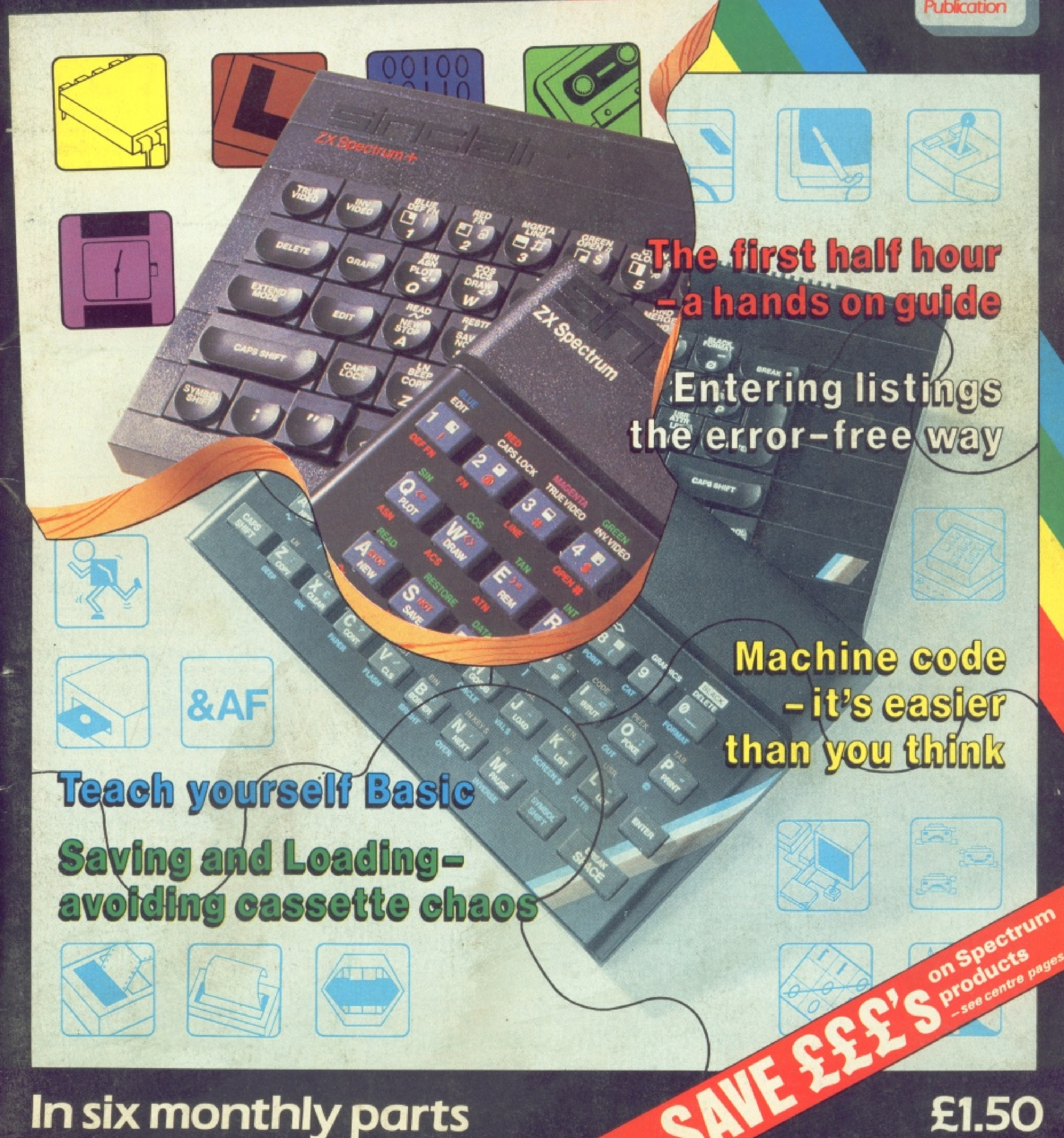
The COMPLETE

PART 1

SPECTRUM

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

A Database Publication



The first half hour – a hands on guide

Entering listings the error-free way

Machine code – it's easier than you think

Teach yourself Basic

Saving and Loading – avoiding cassette chaos

SAVE £££'S on Spectrum products – see centre pages

In six monthly parts

£1.50

KEMPSTON MICRO ELECTRONICS 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

NEW GRAND PRIX SERIES!

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

Come learn with us!

WELCOME to The Complete Spectrum, the ultimate guide to the world's most popular microcomputer. Within these page we're going to explore everything you need to know about this powerful and versatile micro.

Don't worry if you've never handled a Spectrum before, or tried to learn but been put off by the computerese. The Complete Spectrum is written in a down-to-earth, easy-to-follow style that puts the fun back into learning about micros.

So, if your Spectrum's gathering dust up there on the shelf, or if you're just tired of playing games and want to try something new, The Complete Spectrum is for you.

We'll be looking at how your Spectrum works, taking all the mystery out of the jargon computer buffs use.

And we'll show you, step by step, how to write your own programs, from the simplest Basic routines to sophisticated graphics and complex machine code.

We'll also be examining all the amazing add-ons and peripherals available for the Spectrum, showing you how to choose the most suitable and how to make the best of them.

In fact, beginner or expert, you'll find plenty to interest you among the pages of The Complete Spectrum.

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.

In Part One...

Your first 30 minutes

Ready? Then get set and go as IOLO DAVIDSON takes you through your very first session

EVERYONE who sits down with a computer for the first time experiences mixed emotions, most notably those of excitement and fear. With luck the fear will quickly dissipate, while the excitement can persist for years. However it can be very difficult to get through your very first session with any enthusiasm to spare at the end of the day, because computers are just plain complicated.

The main thing to remember is that there is no hope of getting through the ordeal without making mistakes. As you get better at computing you will make different mistakes, but you will never actually reach a point where you don't make any at all. At least, no one else ever has.

People often start out believing that they may harm the computer in some way if they key in the "wrong" command. Luckily this is just not possible. If you turn out to have a talent for confusing computers you may cause a "crash", where the computer resets itself, going back to the copyright messâge, or possibly just stops responding to the keyboard.

This doesn't hurt the computer at all, but you will lose any program that you might have typed in. The cure is to pull the 9v DC plug out and re-insert it, which resets the Spectrum, starting it up again at the copyright message. You're going to see a lot of that!

Your first command

(or try this for a lark)

Set up your Spectrum – or reset it – so that you have a blank screen with just the copyright message (either pull the 9v DC plug and reinsert it, or press the reset button on the Spectrum Plus). Find the ENTER key at the right hand side of the keyboard and give it a brief press. The copyright message disappears and you have only a flashing K on the screen. This is the cursor, and it changes from K for keyword to L for letter or E or C or G for other modes, depending on what the Spectrum is expecting you to key in next. The K cursor means that it is expecting a Basic command keyword.

Now any key you press will be interpreted as a command by the Spectrum. Briefly press the B key. As you can see, the copyright message disappears and the word BORDER is printed at the bottom of the screen. Now press the 5, and then the ENTER key. You have now given the Spectrum your first command, and turned the border of the screen cyan. If something has gone wrong go back to the point where you reset the Spectrum and try again.

Cyan is a pale blue. If your border has turned grey, it means that you need to fuss with the tuning of the TV to get good colour reception, but the computer instruction was all right.

Your first mistake

(really?)

Whenever the Spectrum has successfully performed a command, or series of commands, it will stop with the report "0 OK", followed by a number at the bottom of the screen. When it is unable to perform the command you have given it, it stops with an error report instead. Press B, for BORDER, then press the 2, the 5, and finally ENTER. You should now see at the bottom of the screen that the Spectrum has spotted the deliberate mistake. There is no colour 25, only colours from 0 (black) to 7 (white).

Now press B and then the ENTER key. At the bottom of the screen you should see the word BORDER followed by a flashing ? and a flashing L cursor. The Spectrum has spotted a different kind of mistake, a syntax error. The syntax checker is a special feature of the Spectrum that other computers don't have. It will not let you enter a command which is not complete, or which has a mistake in the "grammar" of the Basic language. Press 6 and then ENTER again. OK, the computer has forgiven you and the border should now be yellow.

Your first program

(in fact everyone's first program)

Press the 1 and then the 0 (zero, not the letter O). This gives you a line number, 10, followed by the K cursor. The Spectrum is still expecting a command keyword, but since you have put a line number first it will place the keyword into a program line instead of executing it immediately. Press the P key and you get the keyword PRINT. Now the cursor has changed to L for letter, indicating that the next keypress will be accepted in the letter mode rather than interpreted as a keyword.

The next part is tricky. You have to type in your own name, surrounded by quotation marks. On the original Spectrum with the rubber keyboard you must hold down the Symbol Shift key and press the P to get the quotation marks. On the Spectrum Plus, you press the quotation mark key (bottom row, third from the left). Then type your name or any other word, and then another quotation mark and finally press ENTER. The line should jump to the top of the screen and look like this:

```
10>PRINT "frederick"
```

If the line was not accepted it is because of a syntax error, probably to do with misplaced quotation marks, and you should reset the computer and start over.

Now press 2 and 0 (zero) and then the G key. You should now have a line number 20 followed by the keyword GO TO at the bottom of the screen. Press 1 and 0 and then ENTER. The new line 20 should now join the line 10 at the top of the screen. If not, then (you know what's coming) start over from the beginning. Practise, practise, practise!

You have now written a complete program. To see what it does, if you have not already guessed, press the R key (for RUN) and ENTER. The program will print your name 22 times and then ask you if you are ready for more with "scroll?" at the bottom. Press Y, or almost any other key, to cause 22 more names to be scrolled up the screen. When you are tired of this press N (for no) or the BREAK key or just the SPACE key to stop the program.

The Spectrum keyword entry system.

The Spectrum uses an unusual system for keying in the command keywords. You do not have to spell out the commands letter by letter, in fact you can't. People either love this or hate it, and under the circumstances it will probably be best if you decide right now that you love it.

If you are a brand new computer user, you will actually have less trouble with this facet of the Spectrum than people who have already had experience on other makes of computer. People who got used to spelling out the Basic keywords on some other machine often become infuriated by the Spectrum's helpfulness and say terrible things about it. All untrue! Most unfair!

Editing your program

It is tedious to have to reset and re-type everything every time you make one little mistake, so the Spectrum has editing functions to allow you to fix things without re-typing more than is necessary. These functions are easy to use once you have got used to them, but they seem complicated at first, so we will just try to make a small, though important, change to your program.

Having stopped the scrolling with N or SPACE, just press ENTER. This will list your program on the screen. You will notice that there is a > after one of the line numbers but not after the other. This is the EDIT cursor, and it tells you which line the EDIT function will act upon. You want to get the > into line 10, which you do by using the up or down cursor control keys. On the original Spectrum, these are the 6 and 7 keys, used in conjunction with CAPS SHIFT. On the Spectrum Plus, these are the keys with the arrows at the right of the SPACE bar.

When you have got the > into line 10, press EDIT to get the line to copy down to the bottom of the screen. The Plus has a separate key for EDIT, but on the original Spectrum you hold CAPS SHIFT and press 1. Now you must move the flashing cursor to the end of the line with the cursor right key, which is at the left hand side of the Spectrum Plus SPACE bar, or, on the original Spectrum, the 8 used with CAPS SHIFT. You can press it repeatedly, or just hold it down for a bit and let the computer make it repeat.

Now use the DELETE key (on the Plus) or CAPS SHIFT and the 0 (zero) key on the original to remove the quotation mark from the end of the line. Press SPACE (this is the unmarked space bar on the Plus) and then type in the quotation mark again, as you did before. Next type the ; (semi-colon). It's SYMBOL SHIFT and the letter O key on the original Spectrum, but has its own key on the Plus keyboard at the lower left corner. Finally press ENTER.

Now RUN the program again – R followed by ENTER – and you will see the difference that one punctuation mark can make in the Basic language. The semi-colon stops the PRINT command from starting a new line each time, and the names spread right across the screen. That was why we put in a space – to stop the names running together.

Try using the EDIT functions to change the semi-colon to a comma and see what happens when you RUN that. You should see your name written in two columns like this:

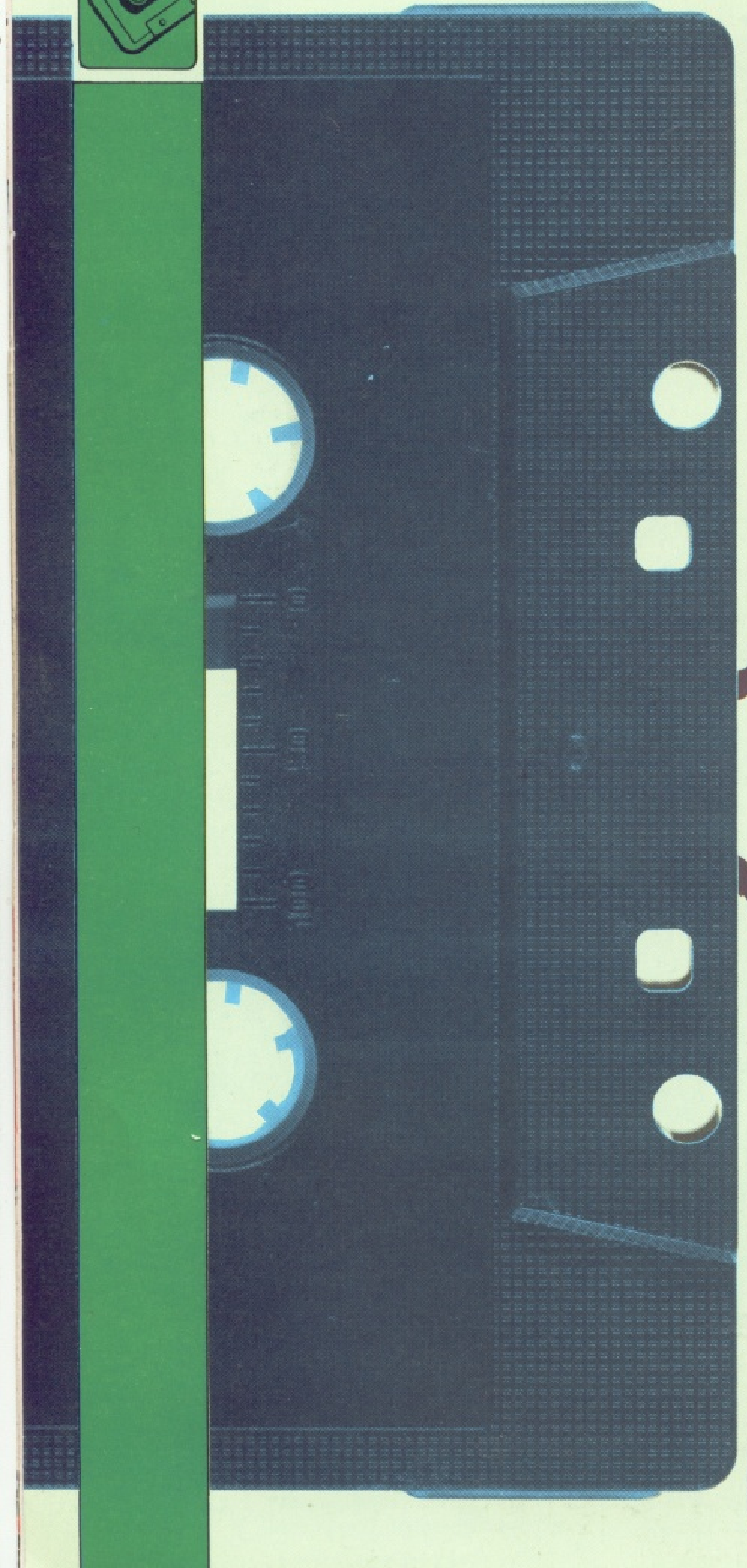
```
frederick               frederick
frederick               frederick
frederick               frederick
frederick               frederick
```
. . . and so on

So your Spectrum can print out in three different formats:

- If there is nothing after the final quotation mark the Spectrum will print anything that follows on the next line down – the same as a carriage return on a typewriter.
- If the quotation mark is followed by a semi-colon text will be printed directly after it.
- If you type a comma text immediately following it will be printed out at the next "tab" stop.

Try replacing your name in line 10 with different names, still keeping the comma at the end. As you will see, this is a very neat way of formatting your text. Hmmm, getting to be quite a programmer aren't you!

Saving and loading



Avoid chaos with your cassette by following the advice of IOLO DAVIDSON

ONCE you have set up your Spectrum and tried out a few commands, you will want to connect the cassette recorder and try loading a program from tape. For this you will need the leads and a suitable recorder, plus a cassette with a program already recorded on it, of course.

The leads are the ones supplied with your Spectrum, having two plugs at each end, one black, one grey. The cassette recorder must be the kind that has sockets for 3.5mm jack plugs, rather than a multi-pin DIN plug. Some recorders have both, which is fine. A mains powered recorder, or one with a mains adapter, is best, as the battery ones can suffer speed variations when the batteries get low. Hi-fi and stereo tape decks will usually not have the right kind of sockets nor the right level of signal output for the Spectrum, so avoid them.

We recommend that you try loading the "User Guide Companion Cassette" or "Horizons" tape that came with your computer first. These use the standard Spectrum format, rather than one of the more ticklish fast loaders used on many of the popular games tapes. Getting the cassette loading sorted out often proves a stumbling block to new users, and there is no point in complicating matters with a "turbo loader", as these are known.

Connecting Up

There are two connections that can be made between the Spectrum and the cassette recorder. One of them – joining the Ear sockets on Spectrum and recorder – is used to LOAD a program and the other – joining the Mic sockets – to SAVE a program. To LOAD we only need to connect the Ear half of the cassette leads. Decide which colour lead you are going to use for the Ear connection, and stick to it from now on.

Insert the plug of your chosen colour into the socket on the back of the Spectrum marked EAR. Insert the other plug of the same colour into the socket on the recorder marked Ear, Headphone, or External Speaker. It may be necessary to read the instruction leaflet that came with the recorder to discover which is the right socket, especially if it is marked with one of the more obscure systems of pictographic symbols, or icons as they're becoming known.

If there is no suitable socket on the recorder then you will have to get an adapter lead from your local computer or hi-fi emporium, or – preferably – use a different recorder.

Having connected the Spectrum to the cassette recorder you then need to insert the program cassette and make sure that it is wound back to the beginning, exactly like a music cassette. Don't press Play yet – first you must tell the computer to LOAD the tape. It might be best to reset the computer to clear it of any program and set it to its initial colour scheme before loading a program. To reset it, simply unplug the power supply for a moment – it's crude but effective.

