboard

IF you only use your Spectrum for entertainment, then the keyboard it was supplied with is perfectly adequate. Employing it for serious work though – especially word processing – will quickly show up its drawbacks and convince you that £30 or £40 is not really too much money to spend for a keyboard you can actually type on in comfort.

There are lots of replacement keyboards made for the Spectrum and any of them will be an improvement on the one you got with the computer. But it is still worth having a good shop around before you buy, and I mean you should actually look in the shops, not just survey the advertisements.

Computing hardware is notoriously difficult to produce on schedule, which means that new products are often not actually available when the advertising appears. This doesn't apply just to keyboards of course, but with keyboards especially because it is difficult to tell from the adverts what the manufacturer has done about printing the complicated legends (with all those keywords) on the keys. One keyboard used to come with black stick-on legends that you had to apply yourself – to black keys!

Adverts have been known to show misleadingly well marked keytops in their artwork that turned out not to exist in real life. This is not wilful deception, but is due to the need to prepare advertising material months in advance of the scheduled release of the product. The problems with manufacture naturally turn up after the advert has been booked.

Besides this important consideration there is the subtler matter of the "feel" of the keys, which can be very important to touch typists, and which is much of the reason for buying a keyboard upgrade in the first place. This is something you can obviously only judge in a hands-on demonstration.

All of these keyboards come with more keys than the original rubber Sinclair keyboard, and some of them have many more than even the Spectrum Plus. Some of these extra keys provide single press access to punctuation or keywords that require the use of shifts on the standard keyboard. A full size spacebar is another obvious improvement.

Personally, I don't like a lot of extra keys – especially not calculator style numeric pads – because I have become used to the layout of the standard keyboard, but you may find this sort of thing very useful. The choice of keys and their layout may make a keyboard better suited to programming than word processing, or vice versa.

The Hard Part

Fitting a keyboard upgrade usually will not require the use of a soldering iron but you will have to open the computer up, detach the old keyboard, and remove the circuit board from the case. Then you have to install the circuit board in the new case and fit the connections from the new keyboard into the sockets on the Spectrum circuit board. Sometimes there is something extra like a reset button or an on/off switch that needs

Saga

Saga supply three different keyboards, with a common emphasis on extra keys. The top-of-range "Elite" has 87 keys and costs as much as the Transform. The 67 key "Emperor" could be another one to look at. Not only is it cheaper, but it has the keywords printed on the keys while the "Elite" has them on a separate chart at the top of the case.

Saga's 52 key "Profile" is a redesign of the "Lo-Profile" keyboard that used to be marketed by a different company, but which has now been taken over by Saga. Low it may be, but it is also very wide and long, covering much more desk space than it ought.

dk'tronics

The first keyboard from dk'tronics (not counting the ZX81 version!) simply reproduced the original Spectrum's rubber keyboard layout with proper key switches. It didn't even have a space bar. That was long ago, and the current model has spacebar and numeric pad as well as a few extra keys. This is one of the cheapest keyboards you can find for the Spectrum. But inexpensive needn't mean low quality: this is a sturdy item with well marked keys and certainly worth thinking about.

soldering, but even then it will probably be optional, and you can just leave the feature unconnected.

Common sense and the ability to follow the instructions is all you need for a successful installation. It is the digital wizards who get into trouble with these things, because they know all about electronics and therefore feel that they don't need to read the instruction sheet.

If you don't like the idea of doing the work yourself, the keyboard suppliers may offer a fitting service, or you can ask a specialist repairer to do it. It will probably cost extra either way. The Sinclair warranty will be invalidated when the computer's case is opened, but a reputable repair centre should give you a warranty of their own.

Transform

Very upmarket, this keyboard is enclosed in a large steel case and has room for Interface I and the Spectrum power supply within it. It is quite expensive at about double the price of the cheaper keyboards. An early version did not have separate cursor keys, but they are now included in the 65 key total.

LMT 68FX2

A medium priced keyboard with – as you can see from the name – 68 keys. LMT are a fairly new supplier, and the dreaded "Please allow 28 days for delivery" appears at the bottom of their adverts. LMT also make a disc interface and do a special price if you buy both keyboard and interface together. Worth considering. This keyboard is also available from Fox Electronics.

Sinclair Plus

This is only an upgrade if you have the original rubber keyboard, and even then I don't recommend it. You get hard plastic keys concealing the same old membrane contacts instead of proper key switches. However, the keys are well marked and you do get extra keys for single press cursors, and so on. Rather expensive for what you get.

Replacement parts

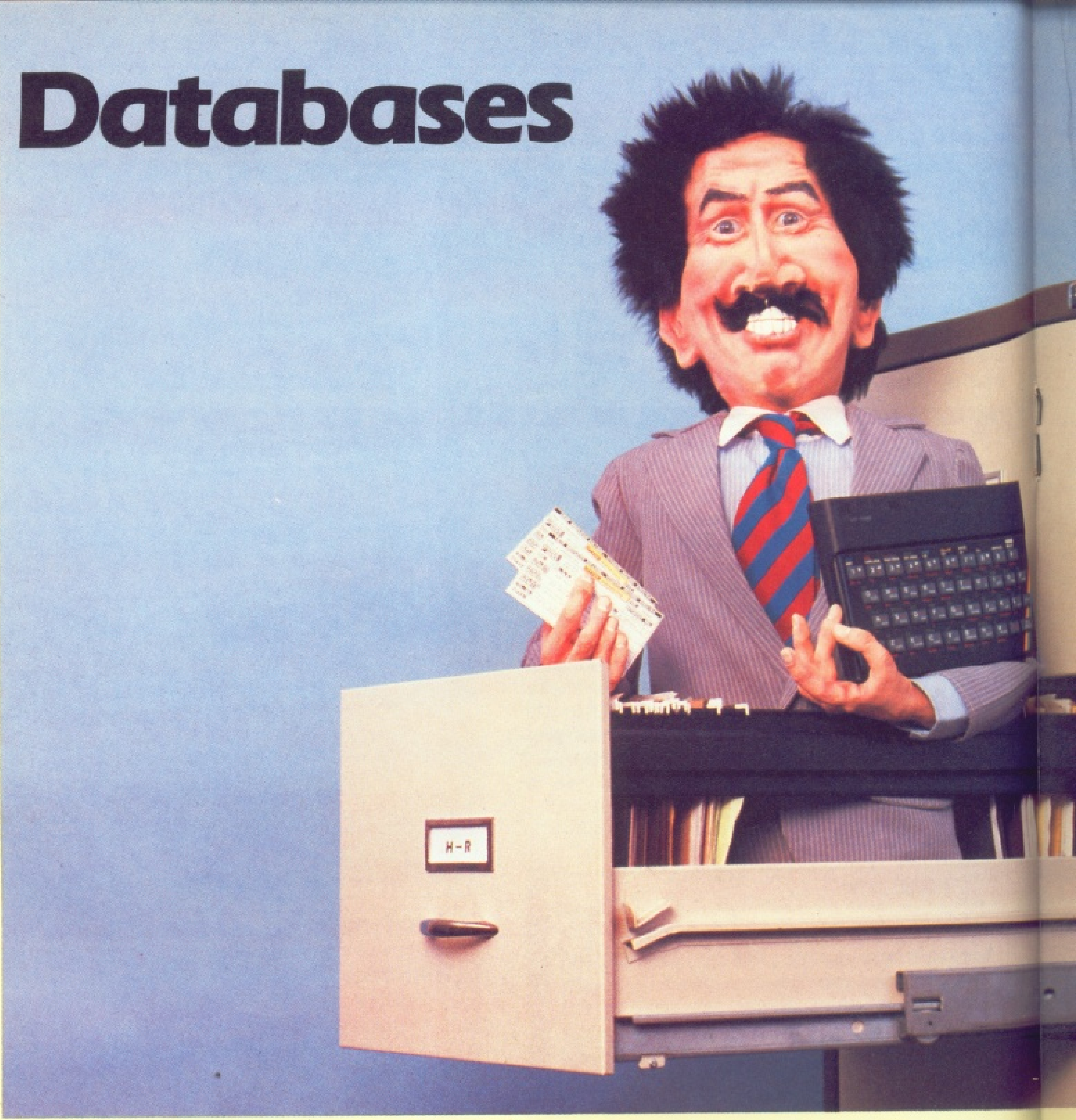
You can get replacement membranes, rubber mats, and metal templates for the original Spectrum keyboard, and similar bits for the Spectrum Plus. This is hardly an upgrade, but it is the cheapest solution if it is just a question of a worn out membrane, and if you don't really need a better keyboard. This route is for the more competent DIY enthusiast as the membranes are fiddly and fragile, and disassembling the Spectrum involves prying glued parts asunder.

If you can hack it, a membrane should cost you only about £3. You will have to telephone around the repairers to find a supplier though, as Sinclair doesn't like these parts to be advertised (try Video Vaults). Alternatively, some repairers will do the job for you for £10 or £12.

Keyboard suppliers

dk'tronics	dk'tronics Longs Industrial Estate Gorleston Great Yarmouth Norfolk NR31 6BE (0493) 602926
Fox	Fox Electronics Fox House 35 Martham Road Hemsby Great Yarmouth Norfolk NR29 4NQ (0493) 732420
LMT	LMT Computers South Street Commercial Centre Bishop's Stortford Herts CM23 3AL (0279) 506801/54437
Video Vault	Video Vault 140 High Street West Glossop Derbyshire (04574) 66555/67761
Saga	Saga Systems 2 Eve Road Woking Surrey GU21 4JT (04862) 22977
Transform	Transform 24 West Oak Beckenham Kent BR3 2EZ 01-658 6350

Databases



An in-depth look at these computerised filing systems

JUST what is a database? The word is part of the modern computer world's jargon. You see it somewhere in every computer magazine. Modern hi-tec salesmen try to sell them every which way. This one is the best, this one has the largest, best, most versatile input – just the thing for stock control, sales contacts, subscription lists or promotional analysis.

You would think that they were something new, hi-tech or just off the leading edge of modern technology. But it's easy to forget that the Romans had them, and the Ancient Greeks before them – databases are really only a set of records. Certainly early ones were carved on tablets of stone or crudely inked onto papyrus but they were still an early form of database.

So just what can your Spectrum do for you with one of the various databases available, installed and up and running?

With games software it is often possible to get a good idea of whether a game is to your liking by asking to see it demonstrated in your local computer shop. With a database program this is not always so simple. Not only are these programs somewhat rarer than the top ten games but those that are available over the counter are often not so easy to judge on the spot.

As with any utility program that you are going to use as a tool rather than as an entertainment package, you should have some idea of what you will want to do with it before you actually dip into your pocket.

As we hinted earlier, a database is a system of filing records just like a box file or filing cabinet. Each individual entry can be looked on as a separate card containing all the information that you want to record about that person or thing. A simple address list may just have the following information:

NAME:
ADDRESS:
TELEPHONE NUMBER:

There is nothing wrong with this approach, but it's a waste of time and money to put such a list on to your computer database just for reference purposes. An address book would be quicker and a darn sight more simple to access, maintain and update. But if you have

On file

FILING card systems are still available and are often quicker to use than their computer controlled counterparts – *providing you only need a simple analysis of what they contain.* Just think for a moment about those small filing boxes available from a well known High Street stationers, who uses them and what for.

The fairly obvious uses are address lists, club records and possibly Mum's store of recipes. The address list normally works well because the records are usually kept in surname alphabetical order, so looking up Joe Brown's address, or Zeri Sadwerski's, is a fairly easy matter.

The club list or the recipes could prove to be a different matter. Both of these databases have more than one way in which they might be referenced. Not only will they need to be in some sort of alphabetical order – Andrews, Askew, Barnes (or goulash, haggis, kebabs) – but also in the case of the club you may need to list them by age, handicap or even if they have paid their club fees.

The recipes also have various different attributes that will make sorting them difficult for some needs. Perhaps you need to know which dishes could be cooked at the same oven settings or perhaps a friend coming to dinner is a vegetarian, so want to know which meals do not use meat.

Even our simple address list can pose problems on occasions: Perhaps like me you can remember where someone lives but seem to have forgotten his name. They also seem to have an almost intelligent knack of falling to the floor and scattering all those neatly written cards to the four corners of the room. Then there is always the card you are sure is there but seems to have been put back in the wrong place.

Perhaps some of those salesmen have got the right idea after all – you do need a computerised database.

a reason for wanting to write to all these people, Christmas cards for instance, then it becomes more attractive. Let your printer type out all those addresses on to those very convenient sticky labels. Perhaps you do not want to send Christmas cards to everyone on your list. Better and better, tell the computer which ones to miss out.

This is exactly the sort of thing a good database program *can* do. It can look at your records and arrange them in an order and a specific format, to suit *you.* Instead of just recording name, address and telephone number, you could also make a note to send a Christmas card.