You will already have noticed that the Spectrum uses a peculiar one-key system for the commands. The LOAD command is produced by pressing the J key, whereupon LOAD will appear at the bottom of the screen. You are meant to follow the LOAD with the name of the program you want to LOAD enclosed in quotation marks, but this is a big source of trouble, as the Spectrum is very literal minded. If you tell it to LOAD "Side 1" it will skip past any program it finds called "Side1" or even "side 1" and keep looking for the exact spelling you specified, paying particular attention to spaces and capital letters.

Fortunately, if you just type the quotes, with nothing in between them, the computer will just LOAD the first program it finds on the tape. Most people find it easier to keep just one program on each tape – or several copies of one program, for insurance – than to have the computer find the program they want by name from a tape with lots of different programs on it.

So, you start with the screen blank, except for a flashing K or perhaps the copyright message. Press the J key, then press the quote key twice. If you are using a Spectrum Plus this is the third from the left, bottom row. The original Spectrum has no separate quote key, instead you hold down the Symbol Shift and press the P key twice.

You should now have LOAD "" at the bottom of the screen. Note that there is no space between the quotes. Putting a space there is a common mistake that confuses beginners, because the Spectrum tries to find a program with a name consisting of one space and won't LOAD anything else.

Another common beginner's mistake is to forget to press Enter after every command. Remember, there is nothing wrong with making mistakes! Press the Enter key, and the screen goes blank. The computer is now listening to the cassette recorder. In an ideal world your program should LOAD perfectly and your only problem will be dealing with the swarms of hostile aliens or whatever your program's about. However, it isn't a perfect world so you might encounter one or two initial hiccups. If so, don't panic, just read our section on LOADing. And, once you've mastered loading, our section on Saving programs is your next logical step.

LOADing – the electronic fingerprints

Check that the cassette volume control is turned up to about three quarters of maximum. If there is a tone control, turn it all the way to maximum treble. Now press PLAY on the cassette recorder, and watch the screen. The Spectrum has a unique feature which is very helpful when using cassette loading, especially the first time. The signal from the cassette is copied to the border of the screen, so you can actually see the loading process happening, or, more to the point, see if it isn't happening. These electronic fingerprints are invaluable in solving loading problems.

The border of the screen should start changing colour, alternating red and pale blue. If the border alternates shades of grey, then you have not yet tuned the television well enough to receive in colour. This will not affect loading and can be left for later sorting out. If there is no border change after a few minutes then something is not right. Rewind, check the Ear connections, turn the volume up and try again. If there is no colour change at full volume then you have not got the right socket at the cassette.

When you get the alternating border colours, this shows that some kind of signal is coming from the recorder. The border should next turn to a pattern of red and blue stripes, indicating that the computer has found the program's leader tone – a sort of electrical overture, or heading. If you don't get this pattern within a minute or so something is still wrong. The drill is: Rewind, adjust volume and try again. You should be able to hear the tones from the cassette coming faintly from the Spectrum's Beep speaker when the volume is correct.

If you still can't get the stripes try unplugging the ear lead from the cassette and playing the tape audibly to make sure the recorder is working. You should hear tones and screeches. If you can't find a volume setting that will produce the striped border, then the recorder socket probably does not produce the correct type of output.

Once you get the striped border you know that the connections are right. The computer should find the program name at the end of the 10 second leader tone and print it on the screen. There will be a short space with no stripes, then the stripes will return, first red and blue, then dark blue and yellow as the actual program is read into the computer's memory.

Things may still go wrong if the volume isn't quite right or there is too much noise in the signal. If you can get red and blue striped borders, but no dark blue and yellow stripes, then the computer is finding the leader tones but not reading the program names from the leaders properly, and hence not starting the actual loading sequence.

The red and blue stripes only last for 10 seconds at the beginning of each load, so remember to rewind before trying again. Trying to find the leader stripes in the middle of a program is just confusing. Don't give up and stop the recorder while you actually have border stripes on the screen. Programs may take up to six or seven minutes to LOAD, and they may LOAD in more than one piece. So as long as the border is stripey, something is going on. Be patient, maybe everything is OK!

When the "User Guide" or "Horizons" programs LOAD successfully, they will run automatically and take over the job of loading other sections of the tape, telling you on the screen when it is time to stop the tape. Other programs may not run after loading, but just give an "OK" message at the bottom of the screen. If you get a message saying "Tape loading error" then rewind the tape and try again at a different volume setting. You will have to type in the LOAD "" again too.

If nothing works, time after time, try a different tape, or a different tape recorder. Most people have no trouble with cassette loading once they know what to expect, but some tape recorders will give more trouble than others. Even if you know what azimuth is – its roughly the angle between tape and tape head – do not attempt to adjust it. It may well be that it needs adjustment, but trying to do it without the right tools and a specially recorded azimuth setting tape will just make things worse. Leave such things to experts.

Once you have a successful LOAD, remember where the volume was set for next time. Unfortunately, software tapes are recorded at different volumes, so you may still have to mess about with this control when loading.

Saving your own programs

(and a pitfall to skirt)

By the time you have a program of your own that you wish to SAVE to cassette, you will probably know quite a bit about the Spectrum and be feeling relaxed and confident. Unfortunately, the Spectrum has set a little trap for you here, just to upset your calm demeanor.

The Ear socket is connected internally to the MIC socket, via resistors, in the interests of circuit economy. In fact, the Beep speaker and both Ear and Mic are all connected to the same pin on the ULA chip. This is why you can hear the cassette faintly through the Spectrum when loading from cassette.

This causes no trouble when loading, and it does mean that you can use the cassette Ear and Mic sockets to connect the Beep to an amplifier if you wish. However when it comes to saving this connection is a liability. Most cassette recorders provide a recording monitor output on their earphone socket, and when this is connected to the microphone input you get a fierce feedback loop at an inaudibly high frequency which swamps the recorder's circuits and pretty well stops anything from being recorded.

You don't have to understand about all that. Just make sure that the Ear lead is disconnected when you are saving a program and the problem will not occur. This means that you will be constantly pulling the Ear lead out and putting it back again as you go from saving to loading, and no doubt you will forget from time to time and wonder why a program won't load or even worse, why the program you saved is missing. There is nothing to be done but to try to remember each time, unless you have one of the special cassette machines made specifically for the Spectrum which avoids this problem.

Having got that out of the way, let's deal with saving programs: Connect the Mic socket on the back of the Spectrum to the Mic or Aux or auxiliary input socket on the cassette recorder, using the other half of the lead you used for the Ear connection. If there is more than one input then it is possible that any of them will work, or one may be better than another. Experiment. Remember to disconnect the Ear lead at one end. It may be better to do this at the cassette end to avoid wear on the socket of the computer, which is the more expensive of the two machines.

The SAVE keyword is on the S key, and it must be followed by a name up to 10 characters in quotation marks. Press Enter and the Spectrum says "Start tape and press any key" at the bottom of the screen. Press the play and record controls on the tape machine, and then press a key on the Spectrum to start the SAVE.

The striped border appears, this time helping you monitor the saving process. The 10 second leader tone is there partly to give a cassette recorder's automatic record level circuits time to react and settle down, so recorders using auto-level will work well with the Spectrum. If your recorder has a manual level setting you should set the level to zero – or perhaps a little above, just into the red part of the scale – during the leader tone. If you take too long to do this it might be a good idea to start the SAVE over, as a warbling leader tone may make the subsequent loading impossible. Make note of the proper setting for future use.

VERIFY – the programmer's safety belt

(or safety in numbers)

Saving a program does not erase it from the Spectrum's memory, but only copies it, so you can save it several times for safety, or you can use the VERIFY command to read the tape and compare it to the program in the memory and see if it saved properly. VERIFY is found on the R key, but it is an extended mode keyword, so learn how to use extended mode entry first.

On the original Spectrum you press Caps Shift at the same time as Symbol Shift. This should change the cursor into an inverse E, for extended mode. Then, holding down Symbol Shift once more, press the R key. VERIFY should then appear on your screen.

(This procedure will also work on the Spectrum +. However you can get to the E cursor stage directly by pressing the extended Mode key on the left of your keyboard.)

Having got VERIFY the micro needs to know what it's comparing the program in memory with. We can either specify the filename we want in quotes or, if we're willing to take the first thing the tape encounters, we can follow VERIFY with "" – a pair of quotes with nothing between. Next press ENTER – the whole idea's much the same as with LOAD.

You'll be prompted to press Play on your recorder, and the Spectrum proceeds to make the comparison. If all is well, and the program on tape matches that in memory, you'll be told. If there's a mismatch you'll also be informed. Heed the warning and reSAVE your program. Incidentally, one silly trick even the best of us perform is to SAVE a program on tape and then try to VERIFY it without rewinding, only to spend several minutes as the Spectrum listens to a blank tape.

If you keep getting VERIFY errors, it's time to think about the tapes you're using. While cheap cassette decks are perfectly acceptable to the Spectrum, use of cheap cassette tapes is not. If you cannot get tapes made for computing, then use good quality audio tape rather than something off a barrow. Even the special computing cassettes sometimes let you down, and a single such experience is usually enough to convince people of the importance of using multiple copy saving and VERIFY.

Putting it into print

PETE BIBBY starts his analysis of Spectrum Basic by explaining the various ways of using PRINT

In the beginning

In the beginning was the Spectrum. And the Spectrum just sat there doing nothing but display the copyright message. How can we get it to do something that's more interesting? The answer is to tell it to do something else – using a language it can understand.

One language the Spectrum can understand and obey is Spectrum Basic, the fundamentals of which will be described in detail in this series. By the time you've finished reading it and doing the examples you'll have a firm grasp of Spectrum Basic and be able to write some quite complicated programs. However, by then they won't seem all that complicated!

It's a practical rather than a theoretical series. In other words, it's best to read it with a Spectrum set up and ready to go. Try all the examples. You can't hurt your micro by what you enter at the keyboard and you'll learn a lot.

So don't be shy – experiment! It's the best way to learn.

WHEN it's switched on, the Spectrum puts the message:

© 1982 Sinclair Research Ltd

at the foot of the TV screen. Pressing the Enter key (the one at the bottom right of the keyboard) has the effect of removing the copyright message and leaving a flashing K at the bottom left of the screen. This flashing letter is known as a cursor and, among other things, it tells you where anything you type in at the keyboard will appear on screen. It's hard to get a K to flash on paper, so for the rest of the series it will be represented by a K with a box drawn round it thus:

$\boxed{K}$

Similarly, some of the other cursors we'll meet will have a box drawn round them.

This flashing K tells the world that the Spectrum is waiting to be told to do something. It needs an instruction. And this instruction can't be in English, it has to be in a language the computer understands. This language is Basic and the flashing K is requesting what is known as a Basic keyword.

You can look on a keyword as a word of power that magically gets the micro to perform an action. Always provided that you pick the right keyword, that is. And there are lots of Basic keywords that the Spectrum understands.

These are the words that are written out in full on top of the keys, such as LET, LIST, LOAD and GOSUB. Each tells the micro to perform a specific action. In this article, however, only one keyword will be dealt with, the keyword PRINT. You'll find that it's quite enough to be getting on with.

That first hello

PRINT is the keyword that tells the Spectrum to display something on the TV screen. Whatever is to be displayed has to follow the keyword. A little thought will show how important PRINT is. If there was no way to get the Spectrum to put messages on the TV screen we couldn't use the micro!

Let's see if we can get the Spectrum to put a word on the screen. Nearly all beginners' courses on Basic seem to have the micro saying "hello" so in time-honoured tradition we'll get the Spectrum to greet us with:

PRINT "hello"

This is what is known as a Basic instruction. It consists of the keyword PRINT followed by the message to be printed, in this case hello. When all this is entered into the Spectrum and then the Enter key is pressed, the micro immediately does what it is told and hello appears on the screen. Don't accept that on faith. Try entering the instruction for yourself and see if it works.

Start with the flashing K. This indicates that the micro wants a keyword, so give it one in the form of PRINT. PRINT begins with P so pressing the P key seems like a good idea. When you do this you'll see that PRINT immediately leaps on to the screen. You don't have to bother with the RINT bit, the computer has done it for you.

This is one of the peculiarities of the Spectrum known as single key entry. When the flashing K is showing the micro is expecting a keyword. And if the next key you press happens to have a keyword written out in full on it, then the micro knows that that's the keyword you want and writes it out in full. In this case you pressed the P key, so PRINT is what you got. For the keyword GOSUB you'd press the H key, for GOTO the G key.

By now you should have:

PRINT $\boxed{L}$

at the bottom of the screen. The flashing L means that the Spectrum is now expecting some letters, numbers or other symbols. We're aiming for:

PRINT "hello"

so get the inverted commas by pressing the Symbol Shift and P key together and then plod on with h, e, l, l, o and finally the inverted commas again.

Notice how laborious it is. When the flashing L is showing you have to type in each character in turn. Only the keywords appear all at once, and then only when the flashing K cursor is showing.

By now there should be:

PRINT "hello" $\boxed{L}$

showing at the bottom of the screen.

K
PRINT $\boxed{L}$
PRINT " $\boxed{L}$
PRINT "h $\boxed{L}$
PRINT "he $\boxed{L}$
PRINT "hel $\boxed{L}$
PRINT "hell $\boxed{L}$
PRINT "hello $\boxed{L}$
PRINT "hello" $\boxed{L}$

How PRINT "hello" is built up

The Basic keyword is there, followed by the message that's to appear on the screen, but nothing's happening. What's needed is a press of the Enter key. This tells the micro to take what it finds on the bottom line, the instruction line, and obey it. Press Enter and you'll see:

hello

appear at the top of the screen. Ignore the message at the bottom for the time being.

An important point to grasp is that the inverted commas don't appear on the screen. These are what are known as delimiters, they're just there to show the micro where the message starts and where it ends.

If the message doesn't appear, then either use the editing keys to correct the bottom line or, more brutally, unplug your Spectrum, replug and start again. The panel shows what should appear on screen at each step.

Remember, you only press Enter when the whole instruction is complete.

PRINT with strings attached

Once you've got your micro greeting you with:

hello

you might like to try getting it to produce other messages. Try making up your own and then try the ones in the panel for good measure. And when you've got the PRINT and your message on the bottom line, remember that you have to press Enter to get the computer to take note of the instruction. It tells the micro "have a look at this".

```
PRINT "HELLO"
PRINT "HELLO MUM"
PRINT "One Two Three"
PRINT "123"
PRINT "    "
PRINT "+*"
```

Experiment with these lines

As you've probably noticed by now, the PRINT command displays everything it finds between the inverted commas, or quotes as they are known in computer jargon. Be they numbers, letters or symbols, so long as they are all between one pair of quotes the PRINT puts them up on screen. It doesn't matter whether they make sense or not, the Spectrum looks at the characters in the inverted commas and treats them

Editing

TYPING errors are the bane of a programmer's life. How many times do you intend to type in something like:

PRINT "something"

and find that you've got:

PRINT psome $\boxed{L}$

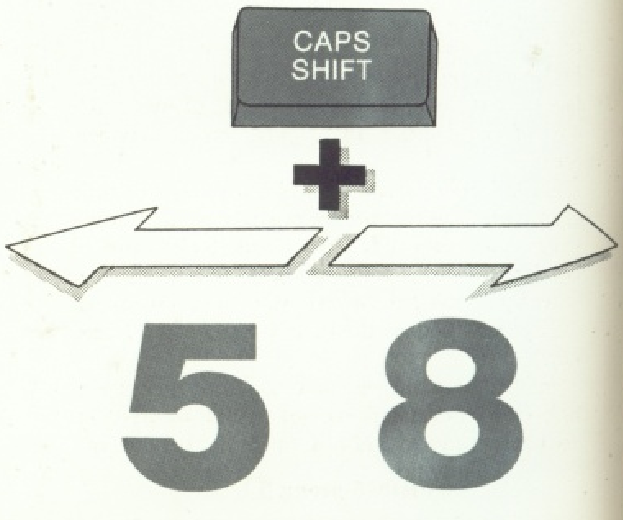
because when you tried to get the quotes you forgot to press Symbol Shift at the same time as you hit the p key? There are three main methods you can use to remedy the situation.

The first is to get rid of the line entirely and start again. One way to do this is to unplug your Spectrum and then plug it in again. Less violently you can press the Caps Shift and 1 keys at the same time and the bottom line will disappear, giving you the opportunity to type it all in again.

This seems a bit wasteful, however, as not all the line was wrong. Like the parson's egg, it was good in parts.

The second course of action open to you is to just get rid of the bad bits. You'll find that if you press the Caps Shift and 0 key at the same time, the flashing cursor moves left, deleting whatever lies in its path. So by using this method you could go from:

PRINT psome $\boxed{L}$



all as one lump, putting them on the screen.

Technically a set of characters between inverted commas is known as a string – they're individual symbols all strung together and treated as one. So what we've been doing so far is using PRINT to display a string on screen.

PRINTing expressions

Putting strings of messages on screen isn't the only use of PRINT, however. It can also be used to do sums. Try entering:

PRINT 2+2

(again, one of the time honoured beginner's examples) and pressing Enter. It will come as no surprise to you to see the number 4 appear at the top of the screen. After all, 2+2 is 4. However, take a closer look at:

PRINT 2+2

and see if you can spot the differences between this way of using PRINT and our previous ones.

For a start, there are no inverted commas. Also only numbers and a plus sign are involved. There are no letters of the alphabet or quotes. The point is that when we use PRINT followed by an expression (the posh word for a sum) then, on hitting Enter, the Spectrum works out the answer and displays it.

See for yourself, using PRINT to do your own sums. And don't just stick to additions, try multiplication and division. Bear in mind, however, that for the division sign the Spectrum uses the slash mark /, while for multiplication it uses the asterisk * rather than the small x we used at school.

While you're experimenting, have a go at the expressions in the panel. You'll notice that you can use brackets to give priority to different parts of the calculation. All this means is that the Spectrum calculates the result of the expression in brackets first, then uses this result with the rest of the expression. So, though the numbers and signs in the two sums may be the same, the positioning of the brackets means that they give us different answers.

```
PRINT 2–2
PRINT 2*2
PRINT 2/2
PRINT 1+2+3
PRINT 1*2*3
PRINT 2*(2+3)
PRINT (2*2)+3
```

Experimenting with brackets

To recap, PRINT can be used in two ways. The first displays a string on the screen, the second solves an expression (does a sum) and gives the answer. There's nothing very difficult about this. A little practice will soon clear up any lingering doubts you may have about PRINT.