On each of these records, certain bits of information are classed as key fields. These are used to facilitate the sorting of your records. So, looking again at our box of addresses, it may now look like this:

NAME:
ADDRESS:
TOWN:
COUNTY:
TELEPHONE NUMBER:
INTERESTS:

This could give us six fields to examine during a search operation. Something like this format would suit a salesperson who had to travel around the country visiting clients. By extracting all the entries for a given TOWN or COUNTY, no client gets missed and in addition, the INTERESTS field will supply information of either personal or business concern that will help that salesperson do his job properly.

A number of database programs have been published for the Spectrum. Some are very versatile and can be tailored to suit your own needs exactly. Others are specialist programs such as address books or club records that are more restricted in what they will do but in theory should be a bit easier to use because of their more single-minded approach.

Still going strong

One of the earliest database programs for the Spectrum came out shortly after the launch of the machine itself and is a Sinclair program from Psion, Vu-File. It will run on a 16k Spectrum, but as with other databases the number of records possible with the restricted memory is severely limited. For any serious use a 48k machine is a must. Many programs have a sample database included within the package, either loaded with the main program or as a separate data file. These are always worth studying closely as they give you the bones of a database to play around with while you are learning how to access the various functions offered.

Vu-File has in addition to a blank database a world gazetteer (Europe-only in the 16k version), which lists 151 countries together with their capital, languages, currency, gross national product and land area. This not only makes interesting reading but also shows the user how quick the sorting routine is. Vu-File will search for a specified character string either in a specified field or globally throughout the entire database. Either way it is darn fast.

On loading the program you are presented with the main menu, which offers you six options:

1. Enter Vu-File
2. Set record layout
3. Set printer layout
4. Save data file
5. Load a new data file
6. Erase current file

With a blank database you first have to define the record layout. This is a very straightforward operation, with clear on-screen instructions. Record "cards" are full screen width and 20 lines deep. Having completed

A cut-price database

THE packages reviewed in detail here are standalone databases. But if you have the Mini Office suite of programs you already have a database that is perfectly adequate for most home computing purposes.

This is one of four different modules that make up Mini Office. The other self-contained programs it contains are a word processor, a spreadsheet and a graphics creator.

Mini Office costs £5.95 for all four modules – which means this is the cheapest database you're ever likely to find.

this you drop straight into the enter mode and have to enter your first set of records. Once this is done you may then either save them as a data file or access any other mode available.

You may step forwards or backwards through your records, already sorted alphabetically by the first field specified. Records may be added to, altered or deleted, search fields may be specified and the order of records displayed. Records may be printed either as a single page (copy) or as a complete set (using your search/select parameters if required) but including or excluding those fields you specify.

In fact, although Vu-File is one of the oldest databases available for the Spectrum, it still has a lot going for it. It lacks the more sophisticated printer commands of more modern programs and neither does it have the microdrive capabilities that some others have. But it is still a fairly versatile operation and very easy to use.

Modern magic

From ancient to modern, we now look at one of the latest versions of Campbell Systems' Masterfile. This is a versatile database program that can be tailored to your own requirements. It is fast in operation and can hold a very creditable number of records. It also supplies full implementation instructions for using microdrives.

Masterfile will work with the ZX printer – and other dot-matrix printers via several common interfaces – using the Spectrum COPY mode. As a compatible utility Campbell Systems also have MF-Print which, in conjunction with most printer interfaces for the Spectrum, allows the printing of full width reports on any printer that can be connected to the Spectrum.

On loading Masterfile, the first program is the equivalent of many business computer read-me files. It brings you up-to-date information on the program and any modifications that have been made. A good idea this, and it saves costs on updating the manual. This manual is 20 pages (150mm × 210mm) of fairly small print and needs careful reading to get the most out of the program.

Once you have the database proper in your Spectrum you will find records already entered for your perusal. Make use of them to learn what Masterfile can do. Note the different forms of report display: Five are shown but up to 36 may be specified. They all use data from the same record cards, but may be sorted or selected in whichever order you specify – quite impressive.

You may have up to 26 data fields for each entry, each coded A-Z (your choice) and each field may hold up to 128 characters. When data is shown in the display mode rudimentary word processing is performed so that the text is displayed in a very clean and readable manner with no word breaks at the end of lines. This makes entry that much easier, as you do not have to worry how the final result will look.

In addition to the standard Spectrum character set giving 32 columns there is also the option of specifying 42 or 51 column micro-print. Although still limited to 128 characters per data field, this does mean that more information can be squeezed into each individual report.

The different report formats give Masterfile a very great degree of flexibility. You can have your records displayed or printed and sorted in alphabetical or numeric order using any one of the 26 data fields as the master. Depending upon how much information you wish to display from each record, one or many records may be shown at any one time on the screen. You may then step through these "pages" either forwards or backwards by one page, or forwards by up to nine at a time.

The search facility will find a specified string of characters within a specified data field and present these records for display or printing. These selected records may be discarded and the remainder then become your selected batch for further searches. This is useful to eliminate, but not delete, records with certain attributes. If required, you may search selected records to further narrow down your choice.

Numeric searches also allow comparisons such as equal to, unequal to, greater than and less than. Masterfile can also compute and display the total and average of numeric items of selected records.

All these operations are menu-driven and are easy to access, but Masterfile is a versatile program and it will take a new user a little while to learn how to get the best out of it. There are even instructions on how to delve into the small Basic part of the program to tailor certain functions to your own use.

What to look for in a database	Vu-File	Masterfile	Address Manager	Mini Office
Data fields: 1. number 2. addition/deletion	30 ****	26 ****	8 fixed	14 fixed
Maximum field size	32	128	20	21
Report formats: 1. number 2. creation 3. lines/boxes 4. selection	1 **** no n/a	36 *** yes *****	3 set no ****	1 no no n/a
Search: 1. by field 2. global 3. invert selection	***** ***** no	***** no yes	** no no	***** no no
Arithmetic functions	no	yes	no	no
Printer: 1. ZX 2. other 3. configuration	yes no n/a	yes yes ***	yes yes ****	yes no n/a
Microdrive	no	yes	yes	no
On-screen information	****		no	***
Movement between records	***	*****	**	***
Menus	*****	*****	** *	***
Display enhancement	no	yes	no	no

Note: The more stars the better but remember that these three programs are for various reasons, not immediately comparable – they do not pretend to all offer the same facilities. The table is intended to give you some idea of what to look for when thinking of buying a database. And don't buy a database until you've tried it yourself.

The direct approach

If you do not want to bother with the sophistication of such a flexible database as Masterfile a more specialised database may be what you need. If addresses are the backbone of your requirements – for club records, business contacts or just a hi-tech address book, then look at Oxford Computer Publishing's Address Manager.

Latest versions of this program from OCP are compatible with microdrives. There is also a version called +80 Address Manager to drive many of the numerous printer interfaces available for the Spectrum which will operate 80 column printers. A growing number of users have disc drives and OCP can help here too. 350 addresses can be stored on cassette and some 6,000 on disc.

The record format is pre-defined and allows eight data fields to be entered – surname, forenames, address, postcode, phone number and three 3-letter index codes. These index codes allow you to ascribe up to three attributes to each record. These may then be sorted into groups dependent upon one or all of the index codes.

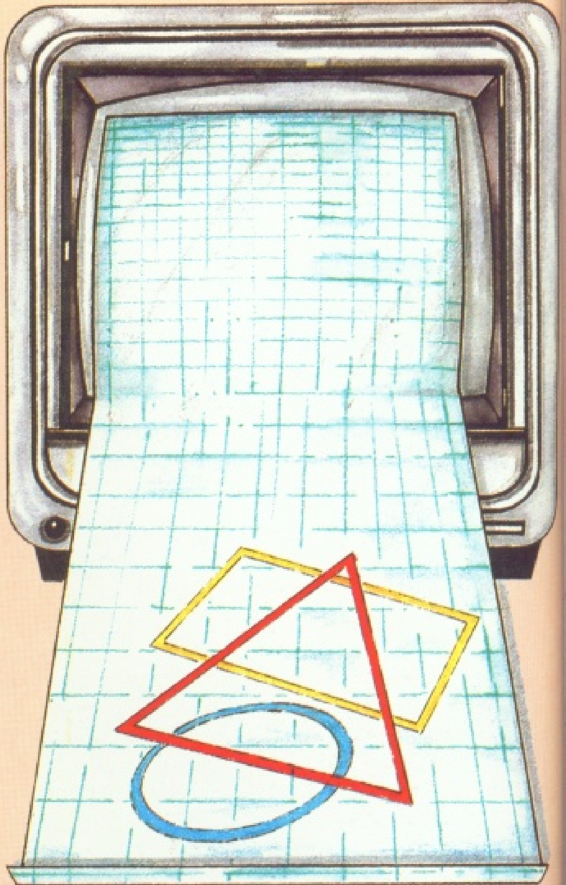
The program is menu-driven and is quite easy to use, but once again a little time spent playing with the testfile database included is well spent. The +80 Address Manager opens with several questions to enable the program to be set up for your selected printer interface. Instructions are also provided for those knowing some machine code to set up the program for their own printer driver.

The data field length for the surname, forenames and each line of the address is 20 characters long, so should be quite adequate for nearly everyone. The major operations such as locating, adding to or amending an entry are all immediately accessible from the main menu.

Printing records calls up another simple sub-menu to define which consecutive block of addresses will be printed. These may be printed in three pre-set formats: Full, where all information is printed; List, name and index only; or Edit, name and address only. There is provision to alter the left margin and the spacing between addresses to facilitate printing labels on continuous stationery.

Selecting any group of records also presents you with a sub-menu, this time requesting the input of the search index codes. The search facilities are rather limited but are probably in keeping with the simple approach of a single use database.

Graphics commands



Get to grips with your high resolution screen display

THE Spectrum's screen is split into two distinct sections. The bottom two character lines are used for entering program lines or data in response to INPUT statements and the top 22 rows are used to print results from programs.

From this you can see the Spectrum screen has 24 character rows in all. Each of these rows can display a maximum of 32 characters, therefore the screen is 32 characters wide by 24 high.

If you cast your mind back to the first graphics article you will remember that each character is made up of dots or pixels on an 8 by 8 matrix. If we now use a bit of maths we can work out the total number of pixels along each side of the top section of the screen.

First we'll work out the total number of dots across. Each row can display 32 characters and each character is 8 dots wide. therefore there are 256 (8*32) dots across the screen.

Multiplying the number of character rows by 8 now tells us the number of dots up the side of the screen. That is 22*8 = 176 (the bottom two character rows are for input only). This leaves us with a grid 256 dots across by 176 down. In other words, there are 45056 (256*176) dots on the screen. These dots, points or picture elements (pixels) as they're known, make up what is called the high resolution screen.

Using special Spectrum Basic commands we can set particular points to the INK colour, connect two points with a line, draw circles and so on. There are four Basic commands associated with graphics on the Spectrum. They are PLOT, DRAW, CIRCLE and POINT.

This article examines how they work and how the screen's laid out. By the time you've worked through the panels you'll understand why the Spectrum has such a reputation for high quality graphics.

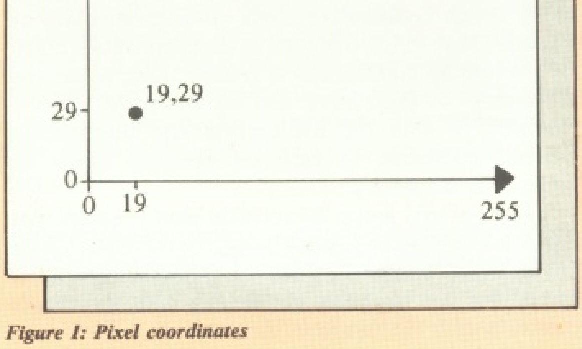


Figure I: Pixel coordinates

How the graphics screen works

IF you're familiar with graphs then you'll have a head start because the screen is mapped out in the same way as a graph. A pixel is specified – computer buffs say "addressed" – by giving its column and row position relative to the bottom left corner. These numbers are known as the pixel's coordinates.

The column position is the column number of the pixel from the left edge of the screen. In fact the pixel number you give is one less than the actual physical position. In other words, the second column is one, and so on. This is because computers prefer to start counting from zero rather than one.

The row position is the row number of the pixel (minus 1) from the bottom edge of the screen. Figure I shows how this works. For example, the coordinates 19,29 correspond to the pixel 20 across and 30 up from the bottom left of the screen – this position is shown on

PLOT

THE PLOT C,R command displays a single pixel at the specified position *C,R* using the current INK colour – *C* and *R* correspond to the column and row position of the pixel to be displayed.

Program I displays a red pixel at the top right corner of the screen.

```
10 REM PROGRAM I
20 INK 2
30 PLOT 255,175
```

The coordinates given in line 30 indicate that the pixel is in row 176, the very top row of the screen, column 256, the rightmost column. Thus the pixel appears at the top right corner.

Program II uses PLOT to draw a horizontal, vertical and diagonal line on the screen. As each pixel is drawn the column or row value which is changing is displayed at the top of the screen.