You should now be able to see why you obtain different results from:

PRINT "2+2"

and

PRINT 2+2

Here we're using PRINT on two different things. One is a string "2+2", the other is an expression 2+2. The inverted commas make all the difference. With them you get a string displayed, without them the result of a sum. You can combine the two quite usefully in lines like:

PRINT "2+2=";2+2

and

PRINT 2+2;" is 2+2"

once you know about print lists and their punctuation marks.

getting rid of the e, m, o, s and p in turn leaving you with:

PRINT $\boxed{L}$

Now you can finish off typing the line.

This method also has its drawbacks. After all, only one character was wrong, yet we've erased five in all. There's a better method which involves holding down the Caps Shift key and either the 5 or the 8 key to move the flashing cursor backwards and forwards along the bottom line. Notice that the cursor doesn't erase or overwrite any of the characters, it just comes between them.

Using these cursor control keys you can shift the cursor to wherever the mistake occurs. Once there you can use the Caps Shift and 0 combination to delete the error or just insert a letter by pressing the appropriate key. (Of course the Spectrum+ has the luxury of its own Delete key.)

In the example we have to get rid of the p before the some and replace it with quotes. So starting with the flashing L cursor at the end of:

PRINT psome$\boxed{L}$

we use the Caps Shift and 5 key to move the flashing cursor over the intervening letters and leave it just to the right of p as in:

PRINT p$\boxed{L}$some

Now we can delete the p with the Caps Shift and 0 keys. We are left with:

PRINT $\boxed{L}$some

To insert the quotes we just press Symbol Shift and the P key (as we should have done in the first place) leaving:

PRINT "$\boxed{L}$some

Then to finish off the line the cursor is moved to the end with the Caps Shift and 8 keys to give:

PRINT "some$\boxed{L}$

and the rest of the letters and symbols are typed in.

Although this final method of correcting typing errors sounds a bit long-winded, you'll find that after a little practice – and you'll get a lot as you learn Basic – it's both fast and easy to use.

Print lists and punctuation marks

Nearly all of the examples of PRINT that we've seen so far have just had one item following the PRINT command. This needn't always be so, you can have several strings or expressions following the PRINT making up what is known as the print list. However these items have to be separated by one of three punctuation marks – the comma, the semicolon or the apostrophe. You can get them by Symbol Shifting keys N, O and 7 respectively. See the effect they have on the expressions and strings after PRINT in the statements in the panel.

```
PRINT “a”,“b”
PRINT “a”;“b”
PRINT “a”'“b”
PRINT 1,2
PRINT 1;2
PRINT 1'2
PRINT “a”,1;“ab”'7
PRINT
PRINT “a”“b”
```

Using punctuation in PRINT

Once you've entered these and seen the results you should have a good grasp of the effects of punctuation marks on print lists.

To be formal, the rules are:

- *The first item after a PRINT always appears at the start of a new line (unless there's some intervening punctuation marks).*
- *A comma between items in a print list means that the item following the comma is printed in one of two positions. It either appears at the sixteenth character position in from the left of the screen or, if that's been taken, at the start of the next line down on the screen.*
- *A semicolon "glues" the item following it to the item that comes before it.*
- *An apostrophe forces the Spectrum to print the next item in the print list on a new line.*

Like most things in computing, they're harder to explain than to use. Try making up your own print lists to explore the various ways you can combine PRINT with punctuation marks. You'll soon get the hang of it. And when you've finished that, try to explain the differences between:

```
PRINT “Bodger”;“Spot”
PRINT “Bodger”,“Spot”
PRINT “Bodger”'“Spot”
```

when I use them to display the names of my two cats.

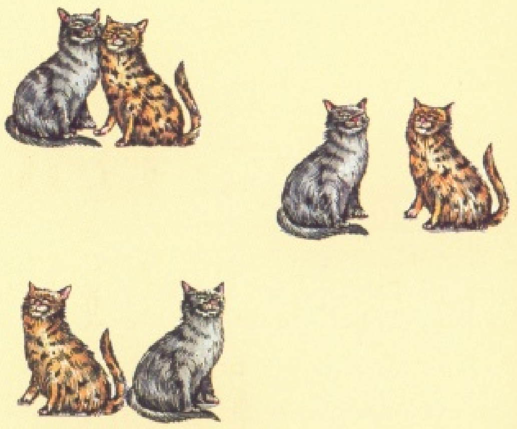
Getting into print

Following on from what's been said, can you use your micro to tell you what is half of two and two? Do you use:

PRINT (2+2)/2

or

PRINT 2/2+2



or even

PRINT 2+2/2

When you've solved that, try getting your Spectrum to display the message:

I said "Hi"

If you can manage those, PRINT should hold no fears for you.

The story so far

If we had to sum up this article in one sentence, we could say that we've seen how to use PRINT in two different ways. However, if you've tried everything out on the micro you'll have learnt a lot more than that. You'll have seen how the Spectrum expects to be talked to in Basic. English isn't good enough, as you'll see if you try to enter:

DISPLAY "Hello"

You know what you mean, the Spectrum doesn't.

You'll also have seen how important punctuation is. Think of the difference a few inverted commas can make to a PRINT statement. And, finally, you'll have learnt that, provided the Basic instruction is correct, the micro does what it's told. The trouble is that this isn't always what we wanted. The Spectrum obeys the instructions you give it, not the instructions you mean to give it. The rest of this course will endeavour to teach you how to use Basic commands to get the Spectrum to do what *you* want.

PRINT – the final problem

As we've seen, you can use PRINT with a selection of characters – numbers, letters, spaces, and so on – inside inverted commas. Alternatively you can use PRINT with just numbers, plus signs and the like without surrounding it with inverted commas. In the first case you get a message, in the second the answer to an expression. What happens if you try to use PRINT with a group of letters without the surrounding inverted commas? Try it with a line like:

PRINT number

and see for yourself. You get the message:

```
2 Variable not found,0:1
```

The Spectrum is telling us that something has gone wrong, but what is it? And what's a variable? The answers and more questions come next time.

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

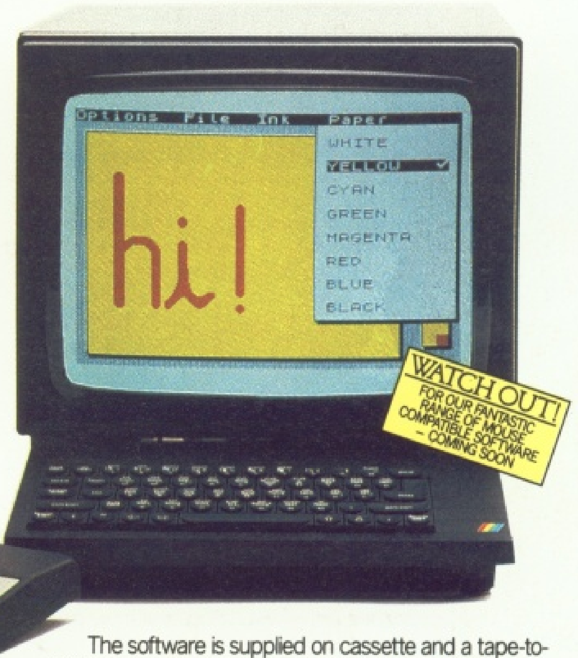
The wonderful pictures you create with AMX ART can be brought vividly to life with rich vibrant colours – using AMX COLOUR PALETTE. And with 8 Foreground and 8 Background colours you won't be short of inspiration.

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.



Options File Ink Paper
WHITE
YELLOW
CYAN
GREEN
MAGENTA
RED
BLUE
BLACK
hi!
WATCH OUT! FOR OUR FANTASTIC RANGE OF MOUSE COMPATIBLE SOFTWARE – COMING SOON

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 ________________

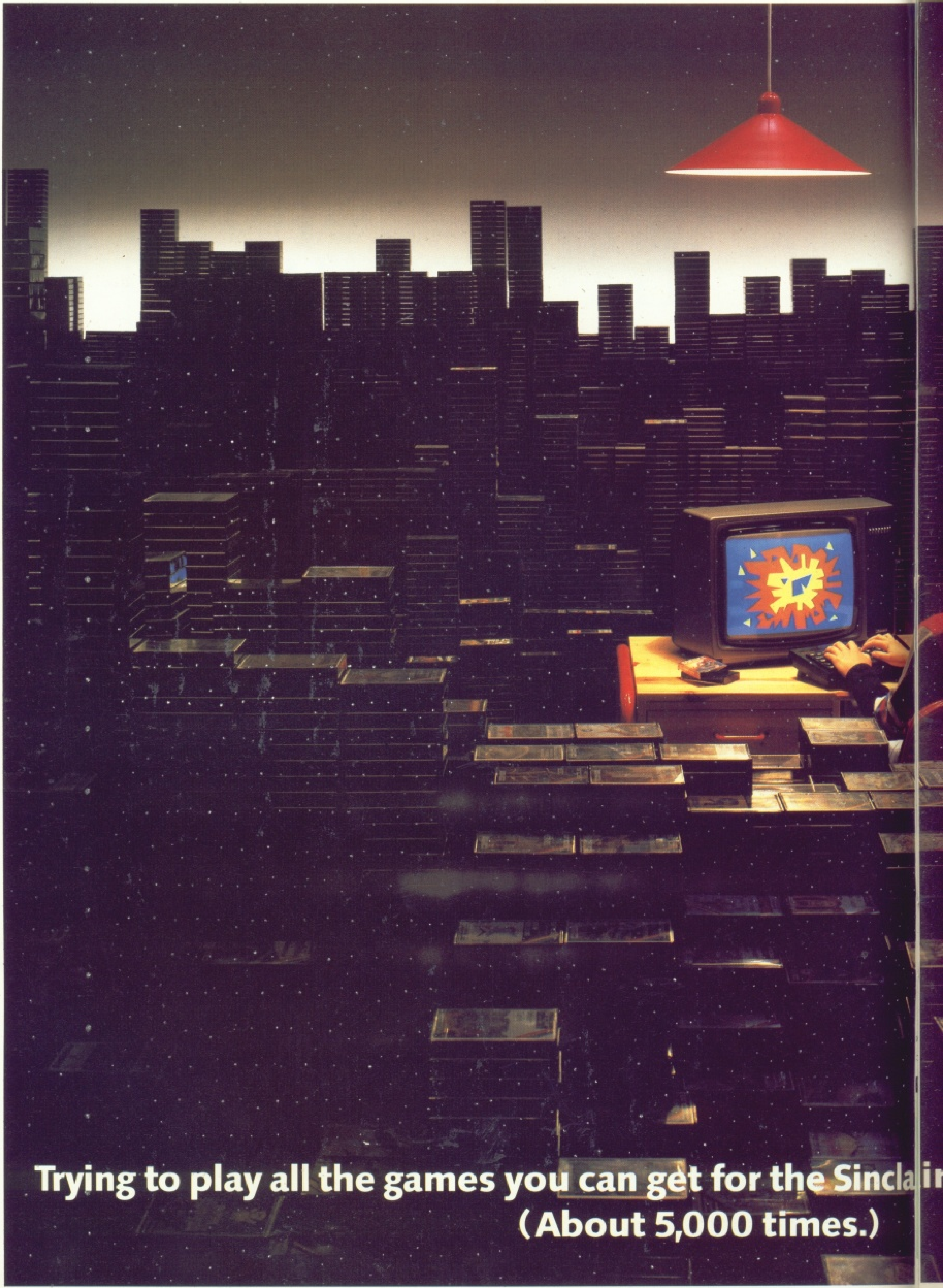
____

________ POSTCODE ________

AMX MOUSE

IMAGINATION AT YOUR FINGERTIPS

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


Trying to play all the games you can get for the Sinclair
(About 5,000 times.)

How would you like to be blasted out of the sky by anti-aircraft guns?

Or perhaps you'd prefer to be vapourised by a Thargoid starship.

Or eaten for breakfast by mutant rats.

Well, you have all this and more to look forward to, if you get a Sinclair Spectrum +.

Because there are more games available for the Spectrum + than any other home computer.

(About 5,000 the last time we counted.)

Not all of which lead to such sticky ends, we might add.

You may, for instance, rather battle it out on the chess board with a Russian grand master. Or cross swords with Nick Faldo on the golf course.

You won't just have the pick of the current titles either.

You'll get first shot at the new ones too. Because most of them become available for the Spectrum + before anything else.

And because there are so many other Spectrum owners there are more books and magazines and clubs to join.

So if you get a Spectrum +, you may well end up in a Grand Prix pile-up.

And your chances of being killed by Gremlins are very high.

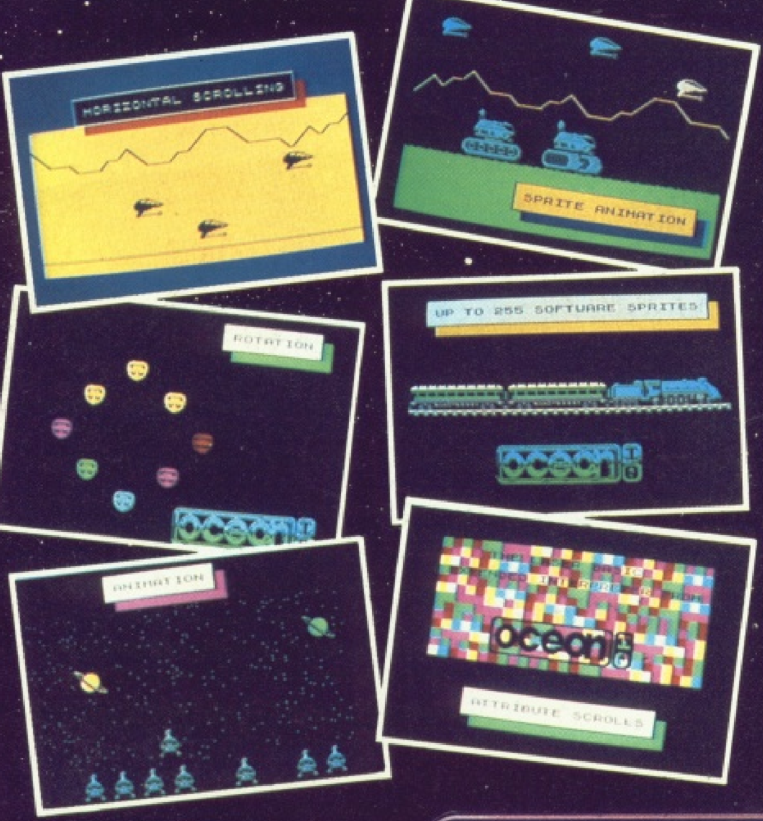
But you'll never die of boredom.

sinclair

LASER BASIC

TURN YOUR COMPUTER INTO A PROFESSIONAL GAMES WRITING MACHINE FOR FUN AND PROFIT!

LASER BASIC adds 100 new commands to Sinclair Basic. These extended commands are semi-compiling so graphic animation is extremely fast. LASER BASIC includes, extended interpreter, sprite/graphic designer, fully documented program and 2 sets of pre-defined sprites.



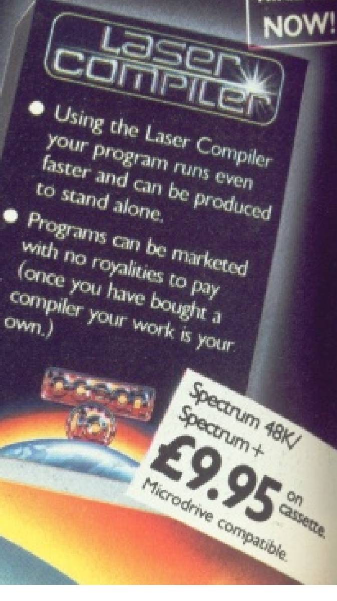
- Up to 255 software sprites can be defined, each with its own user selectable dimensions (up to 7 screens wide!)
- Operations can be carried out, on or between screen windows, sprites and sprite windows.
- Sprites can be block 'PUT' or can use one of three logical operations – AND, OR and XOR.
- Sprites and screen windows can be pixel scrolled in any direction, mirrored, enlarged, spun, inverted or cleared.
- Procedures with local variables and parameter passing.
- TRON and TROF (trace facility).
- 16 bit PEEK and POKE.
- RENUMBER and REM renumber.
- Non destructive MOVE with 2 frame animation.
- Collision detection and pattern recognition facilities.

OUT NOW FOR THE SPECTRUM 48K/ SPECTRUM+ ON CASSETTE

£14.95

Microdrive compatible.

ALSO AVAILABLE NOW!



Laser Compiler

- Using the Laser Compiler your program runs even faster and can be produced to stand alone.
- Programs can be marketed with no royalities to pay (once you have bought a compiler your work is your own.)

Spectrum 48K/ Spectrum+ **£9.95** on cassette. Microdrive compatible.

A new age dawns! The arrival of LASER BASIC: first in a powerful range of development tools for fast programming, brought to you with the combined skill and resources of Ocean and Oasis – (producers of "White Lightning".) Other, easy to use products in this expanding range will include screen artist/designers, music composers and machine code emulators to make your programming more rewarding and exciting.

INTERACTIVE *SOFTWARE*

Ocean iQ 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.

6 Central Street, Manchester M2 5NS. Telephone: 061-832 6633. Telex: 669977 Oceans G.

Extended Mode

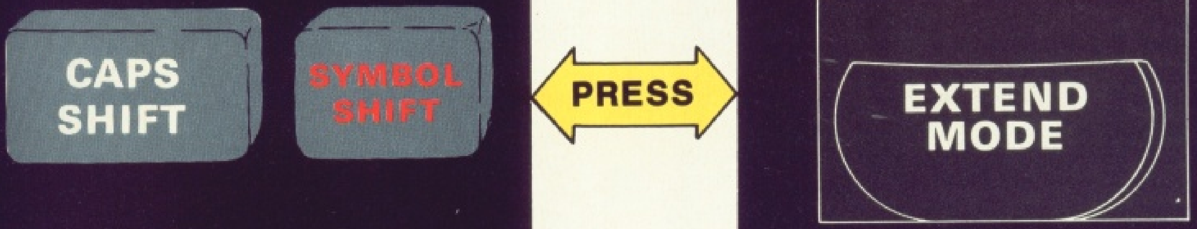
How to harness those "hidden" keywords

THERE are many more keywords in Spectrum Basic than there are keys on the keyboard, so something had to be done. A few of them were treated as if they were symbols, and can be called up by using the Symbol Shift key in conjunction with a letter key, but there were a lot left over. So the Spectrum's designers invented a sort of super-shift called Extended Mode, which allows each key to be used for two more keywords.

Certain soreheads have pointed out that you can end up pressing more keys to get an Extended Mode keyword than you would have to press if you spelled it out letter by letter. Ignore them, they are just upset because they keep mis-spelling the keywords on their Orics and don't get any error message until they try to RUN the program.

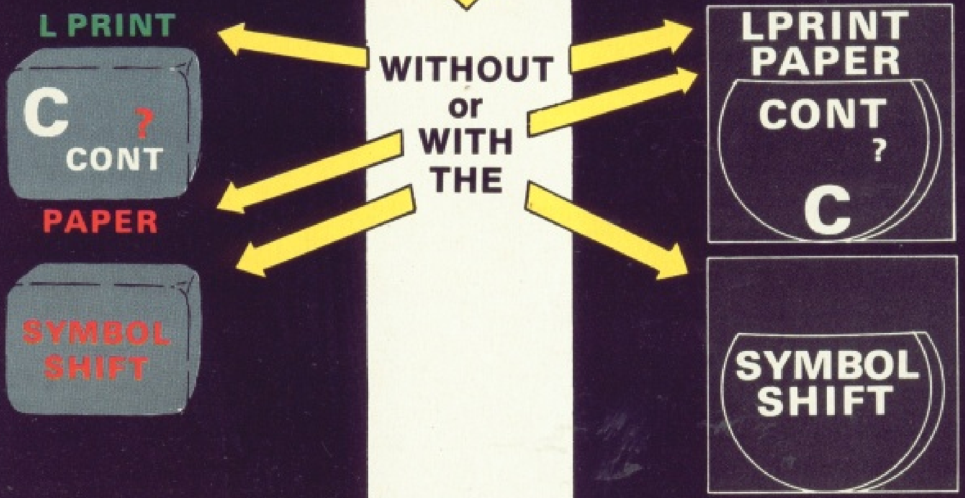
ZX Spectrum

Spectrum Plus



On the original Spectrum you must press the Caps Shift and Symbol Shift together to get into Extended Mode, at which point the cursor changes to a flashing E to indicate that the computer is expecting an Extended Mode keyword. Most keys have two extended mode keywords on them, so you choose between them with the Symbol Shift key. Remember, if the keyword is printed in red, like the Symbol Shift key, then hold Symbol Shift while you press the key for the Extended Mode keyword.

On the Spectrum Plus you must press the Extend Mode key to get into Extended Mode. When you are in it the cursor changes to a flashing E to indicate that the computer is expecting an Extended Mode keyword. Most keys have two extended mode keywords on them, so you choose between them with the Symbol Shift key. Remember, if the keyword is in the lower position, then hold down the Symbol Shift while you press the key for the Extended Mode keyword.



A mis-user's guide to listings

Don't spend hours looking for the faults in other people's listings – cast out the typing errors that bedevil yours . . . PETE BIBBY provides a few pointers to help you avoid the pitfalls

ON the face of it, typing in listings should be quite straightforward. You just type what's on the paper into the Spectrum and there you are, one working program. This, however, is sometimes easier said than done. If you're not careful or are just plain unlucky, typing in listings can be an extremely frustrating experience for you and for anyone in the vicinity.

All programmers, if they're honest, will admit to having had the experience of typing something in and finding that it doesn't work. And no matter how hard they try, the fault is untraceable. If you haven't had this happen to you yet, touch wood, quickly. Or become more honest!

It's so annoying. You know that the syntax is correct, otherwise the line would never have got off the bottom line of the screen into the listing. Eventually you give up in disgust and say that the listing is wrong – usually in fairly strong language.

To be fair, though this can be the case, it very rarely is. Occasionally a part of a listing won't print out properly, or a space will appear where there wasn't one in the first place. It's even been known for part of a listing to "fall off" a page. However this doesn't happen all that often – though once is one time too many for the poor, raw-fingered typist. The sad fact is that if the program won't work, you've almost certainly made an error typing the listing into your micro.

These may be harsh words but they're true. If you've typed in a listing and it doesn't work, the odds are that you haven't typed in the listing correctly. Accept the fact that you're not the world's greatest typist and start checking yet again. Of course, the best idea would be to stop the errors creeping in in the first place, but that's the counsel of perfection.

The first piece of advice to bear in mind when faced with a listing that doesn't work is *don't get carried away*. It's all too easy to become obsessed with a listing that's gone wrong, struggling with it for hour after hour, eventually losing your temper. If you don't spot the mistake in the first quarter of an hour then give up for a little while. Go and have a cup of coffee or something. See if the family still recognise you.

It's surprising how often the mistake becomes obvious when you return to the keyboard after a break. And if you still can't find the mistake, here's a selection of the ones I make all the time. Odds on you'll find the answer to your listing problem here. So for those of us who are less than perfect, here are five of the ways I've managed to make nonsense out of a correct listing.

Miscopying a listing

BELIEVE it or not – and you probably won't after you've been struggling away for hours – the majority of problems are caused by miscopying a listing. One of the daftest mistakes you can make is to leave a complete line out. Of course the program won't work properly if it isn't all there, but lots of times we expect it to.

Sometimes the missing line will contain something vital so the program will grind to a halt and we'll be made aware of the problem. However, some lines if left out don't crash the program but leave it working, but working badly. And if the line you've left out is only used once in a blue moon, that is, it's something like:

```
IF bluemoon THEN do something
```

your program may go off the rails only once in a blue moon. And that's really hard to figure out. Try leaving out a line from:

```
10 LET cat=3
20 LET dog=3
30 LET both=cat+dog
40 PRINT both
```

and see the results. Notice that leaving out line 40 doesn't crash the program, but it does affect its results.

Often when we've left out a line, we've just skipped it by accident as we read the listing.

Another way of leaving one out is where, instead of typing in two lines, 20 and 30, we type in the first line correctly, numbered as line 20 and then number the next one, which should be 30, as 20. The second line overwrites the first and so we have a missing line as in:

```
10 LET cat=3
20 LET both=cat+dog
40 PRINT both
```

To make it worse, the line numbered 20 is actually the one that should be numbered 30. It's amazing how difficult this can be to spot in a long listing, especially one where the line numbers are in an irregular order.

And the trouble is that the Spectrum can't tell you that "line 50 is missing" because it doesn't know that it should be there. So any error message that it comes up with isn't all that helpful, often pointing to some other part of the program that depended on the missing line. If you try running the last program you get the message!

2 Variable not found, 20:1

It's obvious that the variable that's missing is *dog*, which should have been set up by the proper line 20. The program crashes, so you get an error message, but it won't tell you that there's a line missing. You have to realise that as you hunt back through the listing and find that there is no LET *dog* anywhere.

Then again you might assign a value wrongly. Suppose in the original program you typed in line 10 as:

20 LET dog=2

In other words, there's a 2 where there should have been a 3. The program still works, but incorrectly. It gives you the wrong result. The error may be easy to spot in this example, but in a long program with lots of lines it's not so simple.

I've found that it makes life a lot easier if I enter a long listing in three or four sessions rather than one, saving it to tape between times. Incidentally, I recommend that you SAVE your program every 20 or 30 lines. This is in case you lose your listings by an accident such as a power cut or a meddlesome younger brother. You might lose all your work from the Spectrum's memory but you will have the best part of it on tape.

The moral is that when you're typing in a listing, type in all of it. And type it in exactly. There's no point in changing the programmer's code and then feeling aggrieved when it doesn't work.

Yet people do this, correcting "mistakes" that they find in listings and then wondering why the program grinds to a halt 20 lines later . . .

Misspelling

A CLOSE relative of miscopying a listing is misspelling. Watch your spelling because the Spectrum is very sensitive about such things. Misspell a variable and the micro can easily get confused, as can you when you try to sort it out! Suppose that you tried to get the cat and dog program running again, but made a mistake in line 40:

```
10 LET cat=3
20 LET dog=3
30 LET both=cat+dog
40 PRINT bath
```

The micro works through the program, comes across *bath* in line 40, can't find any reference to it anywhere else in the program. Hence it crashes with the message:

2 Variable not found, 40:1

In this case it's immediately apparent that there's been an error, but there can be times when the error is harder to spot. This is when a program uses variable names that are fairly similar such as *TIM* and *TOM*. If you hit O instead of I you end up with *TOM* instead of the intended *TIM*.

And there'll be no error message as the program already has a variable *TOM*.

Confusing letters

CONFUSING similar letters is a classic error that I'm sure everyone has made at one time or another. This is the error caused by mistaking a number for a letter and vice versa. Let's see the effect this can have on a program such as:

```
10 LET LOOP=20
20 FOR l=10 TO LOOP
30 PRINT "A message"
40 NEXT l
```

which gives us 11 messages on screen. Probably the most common mistake, and the hardest to detect, is confusion between the numeral "0" and the letter "o" or "O". On poor quality listings these look very much the same, and even with clear listings it's all too easy to press one key in mistake for the other. Even with a slash across the 0 this can happen.

The Spectrum won't like it and the program will almost certainly grind to a halt. Suppose you made this mistake and line 20 became:

```
20 FOR l=10 TO L0OP
```

When you try to run the program it crashes with a:

```
2 Variable not found, 20:1
```

The same kind of confusion can arise between the lower case letter "l" and the number 1. They look fairly similar and on typewriters they are often the same key, but not on the Spectrum. Confuse the two and you're asking for trouble, trouble that's very difficult to spot and sort out. Can you see why:

```
20 FOR l= l0 TO LOOP
```

brings the message:

```
2 Variable not found, 20:1
```

when the program stops? Look at the character after the equals sign of line 20.

Another pair of lookalikes to be wary of are the minus sign "–" and the underline "_" which are often confused. You can't do a subtraction with the underline sign, though it's amazing how many times you try!

Ghosts

A COMMON pair or errors involve ghosts. The first is to introduce a "ghost" line into the program when you're typing it in. This happens when what was supposed to be:

```
250 PRINT "Boo"
```

is entered as:

```
25 PRINT "Boo"
```

It's all too easy to miss the fact that you've entered a line numbered 25 instead of 250. Now not only have you got a line in the wrong place, you've also got one missing. Two mistakes for the price of one! And if there happened to be a line 25 already, you've just overwritten it so there's a possibility of a three-way error.

So if you've typed in a listing and you notice a gap halfway through the line numbers that shouldn't be there have a look for its clones and think about what they might have obliterated.

The second ghostly error is really stupid but, sadly, all too easy to make. It's done by forgetting that you're already haunted by a program in the Spectrum's memory. You then type in your listing and, if the line numbers of the two programs don't match, they get mixed up. The last line of:

```
10 LET indians=8
20 LET cowboys=4
25 IF door=open THEN PRINT "Enter at
your peril"
```

is obviously left over from an earlier program and shouldn't be there at all. The program won't know anything about *door* and *open* and so will come to an untimely end.

Wrong punctuation

JUST as in ordinary writing, the dots and squiggles of punctuation matter to the Spectrum. Unfortunately it's all too easy to confuse the fullstop ".", the semicolon ";", the colon ":", and the comma ",". These may look very similar on listings and mistakes are easily made. This can cause all sorts of problems, from having displays in the wrong place or in the wrong colour to having the program crash without an error message.

One common mistake is to put commas in numbers. You may try to input 20,000 but the Spectrum won't like it if you type it in. The line will just sit there at the bottom of the screen, sullenly flashing a question mark at you. The micro prefers the number in the form 20000 – that is, without the comma.

Another problem caused by mixed-up punctuation marks comes in data lists, where the items are, or should be, separated by commas. Even a simple program like:

```
10 FOR L=1 TO 5
20 READ number
30 PRINT number,
40 NEXT L
50 DATA 3,3.5,4,4,5
```

can be devasted if the commas in the data line are not exactly right. The output from the program is:

```
3          3.5
4          4
5
```

Suppose, however, what should be:

```
50 DATA 3,3.5,4,4,5
```

is typed in as:

```
50 DATA 3,3,5,4,4,5
```

The decimal point in 3.5 has become a comma. Now the output is:

```
3          3
5          4
4
```

Five items are read from the six in the data list, but because of the inadvertant comma the data is wrong.

This will result in programs working strangely, if at all.

What's particularly annoying is that the Spectrum won't tell you that there is now one item left over in the list. It only displays an error message when it tries to READ from a list with too few items, as you find out if you alter line 50 to:

```
50 DATA 3,3.5,4.4,5
```

This results in:

```
3          3.5
4.4        5

E Out of DATA, 20:1
```

appearing on the screen.

These errors can be very tricky to sort out. If you get some weird happenings in a program it's always a good idea to check that the DATA lines are right.

And while we're on the subject of correct punctuation, see what happens if you use a semi-colon or apostrophe instead of the comma at the end of line 30.

The moral is to be very careful with punctuation marks in listings. They may not mean a lot to you but they do to the micro. Get one wrong and it can be the devil of a job to find the mistake and remedy it.

Now that you've found your mistakes . . .

ONE point to bear in mind is that when you do find a mistake in a line, don't think that it's the only one. If you made an error then your concentration was lacking at that point and there may be another one lurking about on the same line or thereabouts. It's amazing how often they come in pairs.

And beware of making another mistake when you "correct" a line. It's very easy to introduce another error as you correct the first one.

All in all, it just comes down to looking carefully at what you're typing, both on the page and on the screen. If you spend all your time peering at the keyboard – as most beginner typists do – and don't look at the screen to check your work, mistakes will abound.

This seems obvious, yet it's very easy to become hypnotised by the keys, typing rapidly but not checking what you type.

So pay attention to what you're doing when you type in listings. A little time spent checking as you actually enter the lines saves a lot of time later.

Talking tech...

Computer literates are often charged with deliberate obfuscation (see, he did it again!) because they use language that others do not understand when they are talking about computers. "Why can't you use plain English?" they say, and "You're just trying to sound clever".

Well, you can use plain English to talk about computers, but it takes longer. We prefer to call a spade a spade, rather than the equivalent in simple terms: "Steel tool with a wooden handle used for digging holes in the earth". You might as well try to get a musician to stop going on about tempo, or an artist to drop all references to umber.

Anyhow, once you know what all the terms mean, you too will be able to baffle the uninitiated with your brilliance, so here goes . . .

What is a computer, anyway?

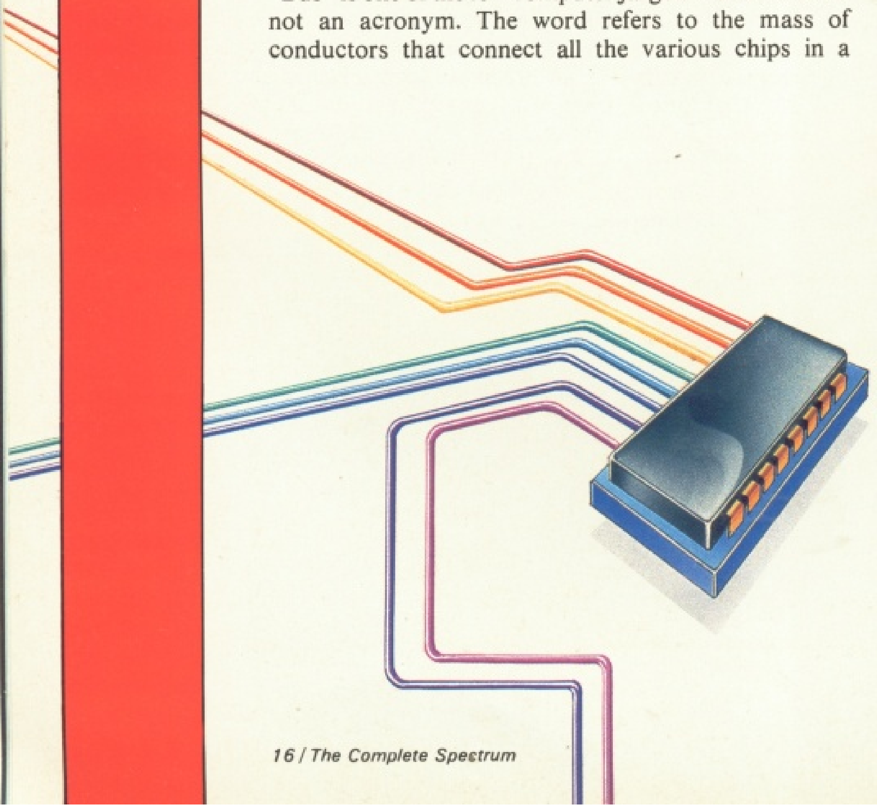
A computer is a machine for manipulating information, far too complicated for it to be made with moving parts, but quite without any intelligence or understanding. In fact the smartest chip in the Spectrum doesn't have enough brainpower to run a budgerigar. In the proper hands such a tool can be very useful, but all the cleverness has to come from the human being who is using the thing.

The construction and the capability of different computers is extremely varied, but the building blocks of computer architecture are functionally similar whether it is a micro or a mainframe.

The computer bus

"Bus" is one of the few computer jargon words which is not an acronym. The word refers to the mass of conductors that connect all the various chips in a computer together, but the bus is not just a lot of wires. It is an organised system which can be logically extended and added to.

On the Spectrum the bus is available at the edge connector, so that other pieces of equipment may be joined to the computer and become a part of it. Communication between the various parts of the computer is what the bus is all about. Sometimes one particular group of bus lines are referred to as a separate bus, so you have the data bus or the address bus.



Read Only Memory

The Read Only Memory, or ROM, is the part of the computer's memory that never forgets or changes its mind, not even when the power is off. The program in the Spectrum which comes alive and prints the copyright message when you power-up lives in a ROM. So does the Spectrum's Basic language and all the other built in commands. This is called the system software, and it has to be in a ROM, or at least the bits that load software from tape have to be. Otherwise you wouldn't be able to load anything in until you had loaded in the bits that do the loading, which you wouldn't be able to load in until . . .

You cannot program the ROM or change its contents at all. The contents of the read only memory chip were actually manufactured into it.

Random Access Memory

RAM is used for storing programs and data that change frequently. The computer can read and write to this type of memory, and that is indeed what the computer spends most of its time doing. The Spectrum ROM is all in one chip, but the RAM occupies a whole bank of chips. As soon as the power is turned off, the RAM forgets everything it ever knew.