Lines 40 to 70 are responsible for drawing a horizontal line of dots across the centre of the screen from left to right. To achieve this the column position increases while the row position remains the same. The vertical line is PLOTted by lines 100 to 130. This time the row position is increased. The diagonal line is achieved by altering both the row and column positions at the same time.

```
10 REM PROGRAM II
20 INK 3: PAPER 7: CLS
30 REM HORIZONTAL
40 FOR c=0 TO 255
50 PLOT c,88
60 PRINT AT 0,0;"column =";c
70 NEXT c
80 GO SUB 500
90 REM VERTICAL
100 FOR r=0 TO 175
110 PLOT 128,r
120 PRINT AT 0,0;"row =";r
130 NEXT r
140 GO SUB 500
150 REM DIAGONAL
160 FOR h=0 to 175
170 PLOT h,h
180 PRINT AT 0,0;"column and row =";h
190 NEXT h
200 STOP
500 FOR h=1 TO 250
510 NEXT h
520 CLS
530 RETURN
```

As with PRINT, the INK command can be embedded in PLOT statements temporarily to change the colour of the graphics pen – the colour the pixels appear in. Program III draws a red pixel in the top right corner of the screen without altering the previous INK setting.

```
10 REM PROGRAM III
20 INK 5
30 PRINT "THIS TEXT IS CYAN"
40 PLOT INK 2;255,175
50 PRINT "THIS IS CYAN AS WELL"
```

Line 30 prints a message in cyan. Now INK 2 is temporarily selected and a red pixel is placed at 255,175. Line 50 proves the INK hasn't been affected by printing another message. This is in cyan, just like the message produced by line 20.

Figure I.

When coordinates are specified the column position must always be given first, otherwise the wrong location will be chosen.

If the Spectrum tries to access a coordinate outside the screen area the error *integer out of range* will occur. This happens when the column position is less than 0 or greater than 255 or when the row position is less than 0 or greater than 175.

Whenever characters are printed on the screen a marker called the text cursor points to the next free character location. This is where the next character will be positioned. However there's another cursor which you probably haven't met before -- the graphics cursor. This invisible cursor marks the position of the last pixel displayed. Unlike the text cursor it isn't moved one place to the right after a graphics command has been used.

Whenever a CLS instruction is executed the graphics cursor is re-positioned, or homed, to the bottom left corner (coordinate position 0,0).

As with text, the colour of the graphics can be changed using INK. This allows you to draw red circles and blue lines, for example. However things aren't quite as easy as you may think. This is because of the way in which the Spectrum's screen has been designed.

To cut down on the amount of memory required to store the screen display without losing the high resolution graphics, the Spectrum only permits two colours at each character position – the foreground and background colour. This means that it isn't possible to have more than two differently coloured pixels in the same character square.

Whenever a new pixel is placed on the screen any ink pixels in the same character cell as the new pixel will be changed to the current INK colour. This is limiting if you want to produce complex colour drawings, but it's one of the things you have to live with, and it is not all that restricting in practice.

DRAW

DRAW C,R connects two screen coordinates with a continuous line. The line always starts at the graphics cursor's current position. The end position is rather unusual in that the coordinate *C,R* isn't the actual end position, but a relative location to the graphics cursor's position. In other words *C,R* gives an offset from the cursor position. This is best explained using an example.

If the graphics cursor is at coordinate 50,50 and a DRAW 20,20 command is executed, a line will be drawn between 50,50 and 70,70. The end position (70,70) is calculated by adding together the corresponding column and row positions of the two coordinates.

This is demonstrated by Program IV.

```
10 REM PROGRAM IV
20 PLOT 50,50
30 DRAW 20,20
```

Of course there will be times when a DRAW statement has a negative relative position change. This occurs when the end position of the line has a column or row position less than the current graphics cursor's. In this case a number has to be subtracted from the current position to get to the new one.

Suppose the graphics cursor is at 100,100. If we wanted to draw a line to position 80,40 we would use DRAW −20,−60. Program V shows this in action.

```
10 REM PROGRAM V
20 INK 4
30 PLOT 100,100
40 DRAW −20,−60
```

After INK 4 (green) has been selected line 30 moves the graphics cursor to position 100,100 and PLOTs a single point. Line 40 draws the line using the relative position −20,−60. Since the graphics cursor is at 100,100 the end position of the line will be 80,40 (100−20=80, 100−60=40).

Program VI draws random lines on the screen. This is quite a tricky task to perform, since the Spectrum produces an error when an invalid screen coordinate is used. To complicate matters, the coordinates for DRAW are relative. Therefore a check must be made

CIRCLE

PERHAPS the most interesting of all the graphics commands is CIRCLE. This draws the outline of a circle on the screen. The CIRCLE statement is followed by three numbers:

CIRCLE C,R,RAD

where *C* and *R* are the column and row positions of the centre of the circle, and RAD is the radius, the distance between the centre point and the edge, in pixels. Thus the command:

CIRCLE 128,88,35

will draw a circle near the centre of the screen with a 35 pixel radius.

You must be careful that no part of a circle leaves the screen, otherwise an *integer out of range* error will occur and your program will stop.

Program VII draws magenta circles in the centre of the screen, each one with a random radius between 0 and 49.

```
10 REM PROGRAM VII
20 INK 3
30 CIRCLE 128,88,INT (RND*50)
40 GO TO 20
```

Line 20 selects the colour magenta. Line 30 draws the actual circle. The last number in the CIRCLE command selects the random radius.

After a circle has been drawn the graphics cursor ends up at the right hand edge of the circle on the same row as the centre. This may seem rather an odd place to finish, but it is the last position visited by the graphics cursor.

The DRAW command can be used to draw part circles, although this is rather complex to understand if you're not mathematically inclined. This is achieved by adding a third number to the DRAW instruction indicating the angle being turned through in radians – where PI (3.1416) radians equals 180 degrees (a semi-circle).

For example:

```
10 REM PROGRAM VIII
20 PLOT 70,70
30 DRAW 50,30,PI
```

will draw a half circle between the coordinates 70,70 and 120,100. If you change the PI in line 20 to PI/2 (PI divided by 2 - 90 degrees) a quarter circle will be drawn between the two points.

to ensure that both the column and row positions of the next coordinate are valid.

```
10 REM PROGRAM VI
20 PAPER 7: INK 4
30 CLS
40 LET r=0
50 LET c=0
60 LET rc=INT (RND*100)–50
70 LET rr=INT (RND*80)–40
80 IF (c+rc)<0 OR (c+rc)>255 THEN GO
TO 60
90 IF (r+rr)<0 OR (r+rr)>175 THEN GO
TO 60
100 DRAW rc,rr
110 LET c=c+rc
120 LET r=r+rr
130 GO TO 60
```

Lines 30 to 50 "home" the cursor and set up the current column and row variables to zero. Now lines 60 and 70 pick a random relative change for the column and row positions.

To ensure that no errors are produced the next two lines check that the coordinates are within the screen area. If they aren't the program branches back to line 60 to re-select the relative direction changes. Line 100 draws the line on the screen. Lines 110 and 120 change the column and row positions so that they point to the end of the line. Finally, line 130 causes a loop back to 60.

Some micros have a special command called MOVE which moves the graphics cursor to a certain position without displaying anything on the screen. However the Spectrum doesn't have a MOVE instruction. This is because whenever you move to a position using PLOT you are almost certainly going to DRAW from that position. Therefore the stray dot that appeared when the PLOT was executed now becomes the first pixel in the line.

POINT

THE final command associated with Spectrum graphics is POINT. This allows you to test if a pixel is set in the INK (foreground) or PAPER (background) colour. If the pixel is in the INK colour the value 1 is returned, otherwise the pixel is in the PAPER colour, in which case 0 results.

POINT is used to check if a pixel is being placed on top of another one. For example, if you wanted to place 10 randomly positioned dots on the screen you would use POINT to ensure you don't overprint another pixel. If not there's a chance you'll end up with less than 10 dots due to the randomness of the column and row positions.

Unlike PLOT, DRAW and CIRCLE, the column and row positions following the statement must be enclosed in brackets. Therefore,

LET a=POINT (100,100)

is valid, whereas:

LET a=POINT 100,100

is not. If you do forget the brackets the line will not be accepted when you try to add it to your program.

POINT is a special instruction because it returns a result, therefore you must use it in the correct way.

10 POINT (100,100)

is an invalid line since you do not use the result returned. However if you alter the line to:

10 LET a=POINT (100,100)

the variable a will contain the number 1 or 0.

To show POINT in action here's a program that counts the number of dots along a particular pixel row and prints out the result:

```
10 REM PROGRAM IX
20 CLS : INK 2
30 PLOT 20,50
40 PLOT 72,50
50 PLOT 153,50
60 PLOT 210,50
70 LET n=0
80 FOR h=0 TO 255
90 IF POINT (h,50)=1 THEN LET n=n+1
100 NEXT h
110 PRINT n
```

Lines 30 to 60 are responsible for placing four dots along pixel row 50 (the 51st row up). Line 70 sets the variable *n* to 0. This is used to count the number of pixels found. Finally all of row 50 is checked for INK coloured pixels and the result is printed out. If line 90 finds an INK pixel, 1 is added to the variable *n*.

POINT does not alter the current position of the graphics cursor. Take a look at Program X.

```
10 REM PROGRAM X
20 INK 4
30 PLOT 50,50
40 LET a=POINT (100,100)
50 DRAW 30,30
```

Line 40 doesn't move the graphics cursor to position 100,100, but leaves it at 50,50. The DRAW in line 50 proves this by drawing a diagonal line to position 80,80. If the DRAW has started at 100,100 then a pixel would have been left at 50,50.

Understanding hexadecimal

Part Three of the series that helps to take the mystery out of the way your Spectrum handles numbers

%10111011 = 187
%10101101 = 173
%10001111 = 151
%11110110 = 246

Table I

THE Spectrum – and all other machines based on the Z80 microprocessor – handles its binary numbers in groups of eight bits at a time. Such a group of eight is called a byte.

However, while handling eight bits at a time is satisfactory from the machine's point of view, from the human side of things it's rather difficult to manage. Those 1s and 0s are far too prone to error. Look at Table I for instance. It contains an error – can you find it?

It's all too easy to slip up when handling binary numbers – a single 1 in the wrong place and all is lost! To make things easier to deal with, when I am copying out binary numbers I put a wavy line between bits 3 and 4 to split the byte into two equal groups of four. For example, if I were copying:

% 10001111(= 143)

I would write:

% 1000 ≀ 1111

Actually, splitting the byte into two groups of four bits is standard practice – each group of four bits is called a "nybble", would you believe. It's not too hard to see that the biggest number you can represent in a nybble is 15, and the smallest is 0, **%1111** and **%0000** respectively. After all, you've only got four bits to play with! So we can split up our byte into two nybbles of four bits each.

Now when we split up a binary number in this manner we call the "left-hand" nybble the most significant nybble (MSN) and the "right hand" nybble the least significant nybble (LSN).

We have already created one new number system – the binary system. Let's design another one that combines the advantages of the denary system with those of the binary. That is, it will be easy to read and write, yet will still allow us to perceive the binary manner in which the machine handles things.

The system we want is called hexadecimal. This consists of using our standard digits 0 to 9 for the number zero to nine respectively, and the letter A to F for the numbers 10 to 15. In this way it allows us to code the numbers available in a nybble (that is, 0 to 15) with just one digit. This digit will be in the range 0 to 9 or A to F.

It may take a while to adjust to the idea of using letters of the alphabet for numbers, but it soon becomes second nature. You just have to get used to counting:

1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Remember, there are B people in a cricket team, D in a rugby league team and F in a rugby union team.

Hex to denary . . .

TO translate a two digit hexadecimal number into denary simply multiply the number in the left-hand column by 16 and add it to the number in the right-hand column – remembering to translate A to F if necessary. The second column has the value 16 since the first column can only handle numbers up to 15 (&F) – the largest you can fit into a nybble (%1111). After 15, you have to use a second column for 16, that is & 10.

Just as in denary, we "carry" at 10 since the largest value our columns can handle is 9, so in hexadecimal we carry at 16, since the largest value our columns can handle is 15 (&F). It is the fact that we carry at 16 that gives this number system its name "hexadecimal" – here "hex" stands for 6, "decimal" for ten. "Hexadecimal" = 6 + 10 = 16.

Given a second column, &10, as we have seen is 16, 17 will be &11, while &12 is 18 and so on until we reach 31, which is &1F. We have then run out of legal digits for the units column, so if we want to go on to 32 we had better give ourselves another 16, and set the units column back to zero, that is &20.

Another way of looking at the second column is that it comes from the most significant nybble. To turn the least significant nybble into the most significant nybble we have to shift it over to the left four times.

If you cast your mind back this is equivalent to multiplying it by two four times in succession, that is 2×2×2×2=16. This is why a hexadecimal digit representing the most significant nybble is 16 times larger than the same digit representing the least significant nybble.

The largest number you can store in a two-digit hexadecimal number is &FF = 15 × 16+15=255. This is, of course, the same as the largest number we could store in a binary byte – we often refer to a two digit hexadecimal number simply as a byte.

There are C months in a year, and E days in a fortnight.

Now just as we prefix all our binary numbers with %, we prefix our hexadecimal numbers with &, to avoid confusion. So &F means 15, while &9 means 9. Studying Table II will really pay dividends – I suggest you practice writing down bit patterns of nybbles and their hexadecimal equivalents until it becomes second nature.

Given that we can encode a nybble in one hexadecimal digit, and that a byte consists of two nybbles, it should readily be apparent that we can encode a byte as two hexadecimal digits side by side, for example:

$$\begin{tikzcd}
& \%\underbrace{1010}\,\vdots\,\underbrace{1001} \arrow[dl] \arrow[dr] & \\
\%1010 \arrow[d] & & \%1001 \arrow[d] \\
\&A \arrow[dr] & & \&9 \arrow[dl] \\
& \&A9 &
\end{tikzcd}$$

That is:

%10101001 = &A9 = 169

You just split the byte up into two nybbles – a left hand and a right hand nybble, encode each as a hexadecimal number, then put the two side by side. You can go from hexadecimal to binary just as easily:

$$\begin{tikzcd}
& \&8D \arrow[dl] \arrow[dr] & \\
\%1000 \arrow[d] & & \%1101 \arrow[d] \\
\overbrace{1000} & \vdots & \overbrace{1101}
\end{tikzcd}$$

That is:

&8D = %10001101 = 141

Although you have probably never thought of it in these terms, you are well aware that the value a digit represents depends on the column it is in. The number 230 is not as large as 320, though both numbers contain the same digits.

In hexadecimal coding too the column a digit is in is important. For example, &10 is far greater than &01. In binary each column is worth twice the preceding one. In denary, our usual number system, each column is worth 10 times the preceding one. In hexadecimal, each column is worth 16 times the preceding one.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Table II

Believe or not, the columns in a four digit hexadecimal number, from greatest to least, are worth **4096, 256, 16** and 1 respectively. This means that:

& 1101 = 4096 + 256 + 1 = 4353

For the moment let's concentrate on the two digit, that is, two column, hexadecimal number, as these are all we need to store our bytes in. In this case the left-hand column is the "sixteens" column, the right hand the units column.

So:
16 1
& 2 1 = 2*16+1 = 33
16 1
& 2 D = 2*16+13 = 45
16 1
& 8 0 = 8*16+0 = 128
16 1
& C 0 = 12*16+0 = 192

As you'll see from the relevant panels, translating from hexadecimal to denary and vice-versa isn't too complicated – it just needs practice. So get started for next time!

. . . and back again

TO obtain the hexadecimal equivalent of a positive integer (whole number) less than 256, we divide it by 16. The quotient is the left hand digit, the remainder the right hand, translating into A to F where necessary. For example: 174÷16=10 R 14

That is:
&A R &E
Hence:
174=&AE

In case you aren't too hot at your sixteen times table, there's a program printed on the right which will do it for you.

See if you can follow it. It will do your understanding of hexadecimal numbering a power of good, not to mention your knowledge of Spectrum Basic!

```
 10 REM Decimal
 20 REM   to
 30 REM  Hex
 40 LET S$="0123456789ABCDEF"
 50 CLS
 60 INPUT "Number ",number
 70 PRINT AT 8,9;"Decimal is ";number
 80 IF number>255 OR number<0 THEN  GO TO 50
 90 LET left=INT (number/16)
100 LET right=number-left*16
110 LET left=left+1
120 LET right=right+1
130 LET L$=S$(left)
140 LET R$=S$(right)
150 PRINT AT 12,5;"Hex equivalent is ";
160 PRINT L$;R$
170 IF INKEY$<>"" THEN  GO TO 170
180 IF INKEY$="" THEN  GO TO 180
190 GO TO 50
```

Spectrum 128

A quick look at the features that distinguish the latest model in the Sinclair range

THE new Spectrum 128 is actually two computers in one. Rather than bring out a new model that would not have been able to use the mountains of software already written for the original Spectrum, Sinclair has come up with a machine that offers advanced features, but which can be turned back into a standard 48k Spectrum Plus merely by typing in the magic word Spectrum.

Best of both worlds

The idea is to give you the best of both worlds. In both 128k mode and 48k mode you get improved sound output because the new machine sends the sound through the television speaker. If you can afford a monitor you will also get improved picture quality, as there are now video outputs to link to either RGB or composite type video monitors.

The other new facilities are only available when in 128k mode, but two of these can still be useful to the 48k mode, even if they can't be used when the micro is actually in 48k mode.

The first is the built-in television pattern generator, which sends a multi-coloured pattern and an intermittent audio tone to the TV. This is an aid to tuning in the TV and will be a great help to getting a good screen display, especially for first-time users.

The other feature will also be a big help to beginners. This is the Tape Tester, an aid to setting the cassette tape player volume to exactly the right volume for best loading of tape software.

The new 128k mode features a new Basic interpreter which supports almost all the features of the original Spectrum Basic, plus some new commands. In fact it is very similar to the original Basic except that you no longer use Sinclair's idiosyncratic single key entry of keywords.

Instead you now have to spell out the keywords letter by letter in the manner used by every other computer. Many people will welcome this, and those that don't like it can simply switch back to 48k mode and have single key entry again.

The new Basic retains the syntax checking feature of the original Sinclair Basic, and will not accept a line with an obvious error of Basic usage. This means that while it is now possible to mis-spell a Basic keyword the computer will latch the mistake and bring it to your attention when you try to enter the line.

The 128k Basic has many more editing functions and features which make changing the program easier and quicker than with the single character delete on the 48k.

The principle change is that you can now use the arrow keys to move around the whole screen instead of having to call each line down into the editing area for alteration.

There is also an optional extra add-on keypad which gives you a range of new editing keys that let you move around the program in steps of more than one character or line per keypress.

Finally there is now a renumber command to tidy up those line numbers after a heated spell of editing.

It is possible to use part of the expanded memory of the new machine as a RAM disc which will allow you to have more than one Basic program in the memory at the same time, ready for instant loading into the active Basic area for use.

PLAYing with sound

One of the main improvements to the hardware is the addition of a three channel sound chip, and the 128k Basic takes full advantage of this facility by use of a single keyword, PLAY.

This is followed by a string containing any number of parameters for the PLAY to work with, and you can use these parameters to make your instructions as complicated as you like.

The simplest form of the command, PLAY "B", will play the note B in the currently selected octave range. More notes can be added to the string, or you can put in specifiers for sharps, flats, sound effects, different sound channels, tempo and so on.

The PLAY command will also control up to eight channels of music via the new Midi port, which is actually the same socket as the RS232 port. This is an exciting new area for computing, and I expect to see some versatile Spectrum programs for composing and controlling synthesisers becoming available very soon.

Sinclair Research has made sure that the new computer is compatible with Interface 1 and the microdrives, but can't be certain that all the devices made by other companies for the 48k Spectrum will work with the 128, so you are advised to ask the supplier if their add-on is compatible.

Save £2
– See Centre Pages

YOU can go for gold ...with the MICRO OLYMPICS

This is the package that broke all records! More than a game – it's a brilliantly written collection of ELEVEN great track and field events!

Ever imagined yourself as another Seb Coe? Then try to run against the world record holder at 1500 metres. And if that distance is too much for you then there's always the 100, 200, 400 and 800 metres to have a go at.

Not much good at running? Don't worry, MICRO OLYMPICS has many more challenges for you. Why not try your skill at the high jump or the long jump?

And if you can't beat the computer at running or jumping then you can always throw things around in frustration! The trouble is that it's just as hard to be a champion at the discus, the hammer or the javelin.

And the pole vault takes the event to new heights!

Yes, it's fast, furious fun, pitting yourself against the world's best times and distances on your micro.

You may not be another Steve Ovett or Alan Wells, but with practice you COULD become the Micro Olympics Champion!

Play Micro Olympics – and let your fingers do the running!

Send for it today

Please send me ____________ copy/copies of Micro Olympics

☐ I enclose cheque made payable to Database Publications Ltd. for £ ____________

I wish to pay by
☐ Access ☐ Visa No.____________ Expiry date____________

Signed____________

Name____________

Address____________

Spectrum 48k
£5.95

Post to: Micro Olympics offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

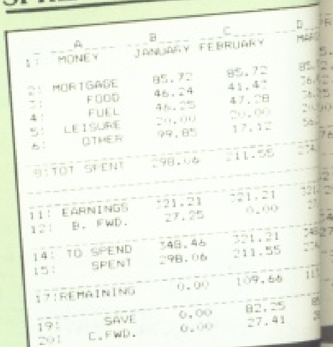
QUICK TO LEA

THAT'S...

FINALIST MICRO BRITISH COMPUTING AWARDS 1985

MINI OFFICE

SPREADSHEET



```
      A           B        C        D
1:  MONEY     JANUARY FEBRUARY

2:  MORTGAGE    85.72    85.72
3:      FOOD    46.24    41.47
4:      FUEL    46.25    47.28
5:   LEISURE    20.00    20.00
6:     OTHER    99.85    17.12

9:TOT SPENT    298.06   211.55

11: EARNINGS   321.21   321.21
12:  B. FWD.    27.25     0.00

14: TO SPEND   348.46   321.21
15:    SPENT   298.06   211.55

17:REMAINING     0.00   109.66

19:     SAVE     0.00    82.25
20:  C.FWD.      0.00    27.41
```

JUST LOOK WHAT THIS PACKAGE CAN DO!

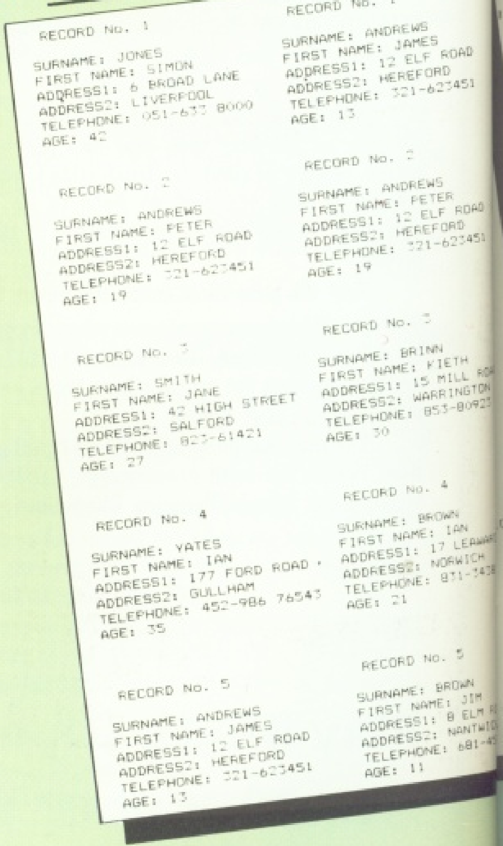
WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

SPREADSHEET – Use your micro to manage your money! *Features:* Number display in rows and columns ● Continuous updating ● Update instantly reflected throughout spreadsheet ● Save results for future amendments.

GRAPHICS – Turn those numbers into an exciting visual display! *Features:* 3D bar chart ● Pie chart ● Graph.

DATABASE – Use it like an office filing cabinet! *Features:* Retrieve files at a keystroke ● Sort ● Replace ● Save ● Print ● Search.

DATABASE



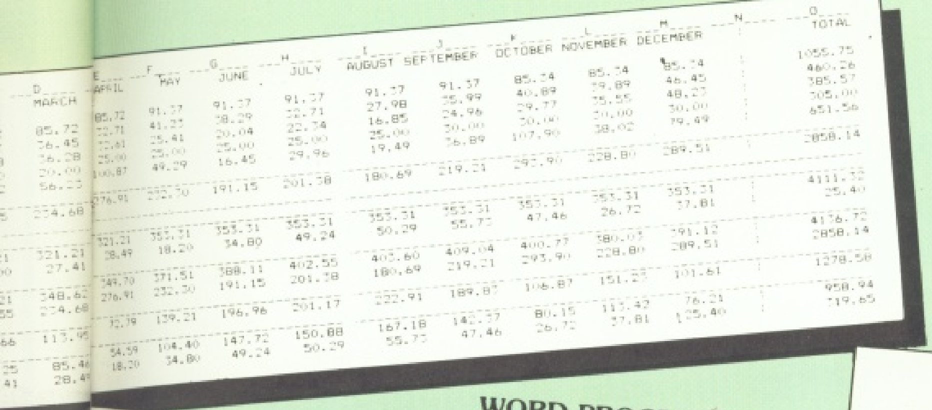
```
RECORD No. 1                       RECORD No. 1

SURNAME: JONES                     SURNAME: ANDREWS
FIRST NAME: SIMON                  FIRST NAME: JAMES
ADDRESS1: 6 BROAD LANE             ADDRESS1: 12 ELF ROAD
ADDRESS2: LIVERPOOL                ADDRESS2: HEREFORD
TELEPHONE: 051-633 8000            TELEPHONE: 321-623451
AGE: 42                            AGE: 13

RECORD No. 2                       RECORD No. 2

SURNAME: ANDREWS                   SURNAME: ANDREWS
FIRST NAME: PETER                  FIRST NAME: PETER
ADDRESS1: 12 ELF ROAD              ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD                 ADDRESS2: HEREFORD
TELEPHONE: 321-623451              TELEPHONE: 321-623451
AGE: 19                            AGE: 19

RECORD No. 3                       RECORD No. 3

SURNAME: SMITH                     SURNAME: BRINN
FIRST NAME: JANE                   FIRST NAME: KIETH
ADDRESS1: 42 HIGH STREET           ADDRESS1: 15 MILL RO
ADDRESS2: SALFORD                  ADDRESS2: WARRINGTON
TELEPHONE: 823-61421               TELEPHONE: 853-80923
AGE: 27                            AGE: 30

RECORD No. 4                       RECORD No. 4

SURNAME: YATES                     SURNAME: BROWN
FIRST NAME: IAN                    FIRST NAME: IAN
ADDRESS1: 177 FORD ROAD            ADDRESS1: 17 LEAWA
ADDRESS2: GULLHAM                  ADDRESS2: NORWICH
TELEPHONE: 452-986 76543           TELEPHONE: 871-343
AGE: 35                            AGE: 21

RECORD No. 5                       RECORD No. 5

SURNAME: ANDREWS                   SURNAME: BROWN
FIRST NAME: JAMES                  FIRST NAME: JIM
ADDRESS1: 12 ELF ROAD              ADDRESS1: 8 ELM
ADDRESS2: HEREFORD                 ADDRESS2: NANTWIC
TELEPHONE: 321-623451              TELEPHONE: 681-4
AGE: 13                            AGE: 11
```