Central Processor Unit

The CPU in the Spectrum is a Z80 microprocessor. This is the chip that runs everything and makes all the decisions. It can perform one of its simpler machine code instructions in about one millionth of a second. In that time a beam of light moves about three hundred metres, and nothing else on earth moves more than a millimetre or so. However machine code instructions don't do very much per instruction, and hundreds of them are required to carry out the simplest Basic commands.

When the computer is turned on, the CPU collects its first instruction from memory location 0 (in the ROM) and performs that instruction. Then it goes to memory location 1, and performs the instruction it finds there, then to location 2 and so on. When you think the computer is doing nothing, it is actually whirling madly around a loop – checking to see if you have pressed a key yet, at a million instructions a second.

Uncommitted Logic Array

The Uncommitted Logic Array (ULA) is the chip that made cheap home computers possible – once all the other chips were invented, that is. This chip is full of electronic building blocks that can be customised during manufacture to provide the sort of link-up of the internal modules that the computer manufacturer thinks would be most useful. The computer then gets a custom chip for little more than the cost of a standard off-the-shelf item.

The functions performed by the Spectrum ULA have to do with the input and output from the computer. The biggest job is converting the contents of the screen memory into a colour video signal. This is so important that the ULA is given priority over the CPU when it wants to look in the screen memory (part of the RAM). Other in/out tasks include the address decoding for the keyboard and cassette ports. When the CPU wants to get data from one of these ports, the ULA makes sure that the right one gets connected.

On other computers the jobs performed by the ULA would be done by a video chip, a parallel port control chip, a serial port control chip, a keyboard decoding chip, and maybe a few others. These extra chips would be referred to as interfaces, so the job of the ULA is interfacing.

Visual Display Unit

The Visual Display Unit (VDU) is where the computer displays the results of its deliberations. In very old computers a printer was used for this function, but the term VDU really only applies to video displays, and a printer is only used now when a "hard copy" is required.

In home computing the display unit is usually the family television, but the term VDU also refers to the part of the computer's circuitry that provides the video signal.

The Spectrum's TV modulator changes the video signal produced by the ULA into the Ultra High Frequency (UHF) television signal. This is an item that would not be found in all computers. Business machines in particular tend to use video monitors, for which a video signal is enough.

Keyboard

This is the human interface, where the computer gets much of its data and all of its instructions. All computers have at least one, but some computers are able to handle many more than one at a time.

The Spectrum keyboard is nothing to type home about, but it is sufficient for most purposes. If you are planning to do word processing, you might want to replace it with one of the better add-ons.

Storage

The programs and data that reside in the RAM when the computer is turned on have to be stored somewhere for future use or they will simply evaporate when the computer is turned off. Usually they are stored on magnetic material, a rather fancy name for the stuff coating cassette tapes, which you can record on and play back from. Discs and microdrives use the same idea.

The form of this storage, the speed of access, and the amount available to the computer at any one time varies enormously as you look from home computer to the big business systems.

Power Supply Unit

The Power Supply Unit (PSU) is an important part of any computer, and it can be a very complicated piece of circuitry. The power supply for a bank's computer, for instance, is designed to keep supplying emergency power even after a mains failure.

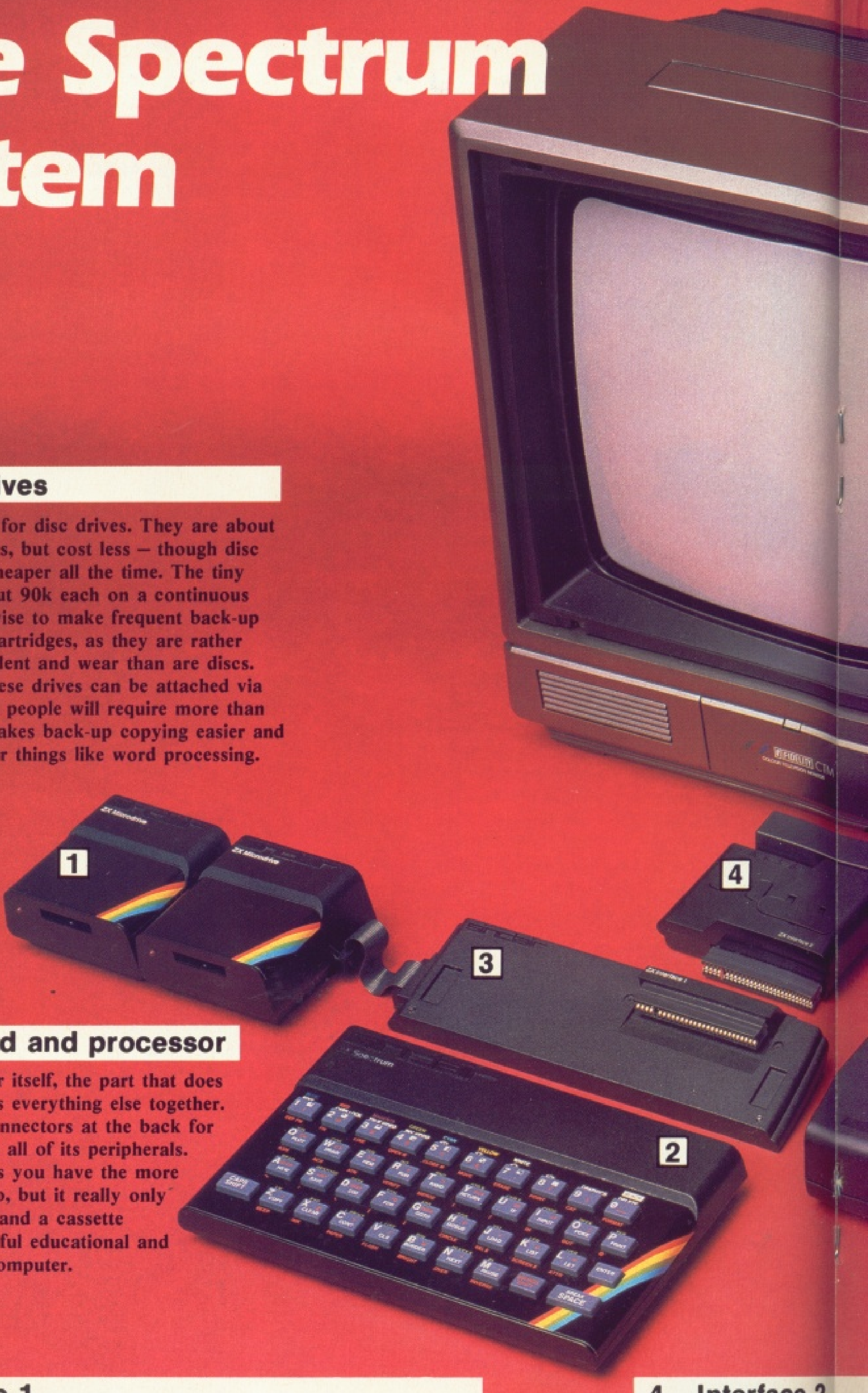
Every chip in the Spectrum needs a five volt power supply, and some need more than one voltage. The Spectrum has a transistor oscillator circuit with a tiny transformer that provides several different voltages from the nine volts fed into it by the external power supply unit. It is this oscillator that makes the little whine or whistle that you can hear when you turn the computer on.

Software

The most intelligent thing about a computer is the program fed into it. Without sensible instructions, a microprocessor is just an expensive form of purified sand. Computer programs may become truly intelligent one day, but the computer itself is just a place where programs live. If you like, the relationship between software and computer can be considered as similar to that between mind and the physical brain.

Keyboard: The human interface

The Spectrum system



1 Microdrives

Sinclair's substitute for disc drives. They are about as fast as most discs, but cost less – though disc drives are getting cheaper all the time. The tiny cartridges hold about 90k each on a continuous loop of tape. It is wise to make frequent back-up copies of all your cartridges, as they are rather more prone to accident and wear than are discs.

Up to eight of these drives can be attached via Interface 1, but few people will require more than two. Having two makes back-up copying easier and gives more space for things like word processing.

2 Keyboard and processor

This is the computer itself, the part that does all the work and ties everything else together. It has a range of connectors at the back for communicating with all of its peripherals.

The more add-ons you have the more the computer can do, but it really only needs the television and a cassette recorder to be a useful educational and entertaining home computer.

3 Interface 1

Used primarily for connecting Microdrives to the Spectrum, but it also provides a serial RS232 interface for driving a "proper" printer or communicating with any other computer that has a reasonably standard RS232 interface. In addition, there are two sockets for setting up a network for local communication between a number of Spectrums.

This interface, like most Spectrum peripherals, connects to the Spectrum expansion bus by way of the edge connector in the slot in the back of the Spectrum. With so many add-ons wanting to use the edge connector, it is an important point that Interface 1 carries this connector through for further items to be attached.

4 Interface 2

Primarily used for games, pr
ports and a slot for ROM ca
which there is little to be had
use Sinclair joysticks as well
other types, but there aren't r
Interface 2 joystick ports. Ar
can be used with this interfac
bought separately. Enough of
carried through so that the Z
connected, but nothing else.

5 Television set

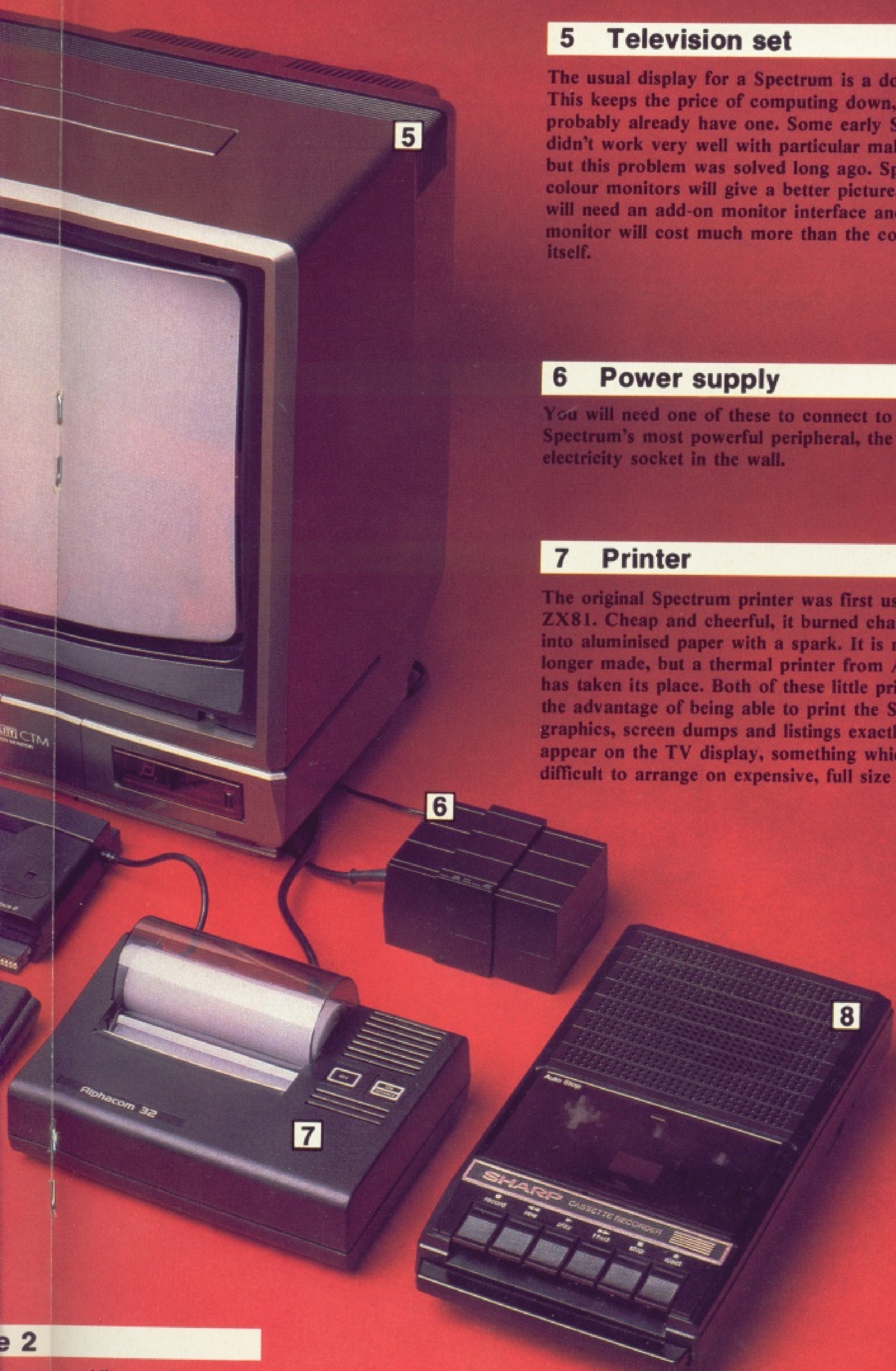
The usual display for a Spectrum is a domestic TV. This keeps the price of computing down, since you probably already have one. Some early Spectrums didn't work very well with particular makes of TV, but this problem was solved long ago. Specialised colour monitors will give a better picture, but you will need an add-on monitor interface and the monitor will cost much more than the computer itself.

6 Power supply

You will need one of these to connect to the Spectrum's most powerful peripheral, the electricity socket in the wall.

7 Printer

The original Spectrum printer was first used on the ZX81. Cheap and cheerful, it burned characters into aluminised paper with a spark. It is now no longer made, but a thermal printer from Alphacom has taken its place. Both of these little printers have the advantage of being able to print the Spectrum's graphics, screen dumps and listings exactly as they appear on the TV display, something which can be difficult to arrange on expensive, full size printers.



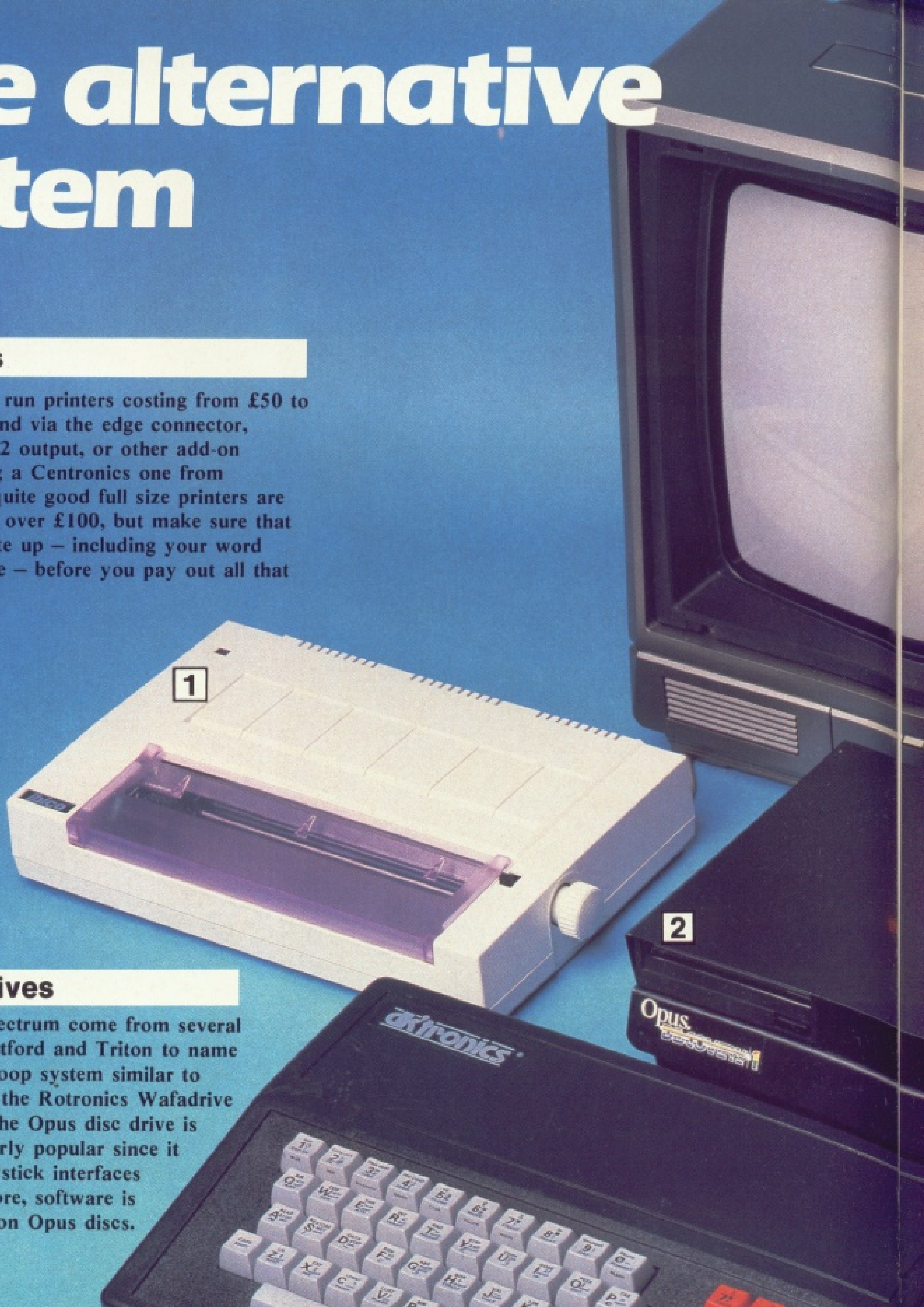
e 2

games, providing two joystick
ROM cartridge software, of
to be had. Almost all games will
s as well as Kempston and
re aren't many that use both the
ports. Any Atari type joysticks
is interface, but they must be
Enough of the edge connector is
hat the ZX printer can be
ing else.

8 Cassette

Cassette tape is the primary Spectrum program storage medium. You will need a tape machine even if you have Microdrives, simply because most software is supplied on tape. Some games software is protected against copying, which will also make transferring it to Microdrive cartridge difficult, so the tape deck stays. You don't need an expensive one, but it should have 3.5mm jacks instead of DIN plugs, and preferably a mains power supply or adapter.