...and it's all a

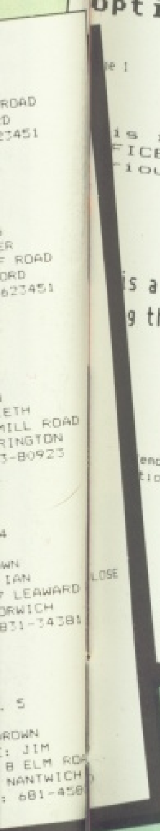
price of just

RN, EASY TO USE

Spectrum 48k and Spectrum+

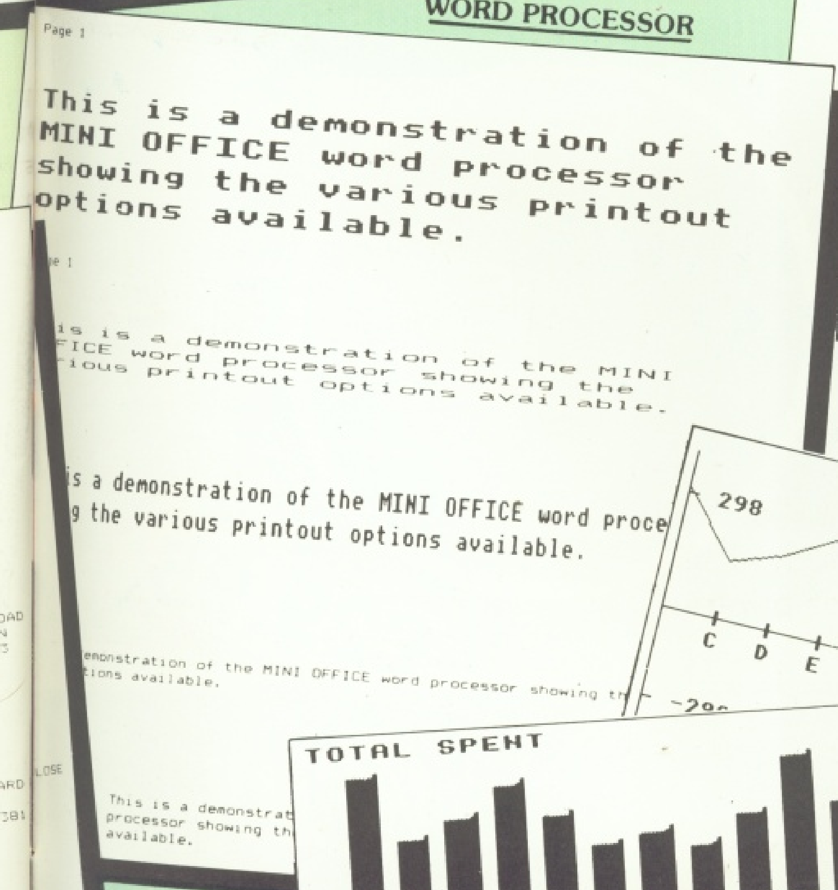


```
         D          E        F        G        H        I         J          K         L         M         N      O
       MARCH      APRIL     MAY      JUNE     JULY    AUGUST  SEPTEMBER  OCTOBER  NOVEMBER  DECEMBER          TOTAL
       85.72      85.72    91.37    91.37    91.37    91.37    91.37      85.34     85.34     85.34          1055.75
       36.45      22.71    41.25    38.29    32.71    27.98    35.99      40.89     39.89     46.45           460.26
       36.28      25.61    25.41    20.04    22.34    16.85    24.96      29.77     35.55     48.23           385.57
       20.00      25.00    25.00    25.00    25.00    25.00    30.00      30.00     30.00     30.00           305.00
       56.25     100.87    49.29    16.45    29.96    19.49    36.89     107.90     38.02     79.49           651.56
      234.68     276.91   232.30   191.15   201.38   180.69   219.21     293.90    228.80    289.51          2858.14
                                                                                                          4111.32
                                                                                                            25.40
      321.21     321.21   353.31   353.31   353.31   353.31   353.31     353.31    353.31    353.31          4176.72
       27.41      28.49    18.20    34.80    49.24    50.29    55.73      47.46     26.72     37.81          2858.14
      348.62     349.70   371.51   388.11   402.55   403.60   409.04     400.77    380.03    391.12          1278.58
      234.68     276.91   232.30   191.15   201.38   180.69   219.21     293.90    228.80    289.51
      113.95      72.79   139.21   196.96   201.17   222.91   189.83     106.87    151.23    101.61           958.94
       85.46      54.59   104.40   147.72   150.88   167.18   142.37      80.15    113.42     76.21           319.65
       28.49      18.20    34.80    49.24    50.29    55.73    47.46      26.72     37.81    125.40
```



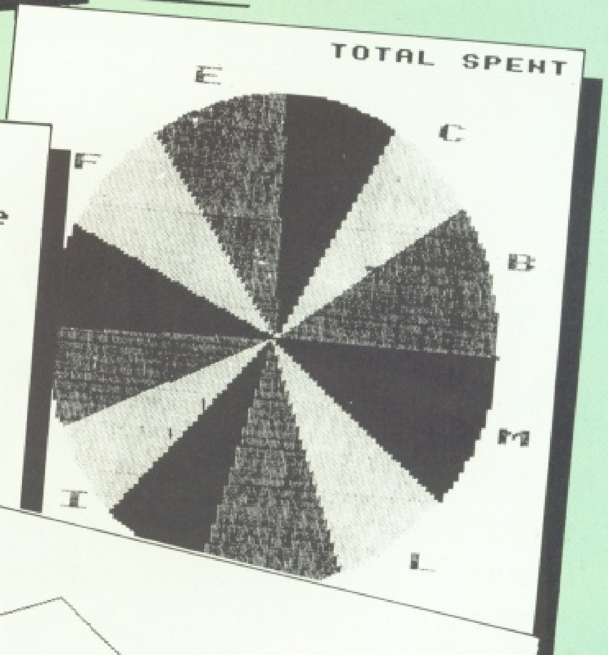
```
RDAD
D
23451

ER
F ROAD
ORD
623451

ETH
MILL ROAD
RINGTON
3-80923

4
WN
IAN
7 LEAWARD
ORWICH
831-34381

. 5
ROWN
: JIM
8 ELM ROA
NANTWICH
: 681-458
```

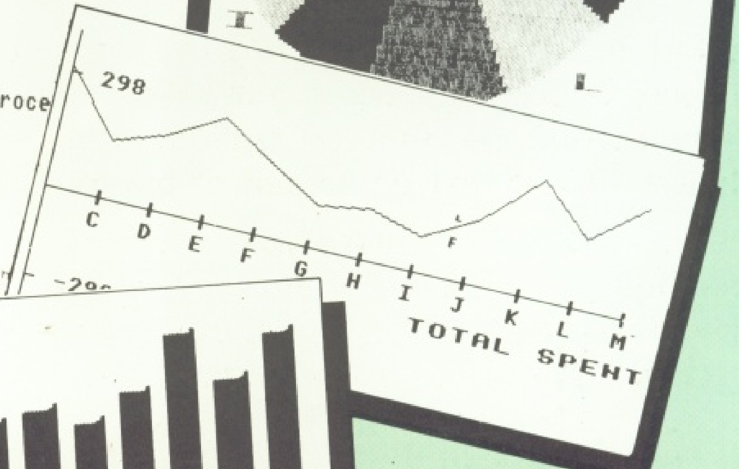
WORD PROCESSOR

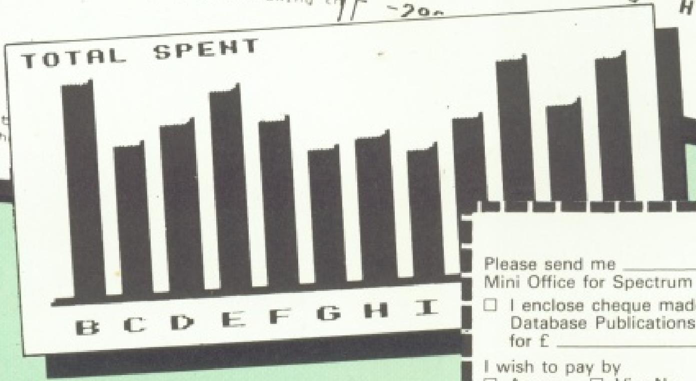


```
Page 1

This is a demonstration of the
MINI OFFICE word processor
showing the various printout
options available.

ge 1

is is a demonstration of the MINI
FICE word processor showing the
rious printout options available.

is a demonstration of the MINI OFFICE word proce
g the various printout options available.

emonstration of the MINI OFFICE word processor showing th
tions available.

LOSE

This is a demonstrat
processor showing th
available.
```

GRAPHICS







at the unbelievable
st £5.95 CASSETTE

DATABASE SOFTWARE

Only £5.95

Please send me __________ copy/copies of Mini Office for Spectrum 48k/Spectrum+

☐ I enclose cheque made payable to Database Publications Ltd. for £ __________

I wish to pay by
☐ Access ☐ Visa No. __________ Expiry date __________

Signed __________

Name __________

Address __________

____

Post to: Mini Office Offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

SOUND IDEAS FOR YOUR SPECTRUM

The Three Channel Sound Synthesiser interface incorporates a BEEP audio amplifier and a 3 channel sound synthesiser.
The BEEP amplifier improves the sound quality and output of the BEEP enormously. The 3 channel sound synthesiser adds a totally new dimension to sound on your Spectrum. It allows you to program your own music with harmonies, explosions, zaps, chimes, whistles and an infinite range of other sounds over a full 8 octaves.

Based around the popular AY-3-8912 sound chip it gives you complete control (from basic or M/C) over 3 channels of tone and/or white noise, plus envelope and volume control. It comes with its own pod mounted (4″) speaker with 1 metre of cable so that it can be positioned anywhere.
Once this is fitted to the expansion port your programs will never sound the same again!

ONLY £29.95

It's Available NOW!

NEW!

Please rush me the following

.... Three Channel Sound Synthesiser Interfaces @ £29.95 each. £______

Please add post and packing £1.25

I enclose cheque/PO/Cash for ...... Total £______

or debit my Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signature ______________________ Name ______________________

Address ______________________

Or send S.A.E. for the New D.K.Tronics Spectrum Catalogue

"Available direct or from good computer shops anywhere"

dk'tronics

DK Tronics Ltd., Unit 6, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ. Telephone: (0799) 26350 (24 hrs) 5 lines

The Spectrum Connection

Speech synthesis

... or how your micro can talk back. We discuss what's involved

ANY computer, from the humblest home micro to the biggest mainframe, must be equipped with a man/machine interface. Put simply this means it must have some way of accepting information from us – normally a keyboard – and giving results back – traditionally a screen of some kind.

While the physical interface, in one form or another, is supplied as a matter of course we are still forced to to communicate with the system by means of a very artificial language. Commands such as RUN, LOAD and SAVE may be meaningful to the operating system but they certainly aren't "natural".