The alternative system



1 Printers

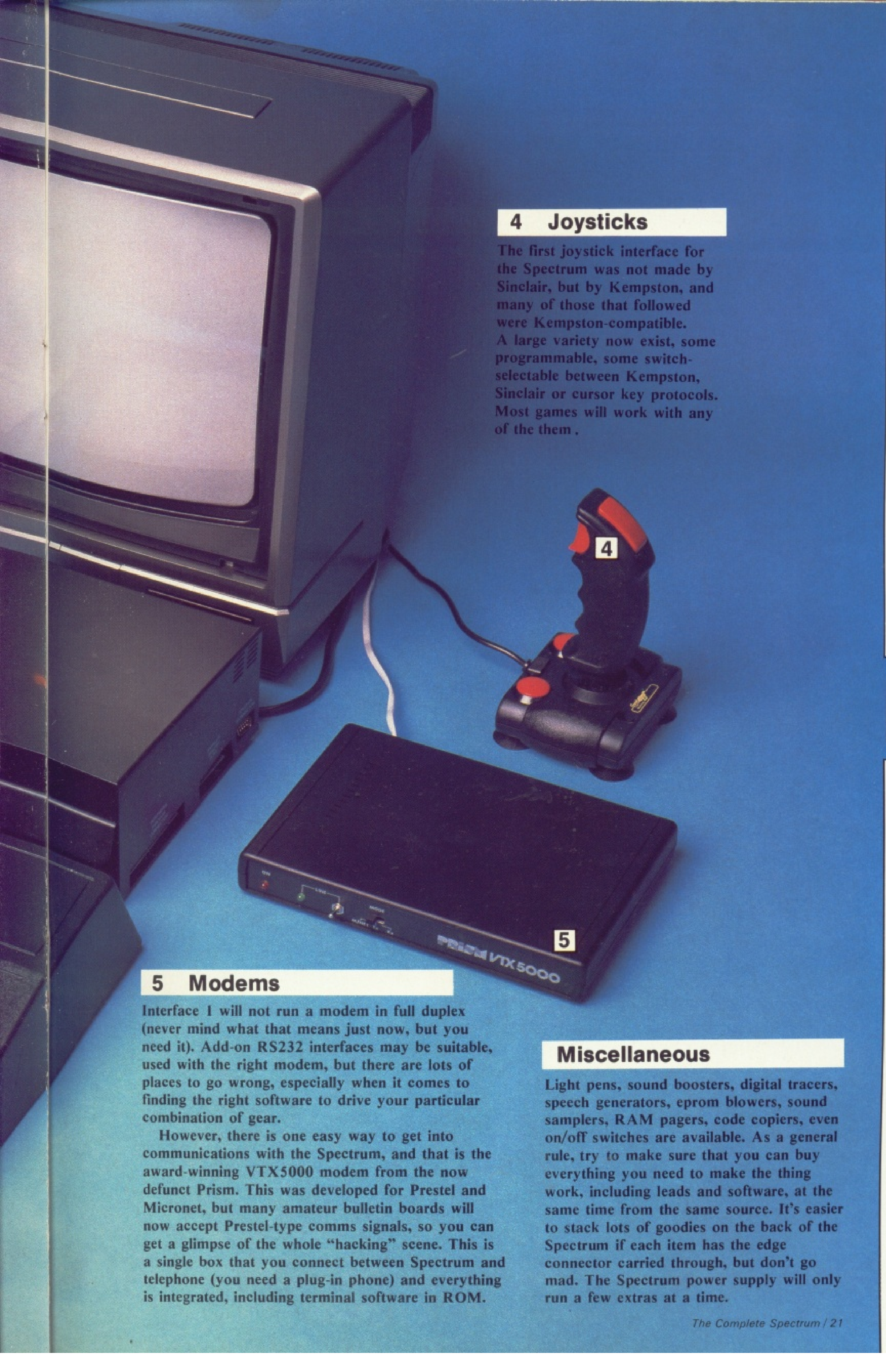
The Spectrum can run printers costing from £50 to a couple of thousand via the edge connector, Interface 1's RS232 output, or other add-on interfaces including a Centronics one from Kempston. Some quite good full size printers are now going for just over £100, but make sure that everything will mate up – including your word processing software – before you pay out all that money.

2 Disc drives

Discs to fit the Spectrum come from several sources, Opus, Watford and Triton to name but three. A tape loop system similar to Microdrives called the Rotronics Wafadrive is also available. The Opus disc drive is becoming particularly popular since it has printer and joystick interfaces built in. What's more, software is starting to appear on Opus discs.

3 Keyboards

Lots of replacement keyboards are available, some of them a big improvement even over the latest Spectrum keyboard on the Plus. Some, however, are dire. Try before you buy. The Spectrum generally fits inside the case of the keyboard.



4 Joysticks

The first joystick interface for the Spectrum was not made by Sinclair, but by Kempston, and many of those that followed were Kempston-compatible.
A large variety now exist, some programmable, some switch-selectable between Kempston, Sinclair or cursor key protocols. Most games will work with any of the them.

5 Modems

Interface 1 will not run a modem in full duplex (never mind what that means just now, but you need it). Add-on RS232 interfaces may be suitable, used with the right modem, but there are lots of places to go wrong, especially when it comes to finding the right software to drive your particular combination of gear.

However, there is one easy way to get into communications with the Spectrum, and that is the award-winning VTX5000 modem from the now defunct Prism. This was developed for Prestel and Micronet, but many amateur bulletin boards will now accept Prestel-type comms signals, so you can get a glimpse of the whole "hacking" scene. This is a single box that you connect between Spectrum and telephone (you need a plug-in phone) and everything is integrated, including terminal software in ROM.

Miscellaneous

Light pens, sound boosters, digital tracers, speech generators, eprom blowers, sound samplers, RAM pagers, code copiers, even on/off switches are available. As a general rule, try to make sure that you can buy everything you need to make the thing work, including leads and software, at the same time from the same source. It's easier to stack lots of goodies on the back of the Spectrum if each item has the edge connector carried through, but don't go mad. The Spectrum power supply will only run a few extras at a time.

On-screen printing

All the facts you need to know about writing to the screen – at your fingertips

PRINT — P key

Prints a string (text) or number to current print position on the screen. Items to be printed may be variables, string variables, text within quotes, numbers, AT or TAB, and colour controls. Items must be separated by a semicolon, or (to move the print position) a comma or apostrophe.

PRINT 0 — P key

Prints in the bottom two lines of the screen (use carefully).

AT x,y — I key, Symbol Shift

Moves printing position to line x, column y. Must be used with PRINT. *(Note that key is I, not 1.)*

TAB x — P key, Extended Mode

Moves printing position to column x by printing spaces. Must be used with PRINT.

Memory Map

16384 to 22527: Screen RAM, can be poked and peeked, but this is usually easier to handle in Basic with PRINT and DRAW, and so on. This memory is organised in a funny, disjointed way, and is not easy to work with.

22528 to 23295: The attribute file, can be poked to change the displayed colours. This is usually no advantage over using PRINT OVER to change the screen colours without changing the text displayed, except in the bottom two lines, where you can't usually print. The attributes for the bottom two lines are stored from 23232 to 23295. This is coded in an awkward form that works best in binary. In decimal it is the INK colour (0 to 7) plus 8 times the PAPER colour, plus 64 if BRIGHT, plus 128 if FLASHing.

INK x — X key, Extended Mode

Change ink colour to x – 0 to 9, where 8 is transparent and 9 is contrasting. Transparent means that the colour is taken from the INK already set at that screen position.

PAPER x — C key, Extended Mode

Change paper colour to x – 0 to 9, where 8 is transparent and 9 is contrasting.

FLASH x — V key, Extended Mode

Turns flash on (x=1) or off (x=0) or transparent (x=8).

BRIGHT x — B key, Extended Mode

Turns Bright on (x=1) or off (x=0) or transparent (x=8).

OVER x — N key, Extended Mode

Turns OVER on (x=1) or off (x=0). When OVER is on the printing or plotting of the new pixels is XORed with what is already on the screen. If either the old pixel or the new one is INK (but not both) then INK will be printed. But if both old and new are PAPER, or both INK, then PAPER is printed. This allows characters to be printed "on top" of each other.

INVERSE x — M key, Extended Mode

Turns inverse on (x=1) or off (x=0). When INVERSE is on PAPER is printed as INK and vice versa.

LIST — K key

Lists the current Basic program. Use LIST x to start the listing at line number x.

ATTR (x,y) **L key, Extended Mode**

Used as variable, returns attribute code for screen line x, column y.

BORDER x **B key**

Changes Border colour to x, (0 to 7).

SCREEN$ **K key, Extended Mode**

When used in a SAVE or LOAD, this substitutes for CODE 16384,6912, causing the whole of the screen RAM to be saved or loaded, as in SAVE "pic" SCREEN$

SCREEN$ (x,y) **K key, Extended Mode**

When used as a string variable this returns the character that is being displayed at line x, column y, but only if it is part of the character set, not a UDG or block graphic. Note brackets.

System Variables

23624: The border colour and the lower screen attributes are stored here. POKE 23624,63 to change both INK and PAPER white in the lower screen, for instance. (Ho, Ho.)
23688: Column number of current PRINT position, reversed: counts from right, column 0=33.
23689: Line number of current PRINT position, reversed: counts from bottom, line 0=24.
23692: Scroll count, used to tell when to stop printing and ask "scroll?". Keep poking with 255 to make screen scroll without stopping to ask.
23693: Permanent colours for INK, PAPER, FLASH, and so on, stored as binary code in same way as in attribute file.
23694: Binary mask for transparent colours. This can be used to protect colours from being overprinted or make colours mix, but it requires experimentation.

Hints & Tips Hints & Tips Hi

NORMALLY you are not able to PRINT in the bottom two lines of the Spectrum's screen, as this area is reserved for use by prompts and the Basic line editor. However it is possible to print there by adding a few characters to the PRINT statement.

This area is actually an independent print window. To direct print there rather than to the main screen use:

```
10 PRINT   0; “Text”
```

You can clear this part of the screen independently too, with:

```
20 INPUT ” ”
```

This part of the screen does not behave in the same way as the main screen though, and some experimentation is necessary. For one thing, you are not limited to the two lines that are normally in the lower screen. If you print more the window expands, scrolling the main screen up out of its way.

You can even use PRINT AT, but remember that the lower screen does not just continue the line numbering of the upper screen, but starts over at zero. Try this:

```
10 BORDER 2
20 PRINT   0; “TEXT”;AT 12,0;“MORE
   TEXT”
30 PAUSE 0: REM – TO GIVE TIME TO SEE
40 INPUT ” ”: REM – NO SPACE BET-
   WEEN QUOTES
50 PAUSE 0
```

The BORDER command is just to make the lower screen a different colour so you can tell that the print is in fact appearing in the lower screen. The PAUSE commands will wait until a key is pressed.

YOU may already know all about using PAPER and INK commands to change the colours that the Spectrum prints on the screen, but did you know that you can even change the colours of your program listings within the lines?

This technique can be confusing because it uses invisible control characters which do not show when you are editing a line. Furthermore, there are two characters for each colour change, so when you try to delete them you can get strange effects if you only delete one.

Entering the colour controls into a line is easier than getting them out again. First you select Extended Mode, then you press a number key, either with or without Caps Shift (not Symbol Shift).

The effects that you can get are as in the chart below. Note that any change you make will continue right past the end of one line into the rest of the listing unless you change it back at the end. Be careful, if you set both INK and PAPER to the same colour, the whole listing could become invisible!

Number	Without Caps Shift	With Caps Shift
1	Paper blue	Ink blue
2	Paper red	Ink red
3	Paper magenta	Ink magenta
4	Paper green	Ink green
5	Paper cyan	Ink cyan
6	Paper yellow	Ink yellow
7	Paper white	Ink white
8	Bright off	Flash off
9	Bright on	Flash on
0	Paper black	Ink black

At the heart of your micro

MIKE COOK examines what makes your machine tick

LIFT the bonnet of many Formula 1 racing cars and you are likely to see the same engine. OK, the chassis is different, the suspension, gearbox and body are all different but most of the engines are the same. This is also true with personal computers. The microprocessor, the powerhouse of your computer, is the same on many different models. So let's take a look at it and see how it shapes the rest of our machine.

There are two microprocessors which go to make the vast majority of computers, the Z80 and the 6502. The Spectrum uses the Z80 as do the Tandy TRS80, Sharp MZ80, Amstrads and many others. They all share the same engine, so a beginner is often puzzled why they don't all share the same software.

The answer is simple – they all have different electronic "bodies" wrapped round them, just like the racing cars. This makes software written for one machine totally incompatible with any other machine. Let's see how this comes about by looking a little deeper into the structure of the Z80.

The Z80 was one of the late comers on the 8 bit microprocessor scene. However it was electronically easy to build it into a system as it simplified the interfacing of dynamic memory chips or RAMs to a microprocessor. It also needed fewer support chips to make it into a working system, and so became very popular in new computers.

It had another bonus in the form of software compatability. The Z80 was made by a new company, Zilog, the founders of which were some of the Intel engineers who designed the 8080 microprocessor.

The 8080 was the first microprocessor to find its way into home hobby stystems. In those days there were no keyboards or ROMs (Read Only Memories). You had to start the computer by entering the code on a large bank of switches. Each switch represented a binary bit – up for logic one and down for zero.

The point was that as the 8080 had been around for some time there was a lot of software available for it. What the designers of the Z80 did was to make sure that the new microprocessor would obey all the instruction codes of the 8080 plus some powerful new ones. The result was that when the processor arrived there was already a great bank of software that would run on it, or at least could be easily modified. This was possible because in those days computers did not have graphic and sound capabilities, the major source of program incompatibility. Most were connected to a simple teletype terminal, and thus conversion was simple.

Instruction set

A microprocessor's instructions are known as the instruction set, and ultimately govern the power of the machine because everything the computer does must eventually be broken down to these basic instructions. In general, the fewer instructions you need to express a problem the faster the machine will go.

The instructions are very simple, involving operations like moving data from one place to another or adding up two small numbers. Most real things you want to do need lots of these "mini" instructions. For example, a program to print "Hello" on the screen would take about 20 instructions.

The art of using the instruction set to get the computer to do anything is quite involved and many books have been written concerning machine code programming. Basically the technique is to break the problem into a small number of steps simple enough to be handled by the Z80.

Some of the Z80 instructions are quite powerful. These are known as block commands and operate on a whole range of memory instead of the more usual one location. They can move blocks of memory around or even search a block of memory looking for a specific byte. They are in fact mini programs commanded by a single machine code instruction. Given the right sort of problem in skilled hands these instructions can be very powerful, resulting in fast programs.

How the Z80 sees the world – memory and I/O

To a Z80 the rest of the world looks like lots of different pigeon holes or memory locations. In fact there are two types of locations it sees. One is reserved for input and output to the system and the other is used for everything else including input and output.

The Z80 can only cope with one of these locations at a time. It signals to the electronics surrounding it which location it wants to access by setting the address of the location on 16 signal wires.

Each signal wire can be in one of two states, with a voltage on it (5 volts) or with no voltage on it (0 volts). We call these states one or zero. 5 volts corresponds to one. You see if we called it 5 which might seem logical it would imply that 4, 3, 2, and 1 volts existed. These voltage levels cannot exist in the circuit, only one of the two states – that's why we call it a binary condition.

As there are 16 of these address signal wires there are lot of combinations of zero and one that they can be in. In fact if you work it out this comes to 65,536, or as we say in the jargon, 64k. This is because 1k is 1024, a sort of baker's dozen version of 1000.

There is an extra wire which indicates whether the microprocessor is performing a memory operation or an input/output operation, so we might expect another 64k of input/output space. In practice only the lower 8 bits of the address lines are used to give 256 input/output locations – more than enough for anyone. In addition to this if we need extra input/output we can always use the 64k of the main memory.

The wires that signal these addresses are known collectively as the address bus. Up till quite recently 64k was a vast amount of memory, quite over and above anything that was practicable or affordable. I remember in 1978 getting a memory board for one of my computers containing 4k of memory at twice the cost of a Spectrum 48k+. Even so I was impressed at how cheap it was – it represented a breakthrough at the time. Nowadays you can get 64k of memory in just two chips, so you see that technology has only recently caught up with the capabilities of this microprocessor.

The data bus – how a micro gets the facts

The microprocessor examines the memory locations by means of eight signal wires. These carry information to and from the locations in the same sort of binary (once and zero) signals used on the address bus. As these wires carry the information or data they are known collectively as the data bus.

So if the microprocessor wants to look at a memory location it puts its address on to the address bus and reads the contents off the data bus. Conversely, if it wants to store some information it again places the address on the address bus and the data it wants to write on to the data bus. It is up to the electronics surrounding the processor to service its needs by taking or placing the data on the bus.

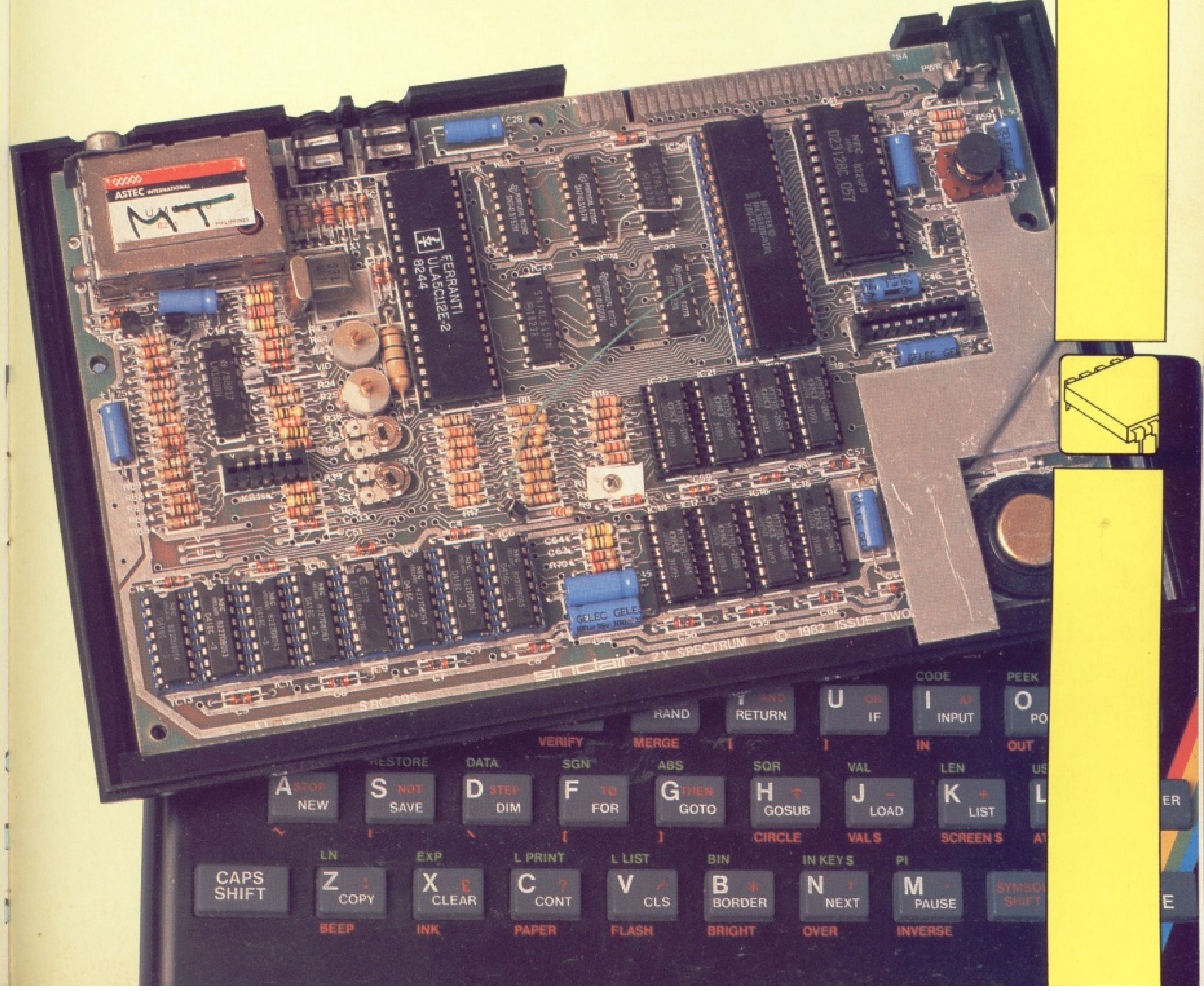
The processor sees everything simply as memory locations. All the devices that make up the computer have to be allocated their own unique address or range of addresses.

This applies to the keyboard, the screen, cassette recorder, joysticks and whatever else goes to make up your particular computer. If designers choose to put these components in different places, or have a different mix of components, then inevitably software becomes incompatible.

Registers – the chip's internal workshops

As well as the external memory locations, the processor has inside it some internal memory locations called registers. They're not given a numbered address, but names. Admittedly they are not very imaginative names but they suffice. There are 24 in all as shown in Figure I. Each has its own use and nearly every instruction that the processor can execute involves one or more registers. Let's take a look at what they are used for.

The program counter is a 16 bit register used to hold the address of the next instruction. The processor puts this out on to the address bus and fetches the data in that location. This is in the form of a coded instruction, but before that instruction is executed the program counter is incremented to point to the next instruction. This can be complicated by the fact that a complete



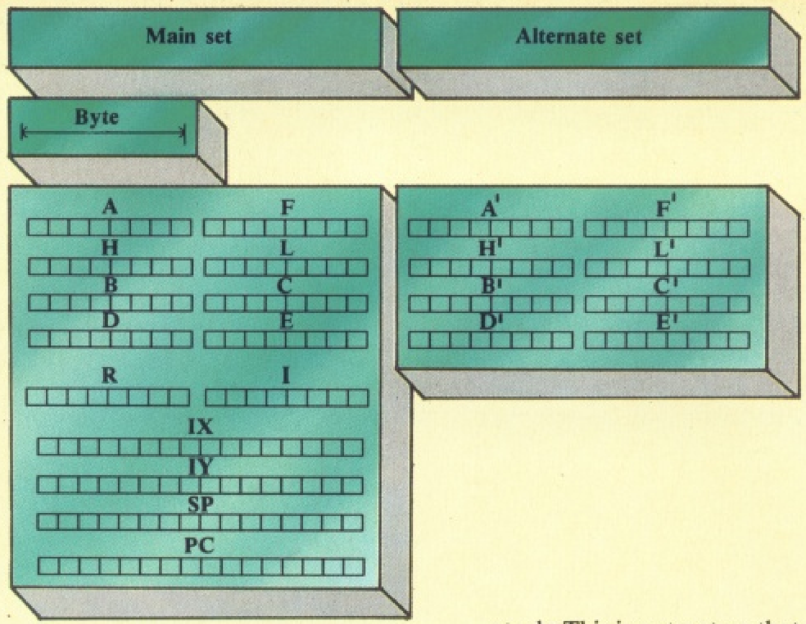


Figure I: Z80 register set

instruction can be stored in one, two, three, four or five successive memory locations.

The first part of the instruction contains the information concerning how many other locations are involved. In this way the program counter looks after itself without any intervention from the programmer.

If any data needs to be moved or manipulated then register A is used. There are instructions to move data from a memory location to the A register and from the A register to memory. While data is in the A register it can be manipulated. This can mean having a value added, subtracted or having individual bits changed. The instruction set also allows multiplying or dividing by two. All other operations have to be derived from these.

The six general purpose registers, H, L, B, C, D and E, are used for holding temporary results as it is much quicker to move these into the A register than it is to move memory into the A register. In addition the registers can be used in pairs to form three 16 bit registers, the pairing being HL, BC and DE.

The IX and IY registers are known as index registers and are also 16 bits wide. These, like the register pairs, are used to point to other memory locations. This means that the program can calculate the memory locations to operate on instead of them being fixed when the program was written. This gives the instruction set most of its power.

The way you arrive at the address of memory you're after is known as the addressing mode. The Z80 has quite a few and they are at the root of its power.

The flag register differs from the rest in that it does not contain numbers as such but a collection of bits. Each bit has its own name and significance. Some operations cause the individual bits in the status register change to reflect the result.

Suppose, for example, we wished to test the A register to see if it contained a certain value. We would use a compare instruction and the result would be the setting of one of the bits in the flag register.

The point is that all the conditional instructions work off this flag register. For example, if you want to skip a section of code if you were comparing identical items, you would use a "Branch if Equal" instruction which causes a specified number of address locations to be skipped if the equal flag is set. This then alters the program counter and causes the next instruction to be fetched from further down the program.

The stack pointer is used to implement a push-down stack. This is a structure that you may have seen if you have used any of the early Hewlett Packard pocket calculators that used reverse Polish notation. In the Z80 the stack is used to store subroutine return addresses as well as acting as a temporary store.

The R register is used to refresh dynamic RAMs. These have to be read periodically or they will forget what they hold. The Z80 continually cycles the value in this register to ensure that this happens. This is done automatically and the programmer need not know it is happening.

The only use of the R register to the programmer is to generate a random number seed. However, your computer memory could commit suicide if you wrote a program to continually write zero into this register. Within a tenth of a second it would forget everything it ever knew. Of course turning the power off and on again would restore its correct operation.

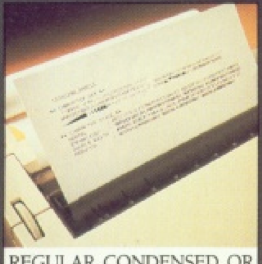
The I register is used when the microprocessor receives an interrupt – a signal on a pin that tells the processor to stop what it is doing and start some other program. It is like a hardware call to a subroutine since, when it has responded to the needs of the interrupting device, the original program is resumed.

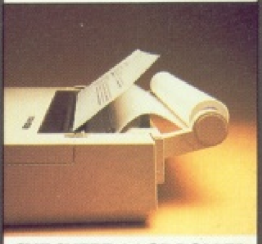
There are three ways the Z80 can respond to an interrupt, depending upon what interrupt mode it has been set to run in. In the most complex one, mode 2, the I register is used as part of the information to find out where this interrupt service subroutine is located in memory.

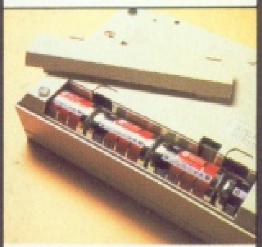
In Figure I there are registers marked with a dash, for example the A. These are known as the alternate register set. This sounds like something the Friends of the Earth would be interested in but it turns out to be less fascinating. There are instructions to swap main and alternate register sets. One exchanges the A and F registers and the other the rest of them. All the instructions will then work with the alternate register set instead of the main one. This allows the main register set to be preserved while another section of the program is run. This is useful when responding to interrupts as the original values of the registers can be preserved and so not affect the interrupted program.

Although the Z80 may seen complex at first, after a while you'll see how straightforward it really is. Then you'll be amazed at how powerful such a simple system can be. However, to take full advantage of the Z80, you need to know a little about machine code. If you turn to Page 32 of The Complete Spectrum you'll see just the thing – a beginner's guide to machine code.

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.

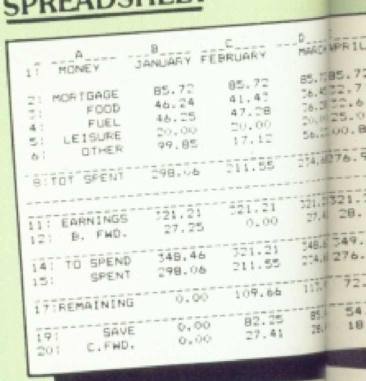
QUICK TO LEAR

THAT'S...

FINALIST MICRO BRITISH COMPUTING AWARDS 1985

MINI OFFICE

SPREADSHEET



```
        ___A____  ___B____ ___C____
 1:   MONEY     JANUARY FEBRUARY

 2: MORTGAGE      85.72    85.72
 3:     FOOD      46.24    41.43
 4:     FUEL      46.25    47.28
 5:  LEISURE      20.00    20.00
 6:    OTHER      99.85    17.12
----------------------------------
 8: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!

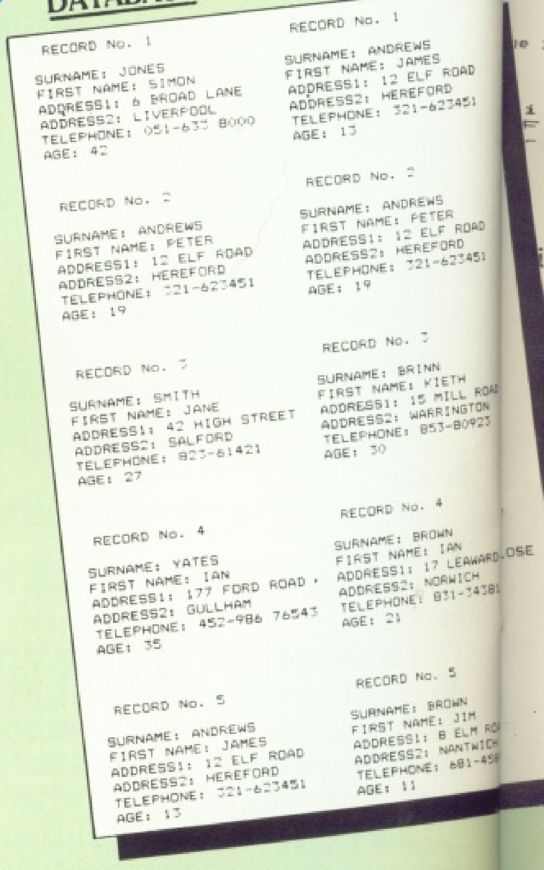
WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

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

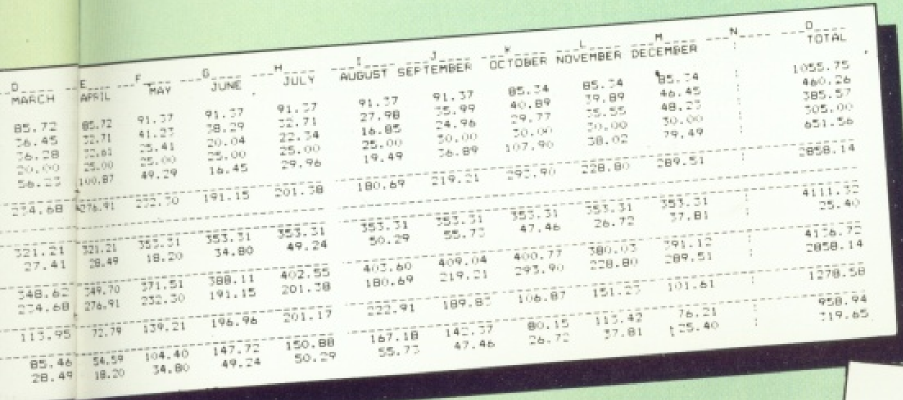


```
RECORD No. 1                      RECORD No. 1

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 ROA
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 LEAWARD
ADDRESS2: GULLHAM                 ADDRESS2: NORWICH
TELEPHONE: 452-986 76543          TELEPHONE: 831-3438
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 RO
ADDRESS2: HEREFORD                ADDRESS2: NANTWICH
TELEPHONE: 321-623451             TELEPHONE: 681-45
AGE: 13                           AGE: 11
```

...and it's all at t
price of just £

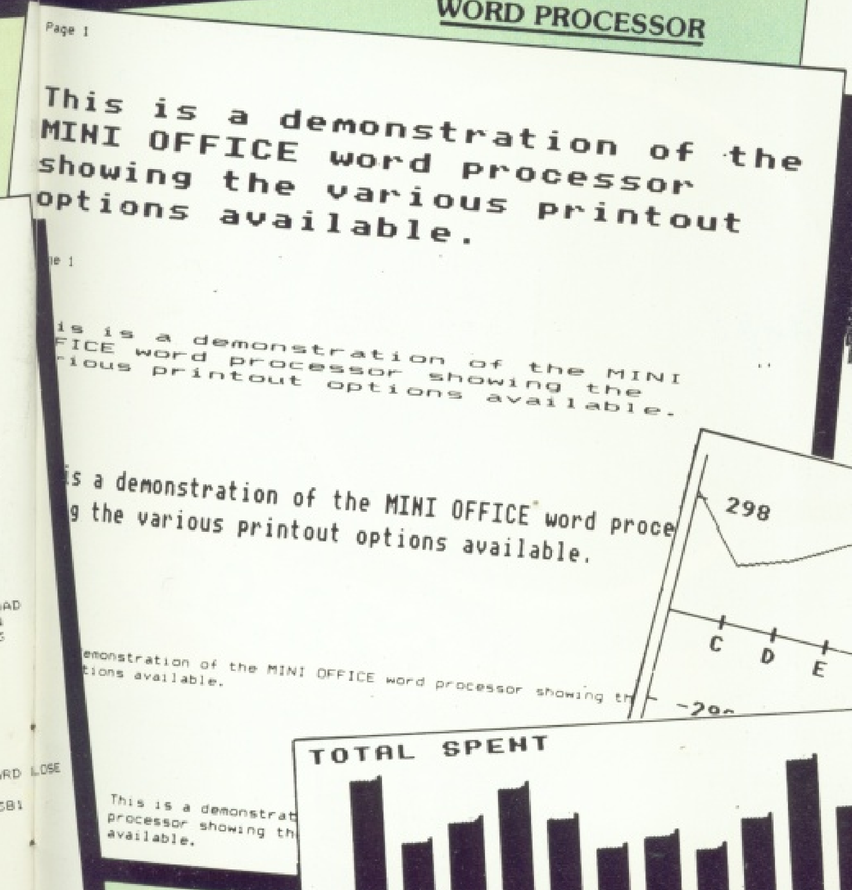
RN, EASY TO USE

Spectrum 48k and Spectrum+

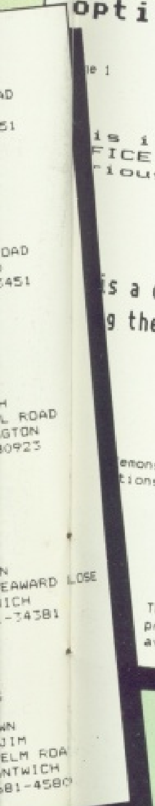


```
_D_____ __E____ ___F____ ___G____ ___H____ ___I____ ___J____ ___K____ ___L____ ___M____ ___N____ ___O____
MARCH   APRIL   MAY      JUNE     JULY     AUGUST   SEPTEMBER OCTOBER NOVEMBER DECEMBER          TOTAL

 85.72   85.72   91.37    91.37    91.37    91.37    91.37    85.34    85.34    85.34      :     1055.75
 36.45   32.71   41.23    38.29    32.71    27.98    35.99    40.89    39.89    46.45      :      460.26
 36.28   32.61   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
 56.23  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
-----------------------------------------------------------------------------------------------------------
321.21  321.21  353.31   353.31   353.31   353.31   353.31   353.31   353.31   353.31      :       25.40
 27.41   28.49   18.20    34.80    49.24    50.29    55.73    47.46    26.72    37.81      :
-----------------------------------------------------------------------------------------------------------
348.62  349.70  371.51   388.11   402.55   403.60   409.04   400.77   380.03   391.12      :     4136.72
234.68  276.91  232.30   191.15   201.38   180.69   219.21   293.90   228.80   289.51      :     2858.14
-----------------------------------------------------------------------------------------------------------
113.95   72.79  139.21   196.96   201.17   222.91   189.83   106.87   151.23   101.61      :     1278.58
-----------------------------------------------------------------------------------------------------------
 85.46   54.59  104.40   147.72   150.88   167.18   142.37    80.15   113.42    76.21      :      958.94
 28.49   18.20   34.80    49.24    50.29    55.73    47.46    26.72    37.81    25.40      :      319.65
```

WORD PROCESSOR

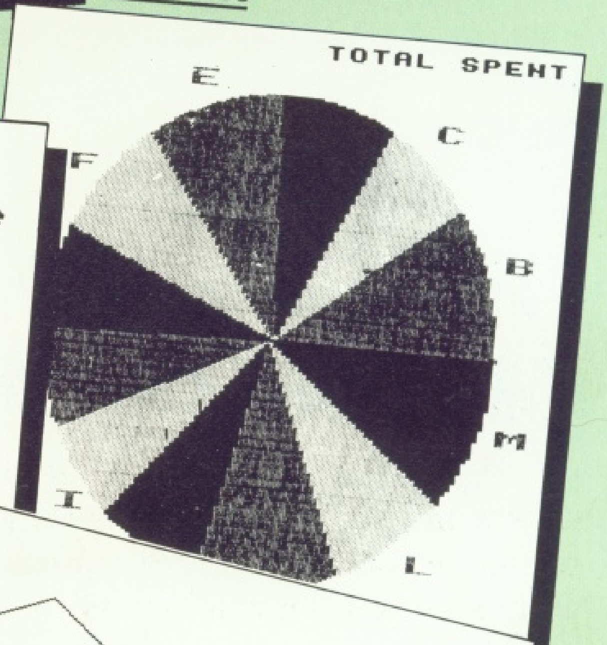


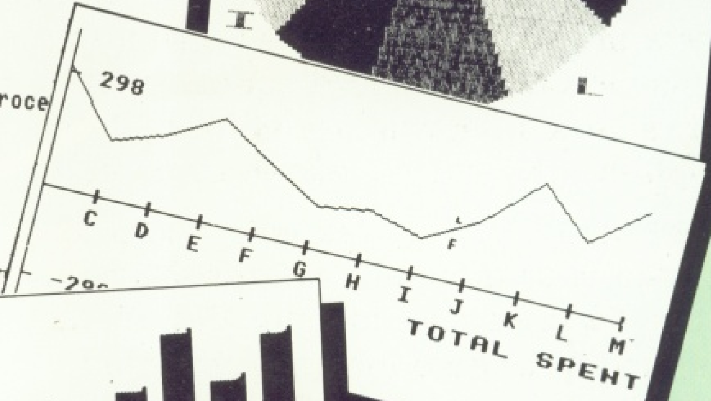
```
Page 1

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.