Talking about it

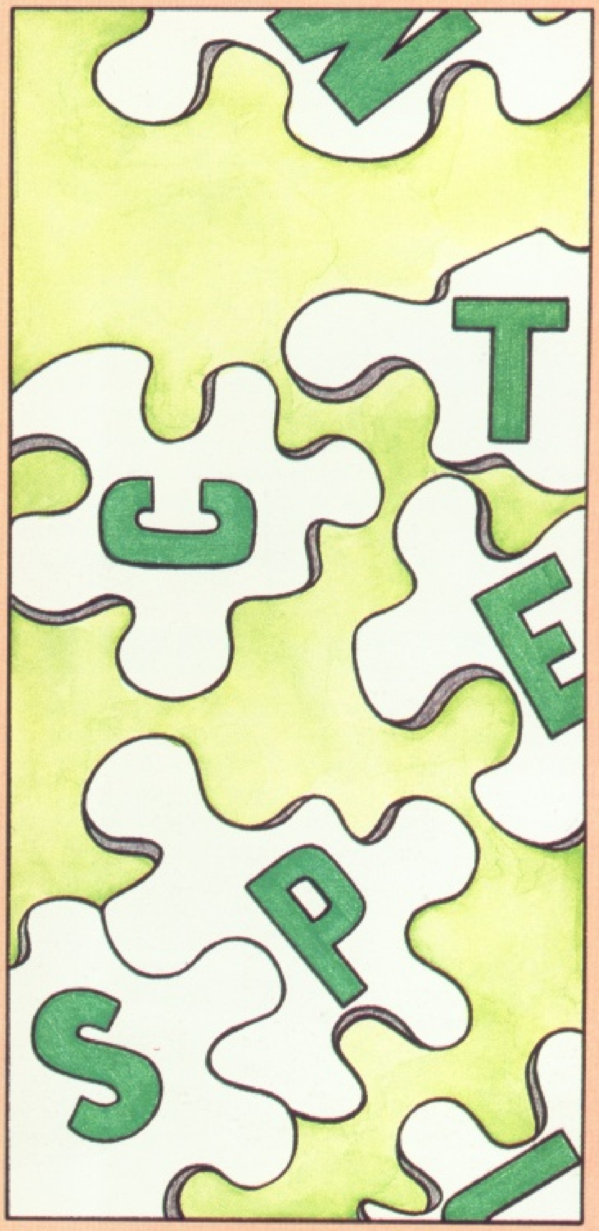
The natural communication system for humans is speech, not typing messages on keyboards and watching messages on television screens. If the computer could be made to reply vocally and, perhaps, understand spoken commands – even if the utterances were phrased in the same way as those given through keyboard and screen – the computer would be easier to use. Before any computer system can generate or understand spoken words the complex waveforms that make up speech must be turned into a digital form that the computer can cope with.

In total contrast to the problems facing speech recognition, which will be looked at in a moment, the task of electronically generating speech has pretty much been mastered. When we talk we produce sounds of three distinctly different types. The first is the "voiced" or vowel-type sound – oo, ar, ee, and so on. These are produced by vibrating vocal chords in the throat and the frequency of this vibration determines the vowel sound we hear.

The second major type of sound is the "fricative" or unvoiced sound such as ss, sh, ff and t. Here the air from the lungs rushes past the vocal chords and the frequency of the sound is controlled by the positioning of the lips and tongue. The third sound type is silence or, to be more precise, the gaps that occur within words like six and eight.

To electronically generate speech-like sounds we can use one of two methods. The first, and until recently the most common, is that of synthesis by rule. If we carefully analyse the frequencies contained within speech it is possible to devise a system of rules that allows us to create any given sound from its basic frequencies. For example, the word "too" could be defined as so many milliseconds of the t frequency followed immediately by the oo frequency.

These individual building blocks are called phonemes, or in some instances allophones, and by using them in various combinations any word can be constructed. The characteristics of the original speaker tend to be lost when speech is generated in this way but



the words themselves can be clearly understood. Because the rules for the generation of the various phonemes are built into the equipment itself we are usually able to give the system a list of the phonemes which are then spoken. With a little practice it is possible to generate complete sentences instantly by simply calling up a string of stored phonemes.

The second method relies on the fact that the human ear and brain are very good at filling in gaps. For example, the speech we hear over a telephone line is perfectly understandable. Yet the quality, the range of frequencies we can detect, is only a fifth of that which we would expect from a reasonable hi-fi system. The reason we can understand what we are hearing is because our brain can fill in the gaps. With the reduction in cost of computer memory it is now possible to convert speech into digital information which is then compressed many hundreds of times and stored in a ROM. To cause any of the stored words to be spoken we get the computer to recover the digital information and convert it back into sound. Because the original speaker's words are stored the personal characteristics remain – even the accent is readily detectable in some cases.

Making use of words

The uses for speech synthesis are so varied that it is almost impossible to list them. To start with it can replace taped announcements at railway stations and airports. In America it is widely used on the telephone system to inform callers of wrongly dialled numbers, engaged or withdrawn services and all the other announcements that would otherwise need constant human attention. Speech synthesis units are being incorporated into cars like the Maestro as part of the standard instrumentation. As well as being something of a sales ploy the synthesiser does provide warnings that the driver can hear and act on without having to take his or her eyes off the road.

In the home computer market speech synthesis is generally used to enhance games. Scores are read out and warnings of enemy attack can be given verbally leaving the player free to concentrate on the tactics of the game. Finally, on the educational scene there are devices like the Texas Instruments Speak and Spell which recites a word that must then be spelt correctly and foreign language dictionaries that pronounce the words.

Listen to me

Speech recognition, on the other hand, as featured in science fiction and typified by the HAL computer in 2001 – A Space Odyssey, is unlikely to appear for many years yet – if ever. The prime reason for this non-appearance is that, in English at least, many words can sound the same but have meanings dependant on the context that they appear in. The processing power needed to solve this problem is simply not available at a reasonable price, or in a compact enough form.

There are systems in research laboratories which approach this goal but increasing the number of speakers who can be recognised reduces the number of words that can be accepted at any one time. Typically a multi-speaker recognition system, the only sort that is going to be acceptable to users, will allow between 20 and 30 words to be recognised at a time with a success rate of around 85 to 90 per cent.

Speech recognition is usually tackled in one of two ways. The obvious way is to simply feed all the speech through an analogue to digital converter and use the computer to perform all the hard work. Unfortunately this method has a number of drawbacks, notably the time taken to perform the analysis. For speech recognition to be of any real use, however, the computer must seem to "understand" the speech as fast as a human. The number crunching approach rarely achieves this.

The second method is to use pre-processing. Rather than analyse speech mathematically it is possible to do much of the work electronically. What is then delivered to the computer is already processed information about the frequency content, pitch, energy and so on. Because all this electronic processing is done at the same time, the original speech signal is fed to all the circuits and the analysis is almost instantaneous.

Getting the message

Once information about frequency content, pitch, energy, and so on has been extracted from the speech input, regardless of the method, the actual recognition is performed by comparing the current set of figures against a number of possibilities stored in the computer. The words that are to be recognised are spoken into the system one at a time and the resulting information is stored as a library. Ideally this will be a continuous process with the library being updated by every new speaker.

To recognise a spoken word the computer must match the pattern of information from the input with one or more of the models stored in the current library. In many cases a number of possible matches will be found as parts of other words will match the input pattern. At the end of the search one word should stand out as being more perfectly matched than any of the other possibilities and this is the one which the computer will "recognise". As most of the time taken is used up in searching through the various models the speed of the system will be a trade-off between the range of speakers it can understand and the number of words in the vocabulary.

Already there are commercially available recognition units which can be plugged into home computers but they are very unsophisticated. Systems like "Big Ears" use much of the computer's processing power to recognise just a few words spoken by one person. What is needed before speech recognition can become really useful is an ability to recognise words spoken by any person, regardless of dialect or accent. The limiting factor, at this stage, is the amount of memory available to hold the models – one possibility is that of using a video disc to hold a standard set of models.

Spectrum add-ons

Already there are commercially available Spectrum which, almost without exception, use the allophone method of synthesis. Although this is by far the most flexible way of tackling the problem, the results depend on the quality of the software and the ability of the user to code English sentences into phonetic strings. The commonest chip used in allophone synthesisers is the General Instruments device and so, in theory at least, all the available synthesisers should be capable of the same output quality given equal skill in coding. The alternative method, as epitomised by the National Semiconductor Digitalker chip, is that of stored compressed speech. Here the quality of output is assured but the user is strictly limited in the number of available words. Tricks can, however, be played with this system to increase the vocabulary by joining bits of words together.

Synthesis system	Method used
Currah Microspeech	Allophone
Fuller Orator	Allophone
Cheetah Sweet Talker	Allophone
DCP S-Pack	Compressed speech (Digitalker)

Handling hex

Making sense of machine code – Part III of our hands-on course

NOW that you know all about the numbering systems used in computing – binary, decimal and hexadecimal – we can get on with the fun part. For this we will mostly be using hexadecimal numbers, as the best compromise between the computer's binary and our own decimal preference.

To go on converting hex to decimal with a chart and then using POKE to insert the bytes into memory would be very time-consuming, and we shouldn't have to do all that work when we have a computer to do it. What we need is a little program that will do the conversions and pokes for us while we get on with the real problems of constructing our machine code routines.

Just such a program is shown here. A simple software tool for examining and altering the contents of memory, it displays the addresses and their contents in both decimal and hexadecimal, and allows you to enter a byte in hexadecimal, or just step through the memory to see what is there.

When you have typed in the hex handler save a couple of copies of it, as we will be using it again in later parts of this series. This program will continue to be of use to you even later on, when you have progressed to a full feature assembler and debugger, because it is compact and easy to alter for special purposes.

Some points to watch when entering the program: The Spectrum's user-defined functions are used to do the hex to decimal (and vice versa) conversions. The DEF FN and FN keywords are on the 1 and 2 keys in extended mode. The INT keyword is on the R key, extended mode.

Be very careful about the brackets in lines 110, 120 and 160, and the semi-colons at the ends of lines 150 to 170. There is no space between the quotes in line 190. In line 220, LEN is on the K key, extended mode, and the <> sign is the symbol on the W key, not two separate symbols. All the variable names are in lowercase to avoid any possible confusion with keywords.

When you have typed in the program SAVE it before trying it out, just in case. Any Basic program that does POKEs can cause a crash and lose itself if something goes wrong.

When you have saved it RUN it and enter 2000 as the address when prompted. This address is entered in decimal. Press Enter a few times and the display should look like this:

```
2000   102   07D0   66   66
2001   251   07D1   FB   FB
2002   221   07D2   DD   DD
2003    94   07D3   5E   5E
2004    11   07D4   0B   0B
2005   221   07D5   DD   DD
2006    86   07D6   56   56
2007    12   07D7   0C   0C
2008   124   07D8   7C   7C
2009   181   07D9   B5   B5
2010    40   07DA   28   28
2011    13   07DB   0D

Byte? "C"
```

If you find that your listing of this example varies from ours, then there is a mistake somewhere in your entry of the program. The Byte? at the bottom is prompting you to input a two digit hexadecimal byte, but we aren't going to try that just yet.

The first column of the listing is the decimal address, the second is the decimal memory contents, the third is the hexadecimal address, and the fourth is the hexadecimal memory contents. The fifth column is the hexadecimal memory contents after alteration.

When you just press Enter the program steps to the next location without any changes. If you enter a U it steps up, or back, one location. If you enter a two digit hexadecimal number the program will convert it and POKE it into the location displayed, and then step to the next location.

Hex handling utility

```
 100 LET h$="0123456789ABCDEF"
 110 DEF FN z(a$)=16*(CODE a$(1)
-48-7*(a$(1)>"9"))+CODE a$(2)-48
-7*(a$(2)>"9")
 120 DEF FN y$(a)=h$(1+INT (a/16
))+h$(1+a-16*INT (a/16))
 130 POKE 23658,8
 140 INPUT "Address? ";addr
 150 PRINT addr;TAB 9;PEEK addr;
 160 PRINT TAB 18;FN y$(INT (add
r/256));FN y$(addr-256*INT (addr
/256));
 170 PRINT TAB 25;FN y$(PEEK add
r);
 180 INPUT "Byte? ";a$
 190 IF a$="" THEN  GO TO 240
 200 IF a$(1)="Q" THEN  PRINT :
STOP
 210 IF a$(1)="U" THEN  LET addr
=addr-1: PRINT : GO TO 150
 220 IF LEN a$<>2 THEN  PRINT :
GO TO 150
 230 POKE addr,FN z(a$)
 240 PRINT TAB 30;FN y$(PEEK add
r)
 250 LET addr=addr+1: GO TO 150
 300 POKE 23658,8
 310 INPUT "Hex Address? ";b$
 320 LET addr=256*FN z(b$( TO 2)
)+FN z(b$(3 TO ))
 330 GO TO 150
9999 SAVE "hexload": GO TO 9999
```

Z80

However at the addresses in the example above you will find that entering a hex byte does not cause the number in column five to change from the number in column four. This is because the addresses are in the Spectrum ROM, which cannot be changed. Column five is produced by a PEEK after the POKE, which is a useful check that the memory location has indeed been set to the number you entered, or not, as the case may be. Enter a Q when you want to quit.

Using the hex handler

In Part 2 we wrote a short machine code program to call the ROM routine that clears the screen. This ROM routine was at address &0D6B, or 3435 in decimal. We can now use the Hex Handler to have a look at this routine in the ROM.

RUN the program again and enter 3435 as the address, then press Enter a few times. You should get this:

```
3435   205   0D6B   CD   CD
3436   175   0D6C   AF   AF
3437    13   0D6D   0D   0D
3438    33   0D6E   21
```

As you can see, the first instruction in the clear screen routine is yet another &CD, or CALL, to another address. This is quite common in machine code programs, and you often have to jump around a good deal before you find the place where things actually happen. In this case the address where the screen memory bytes are actually cleared, by loading them with zeros, is at address &0E59 or 3673 decimal. The ROM has to do a lot of other work first, finding addresses and loading pointers, before it can actually set about the clearing operation. If you examine the memory at address 3763 (decimal), you see:

```
3673    54   0E59   36   36
3674     0   0E5A   00   00
3675    19   0E5B   13   13
3676   237   0E5C   ED
```

The bytes &36 and &00 correspond to the assembler mnemonics:

LD (HL),0

These mean "load the address pointed to by the HL register pair with zero". The brackets in the assembler instruction are important. Without them, the assembler instruction would mean "load the HL register pair with zero" instead of the address pointed to by HL. The HL pair has already been loaded with the appropriate screen address by an earlier ROM routine. This is called indirect addressing, and we will be doing a lot of it later on.

Error trapping

THE hex handler program is not extensively trapped against the input of impossible hex digits or the like. Making it foolproof would require a much larger program, and it might not be possible anyway, because fools are so ingenious, as the saying goes.

You must enter two digits for a hex byte, even if it is a zero. If you enter only one the program will give you another chance, but if you enter more than two unexpected things may happen. Similarly, a hex address requires four digits.

The hex digits A to F must be capital letters, so leave the Caps Lock on. Entering letters or symbols which are not valid hex digits will drop you out of the program with an error message.

Now let's try something really dangerous! The Spectrum ROM stores a lot of numbers in RAM for use by its own routines. These are called the system variables, and you will have seen references to them elsewhere. You can do some quite useful things by messing with the system variables, and you can also completely confuse the computer if you are not careful. So take care, or you may crash and have to reload the hex handler.

You will have noticed a POKE in line 130. The effect of this is simply to set the Caps Lock from within the program without you having to press Caps Shift and 2. If we examine this location we find:

```
23658   8   5C6A   08   08
```

You can change this byte to a zero if you wish. You will then notice that the flashing C cursor changes to a flashing L, and any letters that you enter as part of a hex byte will be treated as lowercase. This upsets the hex to decimal conversion functions, and will get you thrown out of the program with an error message. Never mind, just do RUN again and enter 23659 as the address. You will see:

```
23659   2   5C6B   02   02
```

This is the system variable that tells the ROM how many lines there are in the INPUT area of the screen. If you alter this to zero the computer will get very confused the next time it tries to print "Byte?" in the input area, and will crash with a vengeance. You will have to reset the computer and then reload. This exercise is optional!

Try again at address 23620, and press Enter a few times to get a listing of about 10 locations. Do you notice anything odd about this list?

```
23620   255   5C44   FF   FF
23621   150   5C45   AA   F0
23622     0   5C46   00   00
23623     1   5C47   01   01
23624    56   5C48   38   38
23625   200   5C49   C8   C8
23626     0   5C4A   00   00
23627   211   5C4B   D3   D3
23628    95   5C4C   5F   5F
23629   233   5C4D   E9   F6
23630    95   5C4E   5F   5F
```

There are two addresses (23621 and 23629) which have changed their contents while they were being listed without any input from you. These are the system variables used by the Basic interpreter in the ROM to keep track, respectively, of where it is in a program and the location of the Basic variable currently in use.

The ROM is therefore changing the contents of these locations constantly during the running of any Basic program, and of course our hex handler is a Basic program. You may get different numbers at some of these locations than those in the listing above, as the ROM is in charge of them, not us.

At the moment the hex handler expects the initial address to be specified in decimal. This is very convenient when we have been given the address in decimal, as is often the case in the manual and other books. However there are times when we would find it easier to enter an address in hex, such as when we are chasing from one call instruction to another in the ROM or another machine code program.

It is easy enough to add a hex address option to our little utility. Just type in the additional lines 300 to 330 below. Then instead of RUN use the command RUN 300, and you can enter the address in hexadecimal format.

```
300 POKE 23658,8
310 INPUT "Hex Address? ";b$
320 LET addr=256*FN z(b$( TO 2)
)+FN z(b$(3 TO ))
330 GO TO 150
```

A bigger bit of code

Enough of this fooling around – we have a hex handler and we know how to use it, so let's enter a real code program. Here is a little routine needing only 16 bytes of memory which will turn every INK pixel on the screen into a PAPER pixel, and vice versa. In other words it inverts the screen memory.

This routine is completely self-contained and does not call any ROM routines. In assembly language it would look something like Figure I.

Note the comments added to each instruction. These are optional when writing assembly language, but the more comments you put in the easier it will be to understand what is going on when you come back to the listing months later, or when someone else has to work from your listings. Comments are preceded by a semi-colon in assembly language, which has the same effect as a REM in Basic. Most of the instructions are new to this series, so we had better explain them all.

LD HL and **LD BC** are instructions that load a pair of registers with a two byte number, such as an address. We have loaded HL with an address and BC with a pair of numbers to act as loop counters. We could have loaded B and C using separate instructions, but it would have used more memory.

LD A, (HL) and **LD (HL), A** are instructions for moving the contents of the memory location pointed to by the address in HL into the A register and back again. The brackets make this an indirect move, so that A is not loaded to and from the H or L registers themselves.

```
START: LD HL, &4000  ;load HL with address of start of screen memory
       LD BC, &0018  ;load B with zero and C with &18 to count loops
LOOP:  LD A, (HL)    ;load the A register from location pointed to by HL
       CPL           ;complement (or invert) the A register
       LD (HL), A    ;load A back into memory
       INC HL        ;increment HL register pair to point to next location
       DJNZ LOOP     ;decrement B, jump back to loop if not zero
       DEC C         ;decrement C
       JR NZ, LOOP   ;jump back to loop if not zero
       RET           ;return when counters are all zero
```

Figure I: An example of an assembly language routine

Hints & Tips Hints & Tips Hi &

NORMALLY you cannot do anything in the border of the Spectrum screen except change the colour. Of course with many computers you cannot even do that, but here is a tip that will let you have a colourfully striped border whenever the computer is sitting idle waiting for a key to be pressed.

Unfortunately that is the only time you'll be able to use this tip because it requires the computer to do a lot of work to a strict timetable.

In the listing below, the PAUSE 1 in line 10 synchronises the border colour changes to happen at the same time in every frame scan, making them appear to be stationary. If you leave out the PAUSE 1, you will have stripes that roll and flicker and are hard to see.

You can experiment with the program to get more or fewer stripes. If you put the PAUSE 1 as the first statement in the line you can get more stripes in, but it will be difficult to get a balanced look. Too many BORDER commands will force the timing out of sync and you will get flickering stripes.

The routine in line 10 must be kept right at the beginning of your program. Otherwise the GOTO command might slow it down too much to work properly. This is because the GOTO command always starts looking for the line it is going to at the beginning of the program, and searches through the whole listing until it finds it. Obviously it will find line number 10 before number 8620.

Line 1 is just to make a RUN command skip to the start of the main program, and line 11 is only needed to get a "hard" read of the keyboard into a variable before returning, so as not to miss reading a very short keypress. Lines 100 to 130 are for demonstration only, and have nothing to do with the stripes.

```
   1 GO TO 100
  10 BORDER 1: BORDER 2: BORDER
4: BORDER 5: BORDER 6: BORDER 7:
 PAUSE 1: IF INKEY$="" THEN GO T
O 10
  11 LET i$=INKEY$: RETURN
 100 REM main program (Your Prog
 Goes here)
 110 GO SUB 10
 120 PRINT i$
 130 GO TO 100
```

Z80

32000	33	7D00	21	21	LD HL	
32001	0	7D01	00	00	address	
32002	64	7D02	40	40	address	
32003	1	7D03	01	01	LD BC	
32004	24	7D04	18	18	counters	
32005	0	7D05	00	00	counters	
32006	126	7D06	7E	7E	LD A, (HL)	LOOP comes here
32007	47	7D07	2F	2F	CPL	
32008	119	7D08	77	77	LD (HL), A	
32009	35	7D09	23	23	INC HL	
32010	16	7D0A	10	10	DJNZ	
32011	250	7D0B	FA	FA	displacement (trust me!)	
32012	13	7D0C	0D	0D	DEC C	
32013	32	7D0D	20	20	JR NZ	
32014	247	7D0E	F7	F7	displacement	
32015	201	7D0F	C9	C9	RET	

Figure II: Our hex handling utility in use

CPL stands for complement. This instruction acts on the binary number in the A register, changing all the 1 bits into 0s, and vice versa.

INC and **DEC** increment and decrement registers, which means they add one or subtract one. These can be used on single registers or register pairs. If you subtract one from zero in machine code, you get 255. Also if you add one to 255, you get zero. When the operation is performed on a double register, the subtraction or addition is applied to the lo byte, and the hi byte is affected only by the carry, if any.

DJNZ is a rather special jump instruction. Jumps are the machine code equivalent of the GOTO in Basic, but this one does more than just jump. First it decrements the B register, then it makes the jump only if the B register became zero after the decrement (in our routine the B register starts at zero and therefore equals 255 after the first decrement).

Furthermore, it makes a relative jump. That is, it jumps back (or forward) a specified number of locations from where the instruction is, rather than jumping to a specified address. When you are writing in assembler you can just give it a label to jump to, like the label LOOP in our example, and the assembler will work out the right number to put in for the displacement as it is known, but when you write in hex you have to figure out the displacement the hard way. This is a subject we will wrestle with later.

JR NZ is another relative jump, acting on the state of the zero flag, which in our routine will have been set or reset by the result of the DEC C instruction. This jump will happen only &17 times, once every time the B register has been counted down to zero by the DJNZ instruction. This means that the DJNZ jump will take place &18 (decimal 24) multiplied by 256 times, which works out to 6144. This is the number of bytes in the screen memory, not counting the attribute area. **RET** is, of course, the return instruction that takes us back to Basic.

One of the nice things about using relative jumps is that the routine is not tied to any particular address by having specified jump addresses. The displacements work the same wherever the code is put. However there are some places where it is not safe to insert our bytes, as we found above with the system variables, so for the moment we will enter our routine at address 32000 decimal, or &7D00.

Another nice thing is that specifying a displacement requires only one byte, where an address would have needed two, so we save memory.

Run the hex handler and enter the bytes as shown in Figure II in the fourth and fifth column. Note that this listing was produced when I went back through to check that everything was all right. It won't look quite like that when you are entering the codes, as columns two and four will show all zeros. I have annotated the listing so you can see how the assembler mnemonics have translated into hex "opcodes".

Check the routine out carefully before running it. It is a lot easier to find and fix a mistake now than it will be to re-enter the whole thing if it crashes. Remember that you can enter a U to move back one address if you need to fix a mistake. When you are sure it is right quit the hex handler with Q. You can now save the code with:

```
SAVE "invert"CODE 32000,16
```

Saving the code before trying it out is merely a precaution against having to type it all in again if there is an error. Almost any error in a machine code program will cause a crash. To run the machine code program, first make sure there is something on the screen (LIST will do nicely) and then enter:

```
LET X=USR 32000
```

Shazam! A Basic program could do the same thing in about a minute, but machine code is fast. That's why we are learning it. For comparison, here is the Basic equivalent of our machine code screen inverter:

```
10 FOR i=16348 TO 22527
20 POKE i,255-PEEK i
30 NEXT i
```

The Basic version was much easier and faster to write, but it is much slower in operation and it requires more memory to store the program. These are the essential differences between Basic and machine code programming. Each method has its advantages, so it is as well to know something of both.

● ***In Part 4 we will take a more systematic look at the Z80 instruction set, those vital machine code commands.***

When things go wrong...

...the cause may turn out to be something quite simple. We help you identify the problem and guide you on the road to recovery

Aaaargh!!

YES, computers do go wrong sometimes, and the Spectrum is not an exception by any means. But hold on a minute, it might not be the computer that is to blame for a fault. In fact Sinclair's repair department says that a large proportion of the Spectrums returned under warranty have nothing at all wrong with them.

Before posting your computer off for costly and perhaps futile repair, it pays to take a few minutes to try to isolate the cause of a problem. You can often do this by using common sense without any knowledge of electronics.

The fault may be due to the failure of a peripheral device or a software bug. A cassette recorder that has faithfully loaded 1,000 times may start to give problems due to dirty, magnetised or misaligned heads. So if you have loading problems, try swapping recorders. If a different recorder improves things, then it is simple logic that the computer is not at fault. Saving and loading problems are almost always down to the cassette recorder or the tape, although the MIC and EAR sockets in the computer may fail after a long period of use.

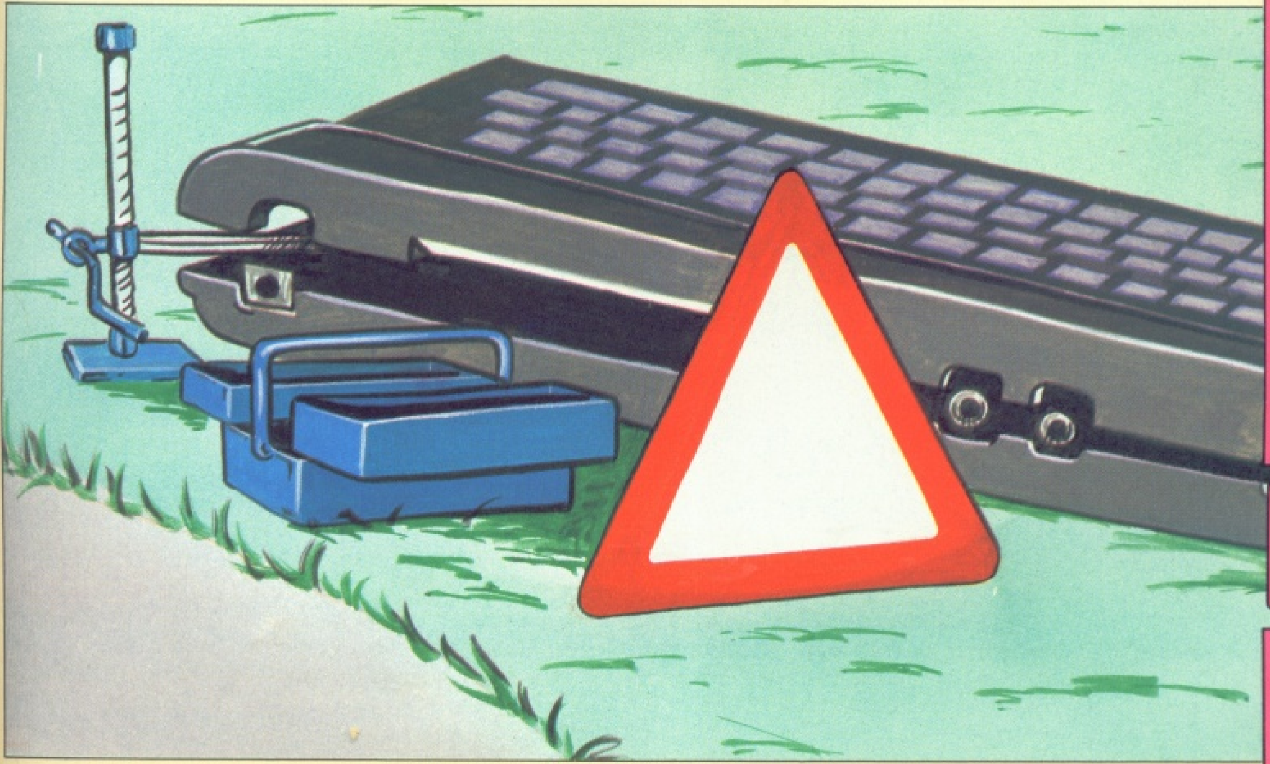
Microdrives can give problems due to wear on the fragile tape cartridges and a few bugs in the Interface I operating system ROM. There is no real cure, but the problem is minimised if you make back-up copies of everything.

Other peripherals can cause problems and they may be incompatible with each other, so that connecting two at once causes a crash. Connecting too many add-ons at the same time can also overload the power supply. A quick check for add-on trouble is to simply disconnect the add-ons and see if the computer functions all right on its own.

If a particular program causes the computer to crash, then it may be a bug, or it may be the software author's way of protecting the program from being broken into. Some adventure programs will crash the computer if you input a naughty word. ("Kill" is a four-letter word in Automata's games!)

There are a few games that won't function with certain add-ons connected, notably the Kempston joystick interface. Obviously, if you have a problem with only a few pieces of software but not with others, it is not a computer fault. Beginners to computing sometimes manage to make themselves look foolish when they fail to understand this.

It is also possible to crash the computer when writing your own programs. This often happens even to experienced programmers, although they know that it is not a computer fault, but what is sometimes called "finger trouble". The computer seldom does exactly



what you think you have programmed it to do, preferring to do what you have actually programmed it to do. Old hands expect this to happen and immediately begin looking for their own mistakes, but beginners sometimes think the computer has failed.

Real computer faults

If on power-up you get a blank black rectangle or a scattering of coloured squares instead of a copyright message, then you may have a genuine computer breakdown. Disconnect all the add-ons and try again, just to eliminate the possibility of a faulty peripheral. This symptom usually indicates a failure of the Spectrum's internal power supply circuits, and fixing it is a job for an experienced technician.

The problem can be caused by you overloading the power supply with too many add-ons – or by removing an add-on while the Spectrum is powered up – but the power supply can also fail for no apparent reason.

The power supply referred to is not the external power transformer that goes between the mains and the computer. This seldom goes wrong, although it is not unknown for the conductor to break inside the insulation near the plug on the power lead to the computer. The computer's power socket can also give trouble. As there is no on/off switch, this plug can get a lot more wear than it is designed to stand.

If the computer functions normally except that it stops responding to certain keys, then your keyboard membrane has worn out. This can be replaced fairly cheaply by any of the specialist repairers, but it may be a better idea to take the opportunity to buy one of the "real" keyboard upgrades.

Very rarely there may be a defective cell in a memory chip that will not stop the computer from seeming to work normally, but will cause crashes when programs are loaded, or "Out of Memory" error messages when there should be none. This could be confusing but fortunately there is a simple way to check for this remote possibility. The Spectrum tests every memory location in the RAM at power up, and it stores the address at which this checkout fails in the system variables at locations 23732 and 23733.

Enter PRINT PEEK 23733 as a direct command, and a 48k Spectrum should print 255, while the 16k Spectrum should print 127. Any other number indicates that the memory check has failed at a faulty chip before reaching the end of the memory map. I once found myself the owner of a rare 12k Spectrum by using this test!

Who do you turn to?

If your computer is still under warranty it will certainly be cheaper to get Sinclair Research to put it right, but it may take some time for them to fix and return it. There was a time when the shops would just swap you a new one off the shelves, but Sinclair decided that this was encouraging people to turn in machines with imaginary faults near the end of the warranty period, just to get a new computer. When a few bright sparks managed to get new computers by turning in the empty shells of their Spectrums, having installed the circuit boards in upgrade keyboards, it was the last straw.

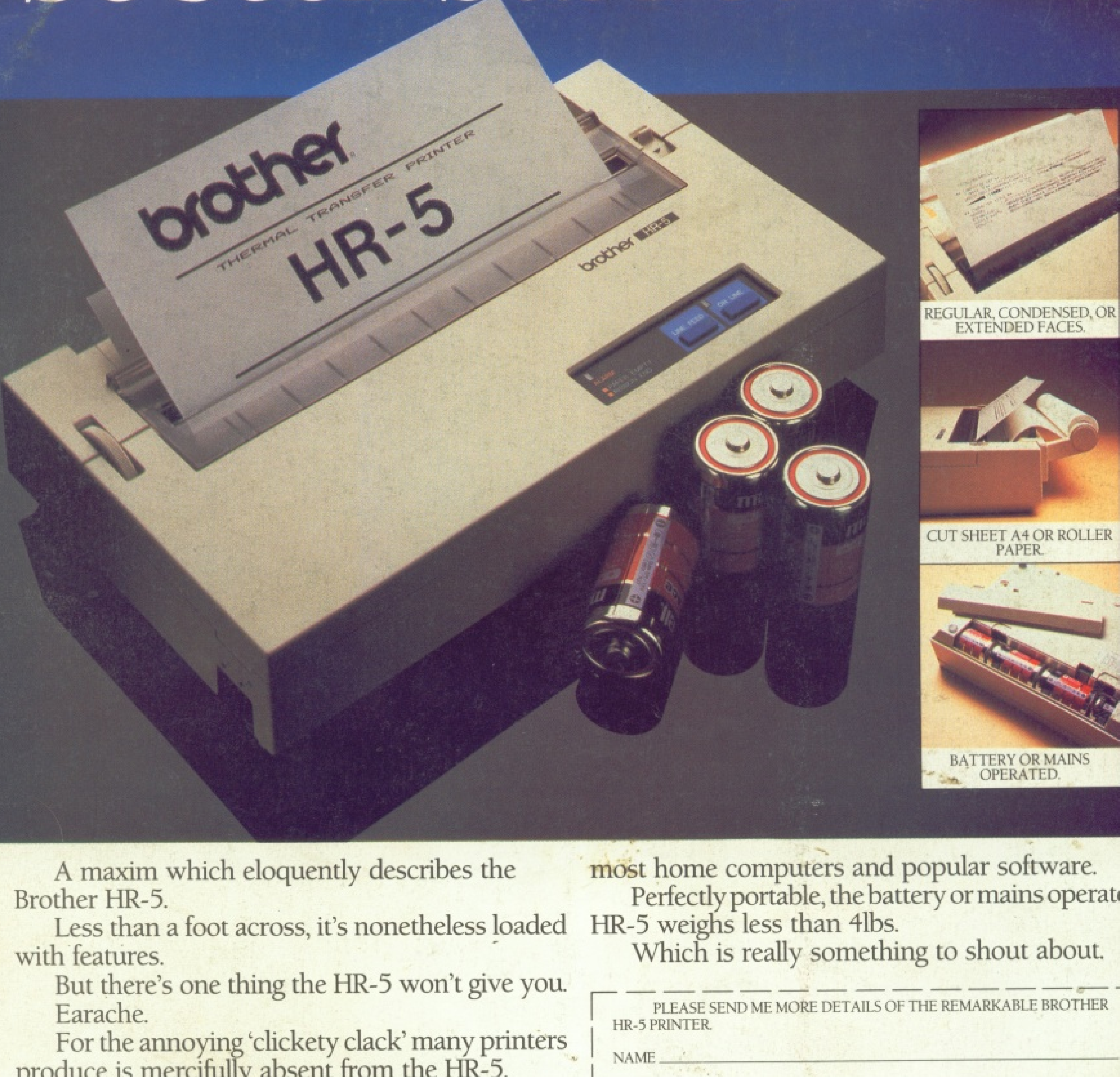
If the warranty has expired, or you are just in a tearing hurry, then you will have to go private. It costs less than you think. There are now plenty of workshops specialising in home computer repair, or even in repairing only Spectrums. Because they are familiar with the usual problems, and have worked out techniques to find the unusual faults, it will be quicker and cheaper to go to them than to a general electronics repair shop who will be more used to fixing TV sets. Home computer specialists will also have the proper Sinclair bits – like keyboard membranes – in stock, and will be prepared to fit RAM expansions or keyboard upgrades.

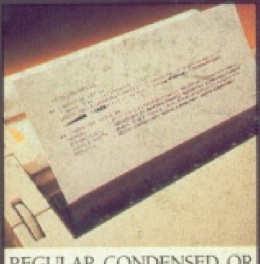
The question of which specialist repairer to go to is probably best solved by looking through the magazine adverts for one who is nearby, so that you don't have to entrust your computer to the tender mercies of the Royal Mail (twice!). Some also advertise special offers from time to time which may be of interest. Two such workshops that have been in existence for an appreciable period and promise 24 hour turnaround are:

Mancomp
Printworks Lane
Levenshulme
Manchester M19 3JP
061-224 1888 or 061-224 9888

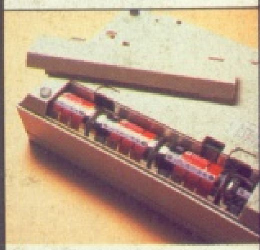
Video Vault
140 High Street West
Glossop
Derbyshire
(04574) 66555/67761

Little Brothers should be seen but not heard.




REGULAR, CONDENSED, OR EXTENDED FACES.

CUT SHEET A4 OR ROLLER PAPER.


BATTERY OR MAINS OPERATED.

A maxim which eloquently describes the Brother HR-5.

Less than a foot across, it's nonetheless loaded with features.

But there's one thing the HR-5 won't give you.

Earache.

For the annoying 'clickety clack' many printers produce is mercifully absent from the HR-5.

Quietly efficient, it delivers high definition dot matrix text over 80 columns at 30 c.p.s.

The HR-5 also has something of an artistic bent.

Being capable of producing uni-directional graph and chart images together with bi-directional text.

It will also hone down characters into a condensed face, or extend them for added emphasis.

Incorporating either a Centronics parallel or RS-232C interface, the HR-5 is compatible with most home computers and popular software.

Perfectly portable, the battery or mains operated HR-5 weighs less than 4lbs.

Which is really something to shout about.

PLEASE SEND ME MORE DETAILS OF THE REMARKABLE BROTHER HR-5 PRINTER.

NAME ____________________

ADDRESS ____________________

____

____

____________________ TEL NO. ____________________

brother

The future at your fingertips.

DEPARTMENT P, BROTHER OFFICE EQUIPMENT DIVISION, JONES + BROTHER, SHEPLEY STREET, AUDENSHAW, MANCHESTER M34 5JD. TELEPHONE: 061-330 6531. TELEX: 669092. BROTHER SHOWROOM: 83 EUSTON ROAD, LONDON NW1 TELECOM GOLD: 83: JBC002. BROTHER INDUSTRIES LIMITED, NAGOYA, JAPAN.