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

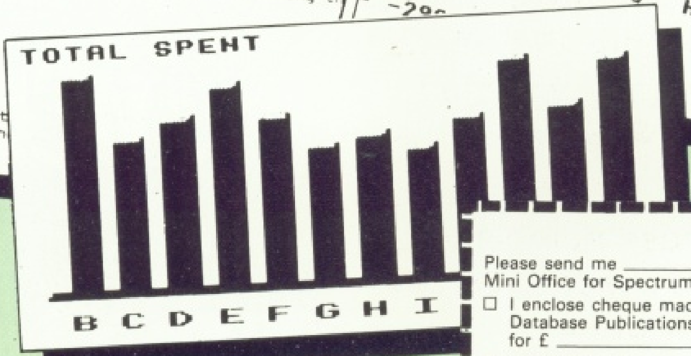


```
AD

51

OAD

3451

H
L ROAD
GTON
30923

N
EAWARD CLOSE
ICH
-34381

WN
JIM
ELM ROA
NTWICH
581-4580
```

GRAPHICS







at the unbelievable
st £5.95 CASSETTE

DATABASE SOFTWARE

Please send me ____________ copy/copies of Mini Office for Spectrum 48k/Spectrum+	Only £5.95
☐ 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

Character reference

KEVIN EDWARDS helps you get to grips with graphics

THE Spectrum is renowned for the amazing power and versatility of its graphics. To take advantage of its capabilities, however, you have to begin with the fundamentals – the A, B, Cs of Spectrums.

Each letter, digit, graphic symbol and punctuation mark the Spectrum can produce is known as a character. Computers find it much easier to work with numbers rather than shapes, so each character is given a special number or code. This is always between 0 and 255.

The complete group of characters is known as a character set and is split into five main groups. In ascending numerical order they are:

- **★ Control**
- **★ Letters/punctuation/digits**
- **★ Block graphics**
- **★ User definable**
- **★ Tokens**

It's well worth looking at each of these in turn, as an understanding of their very different properties can make the difference between amateur and professional – looking programs.

The Spectrum's ABC

THE second group – from code 32 to 127 – contains the everyday characters that you'd find on a typewriter, along with a few special symbols. Table I shows them together with their character codes. You can display this group of characters with this program:

```
10 FOR L=32 TO 127
20 PRINT CHR$ L;
30 NEXT L
```

The CHR$ command informs the Spectrum that a character is to be produced. The number following it is the character's own unique, reference number. The command CODE, appropriately named, complements CHR$ and produces the code for a particular character.

PRINT CODE "A"

will show you the Spectrum's character code for the letter A.

Code	Character	Code	Character
32	space	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	/
45	–	93	]
46	.	94	↑
47	/	95	_
48	0	96	£
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	|
77	M	125	}
78	N	126	~
79	O	127	©

Table I

Keeping control

THE control characters, from 0 to 31, are very special. Although they don't have an associated, visible character they can, in several instances, help to produce the same effect as the more usual Basic commands. Table II gives all the details.

Just as the Basic commands require additional numbers to complete the instruction as in, say, INK 2, so we follow their control code equivalents with similar numbers. The range, quantity and order of these should always be identical to those of the Basic version. When such additional parameters are required for an instruction they should be "glued to" CHR$ and tagged together with +. For example, if we wish to change the INK colour to 3, magenta, and print HELLO, we can use:

```
PRINT CHR$ 16+CHR$ 3;"HELLO"
```

This first CHR$ selects the INK change control character (16) and then the new INK colour (3). It is the exact equivalent of Basic's:

```
PRINT INK 3;"HELLO"
```

From Table II you can see that only about half of the possible control characters are used. Printing the unused control characters has no effect.

Probably the most useful of all the control codes are ones associated with cursor movement. Codes 8 and 9 move the text cursor left and right respectively relative to its present position.

This permits us to print things on the screen and alter them by moving the cursor across the line and

Code	Character	Code	Character
0		16	INK
1		17	PAPER
2		18	FLASH
3	unused	19	BRIGHT
4		20	INVERSE
5		21	OVER
6	PRINT comma	22	AT
7	EDIT	23	TAB
8	cursor left	24	
9	cursor right	25	
10	cursor down	26	
11	cursor up	27	
12	DELETE	28	unused
13	ENTER	29	
14	number	30	
15	unused	31	

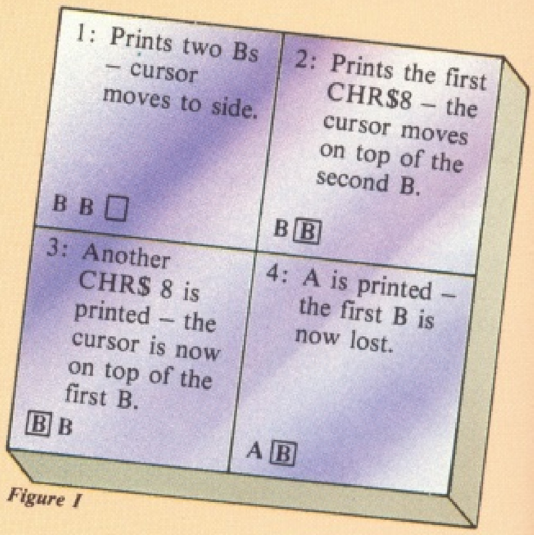
Table II

overprinting with new text. The program below shows how control code 8 can be used to change something previously printed:

```
10 LET A$="BB"+CHR$ 8+CHR$ 8+"A"
20 PRINT A$
```

First of all two Bs are printed on the screen. After this the cursor is automatically moved one place to the right by the Spectrum – ready for the next letter. Printing control code 8 twice, however, moves the cursor back – that is left – two characters from this position. This places the text cursor directly on top of the first B. By now we've reached the A which is duly printed, over-printing this first B. The final effect is to leave AB on the screen. Figure I shows this happening step by step.

After all this the cursor ends up at the start of the


1: Prints two Bs – cursor moves to side.
B B □
2: Prints the first CHR$8 – the cursor moves on top of the second B.
B B
3: Another CHR$ 8 is printed – the cursor is now on top of the first B.
B B
4: A is printed – the first B is now lost.
A B
Figure I

next line, since PRINT A$ is not followed by a semi-colon.

Codes 10 and 11 are supposed to move the cursor up and down. If you experiment you'll see this isn't the case.

Adding strings together using + combined with the use of control codes is a useful programming technique that can save typing as well as time.

Block graphics

THE graphics characters, codes 128 to 143, consist of simple 2 by 2 blocks – as shown in Table III. These are perfect for simple low resolution drawings where detail is not crucial.

All possible shaded/un-shaded combinations of quarter blocks are available so there shouldn't be any problem designing decent artwork. The program below demonstrates how the block characters can be used to draws a car:

```
10 PRINT CHR$ 132+CHR$ 140
20 PRINT CHR$ 139+CHR$ 143+CHR$ 139+
   CHR$ 138
```

If you do decide to make a picture out of block graphics it's a good idea to plan it on squared paper first – it can save a lot of time!

DIY graphics

THE Spectrum allows users to define their own graphic characters. These are known as user definable graphics – UDGs – and are given the codes 144 to 164 – 21 in total. You may not know it, but each character is made

Code	Character	Code	Character
128	□	136	▖
129	▝	137	▞
130	▘	138	▌
131	▀	139	▛
132	▗	140	▄
133	▐	141	▟
134	▚	142	▙
135	▜	143	█

Table III

up of 64 dots (or pixels as we call them) in an 8 by 8 grid or matrix. Take a look at Figure II.

This shows a matrix containing a pattern for a stick man. The asterisks show where a pixel is to be set and the dots indicate spaces. All that is needed now is a way of converting the picture into a series of numbers which the Spectrum understands. This is achieved using eight binary numbers. These are easy to handle thanks to the Spectrum's BIN command – the one lesser micros miss out.

For each pixel set we use a binary digit 1 and for each blank space a 0. In other words the asterisks are changed to '1's and the dots become '0's. These are put into a program as a series of data statements:

```
100 DATA BIN 00111100
110 DATA BIN 00111100
120 DATA BIN 00011000
130 DATA BIN 11111111
140 DATA BIN 00011000
150 DATA BIN 00100100
160 DATA BIN 01000010
170 DATA BIN 10000001
```

Next, the new character must be given its own character code between 144 and 164. Normally these codes produce the letters A, B, C and so on up to U duplicating the codes for A to U in the lower range. However, this is just a temporary filling. Once they have been redefined they will produce the new character.

Don't be confused by the fact that the character set appears to contain two letter As, Bs and so on. They are totally separate. If you've redefined the UDG that was initially filled with A (CHR$ 65) it is totally different from the "authentic" A of PRINT "A" and PRINT CHR$ 65.

The way in which the character code of the new UDG is specified along with its definition (data bytes) seems rather strange at first, but is actually quite easy to get used to.

The USR command is used to find the location in memory of the character definitions for the UDGs. Using this address the new character definition bytes can be poked into memory to create the new character. The five lines below, in conjunction with eight DATA lines such as the ones above, will re-define character code 144 (initially the counterfeit A) and print it.

```
10 FOR L=0 TO 7
20 READ Q
30 POKE USR "A"+L,Q
40 NEXT L
50 PRINT CHR$ 144
```

The letter within quotes in line 30 indicates the character which is to be re-defined. This should be a letter between A (code 144) and U (code 164). Any invalid character, such as Z, will cause an 'Invalid argument' error since it cannot be re-defined.

The UDGs can be displayed using CHR$ or by entering the graphics mode and pressing a letter key up to and including U. It seems confusing to use the letters A to U when re-defining characters when in fact you don't alter A to U but character codes 144 to 164. You'll soon get the hang of it, though.

Basic tokens

THE final group in the character set is the tokens, codes 165 to 255. Tokens are numbers which represent the Basic commands (keywords) available on the Spectrum – a type of shorthand. When Basic programs are stored in memory, rather than storing each letter of the keywords, the token byte is used instead.

This is because storing a complete keyword requires the same number of memory locations as there are characters in the keyword. However, if tokens are used only one location is required no matter how long the keyword is. This saves a lot of memory, especially with larger programs.

You'll probably never use character codes 165 to 255 in your Basic programs. However once you move on to machine code you'll find an understanding of how Basic works invaluable.

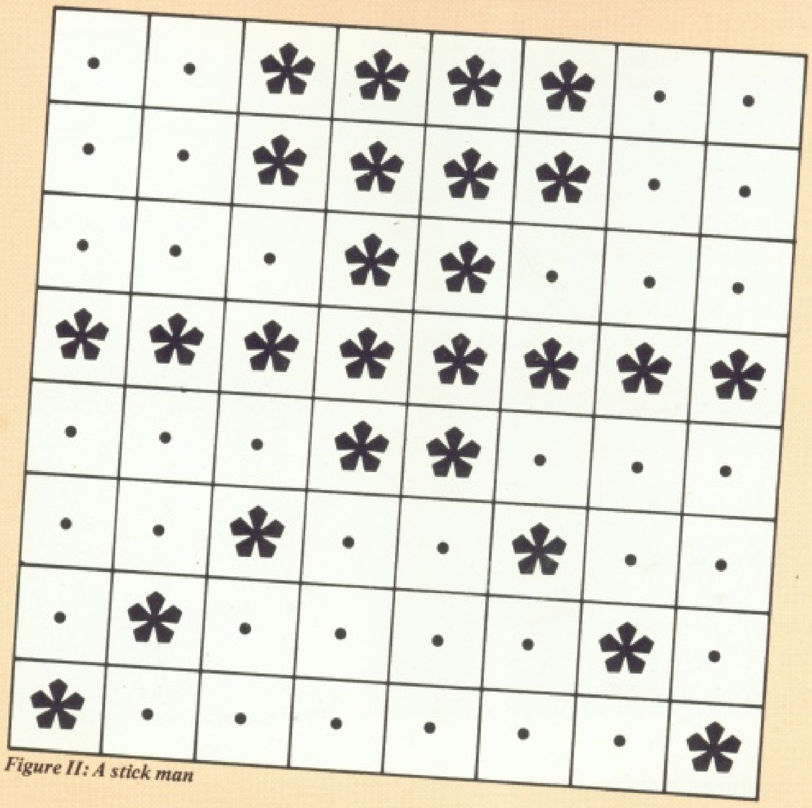


Figure II: A stick man

Code of binary conduct

THIS is the first in a series of articles in which we hope to take the mystery out of understanding the fundamentals of how your Spectrum works.

All too often even competent Basic programmers tend to shy off such topics as binary coding, hexadecimal and assembly language because it seems too "mathematical". This is a great pity, because a little knowledge in these fields allows you to take full advantage of the Spectrum's advanced facilities.

MIKE BIBBY helps take the mystery out of the way your Spectrum handles numbers

To start we'll look at binary code – a way of handling numbers essential to our understanding of what goes on inside a computer. Binary is just a means of coding numbers in a way particularly suitable for computers. It's actually quite simple. What often confuses beginners is the fact that the binary system codes numbers in a way that can look extremely like the way we normally code numbers.

For example, if you were presented with a number 100, you would probably decode it in your normal manner and say it was "one hundred". That, however, is just one way of interpreting it. If you decided to

How binary works – a monetarist example

TO give us an idea of how binary numbers work think about the coins we use every day. Our currency consists of these coins:

50p 20p 10p 5p 2p 1p

We can combine them to give any sum we wish. For example 75p is

50p + 20p + 5p

or

50p + 10p + 10p + 5p

and so on.

We are all familiar with this – often we use multiples of coins to make up a sum. For example, 5p can be **2p** + **2p** + **1p**. Using the same coin twice, though, often means that we end up carrying unnecessary amounts of change, and I for one don't like doing that.

Sometimes, however, with our present coinage system we have to use the same coin twice to obtain certain sums. You cannot, for instance, make up the sum of 4p without doubling up on coins. To avoid repeating coins we would have to invent a 4p coin! Let's do that; in fact, let's invent a coinage system where you never have to use the same coin twice.

First of all we would need a **1p** coin and, of course, a **2p** coin, because we cannot use **1p** + **1p** for 2p – it breaks the rule! Now 3p can be made up of **1p** + **2p**, but for 4p we'll have to invent a **4p** coin.

Equipped with that we can make 5p (**4p** + **1p**), 6p (**4p** + **2p**), and 7p (**4p** + **2p** + **1p**). In obtaining 7p we used all our available coins, so now we have to invent an **8p** coin. If you work it out (and I suggest you have a go) you will find that with the coins you have at your disposal (**8p, 4p, 2p, 1p**) you can make any sum up to 15p. Then you would have to invent a new coin, **16p.**

Notice how the coins we have created have doubled in value: **1p, 2p, 4p, 8p, 16p.** No prizes for guessing what the next one is.

Let's summarise our results in a table (Figure I). Here I have used the columns to show the coins available and the rows to show how the various totals are made up. A 1 in a particular column means that we use that column's coin, and 0 means that we don't use it. Look at the row for 5p. It has 101 on it. According to our rule, this means we pick out the coins **4p** and **1p** (and NOT **2p**) to make up the 5p total.

```
    4p  2p  1p
%   1   0   1
    4p  +   1p = 5p
```

Now let's get back to computers by dropping all this talk about coins and redraw Figure I to show the same information but without referring to money – just numbers. Figure II is the new table. As you can see, there is little

		COINS			
		8p	4p	2p	1p
TOTALS	1p				1
	2p			1	0
	3p			1	1
	4p		1	0	0
	5p		1	0	1
	6p		1	1	0
	7p		1	1	1
	8p	1	0	0	0
	9p	1	0	0	1
	10p	1	0	1	0
	11p	1	0	1	1
	12p	1	1	0	0
	13p	1	1	0	1
	14p	1	1	1	0
	15p	1	1	1	1

Figure I: Combining coins

decode it as a binary number, you would interpret 100 in a completely different way and say it meant the number "four." *(Never mind exactly how you arrived at that conclusion for the moment.)*

This is what often causes problems – people are so used to dealing with their numbers in the normal way that 100 is always "one hundred" to them, and they can't make the shift necessary to decode it in binary as "four". Actually it is rather ambiguous. Presented with 100, do you interpret it as "one hundred" or "four"? Our rule will be, if you mean our usual way of dealing with numbers *(the hundreds, tens and units you learnt at school – or put it more formally, the denary system)* you write the number in the normal way.

If you wish the number to be decoded as a binary number you put the symbol % in front of it – 100 means "one hundred" while %100 means "four".

So far so good. We now have a marker (%) to warn us that we have to decode the number in a special way as a binary number. However, before you decode you need a rule for decoding – so how do you get the number "four" from %100? What's the rule?

If you were taught properly, you probably remember putting htu at the top of your sums. The letters were to remind you of the values of the columns: hundreds, tens and units. In 631, for instance, you've got 6 hundreds, 3 tens and 1 unit.

Understanding column value is a vital stage in children's learning – once they've mastered it they've "cracked" arithmetic. The point is, though, that the columns don't have to hold units, tens and hundreds and so on. In fact, in binary the columns contain units, twos, fours, eights and so on upwards, doubling for each new column. This means that %100 has a 1 in the fours column, and nothing in either the twos or the units, so %100 boils down to just four.

Once you get used to it, it's fairly straightforward. However, to make certain of the ideas involved, you might like to take a glance at the panel below, showing how binary works.

change, and we can use this table to encode numbers in general, not just coins. We call this method of encoding the binary system.

Remember, to show that we mean a binary number we precede it with %. So if you see, for example, %101 means:

```
     4  2  1
  %  1  0  1
     4  +  1  = 5
```

that is we add together the values of the columns containing 1. Look at row 5 of the table to check it. Similarly, %1101 would mean 13 in the denary system since

```
     8  4  2  1
  %  1  1  0  1
     8 + 4  +  1 = 13
```

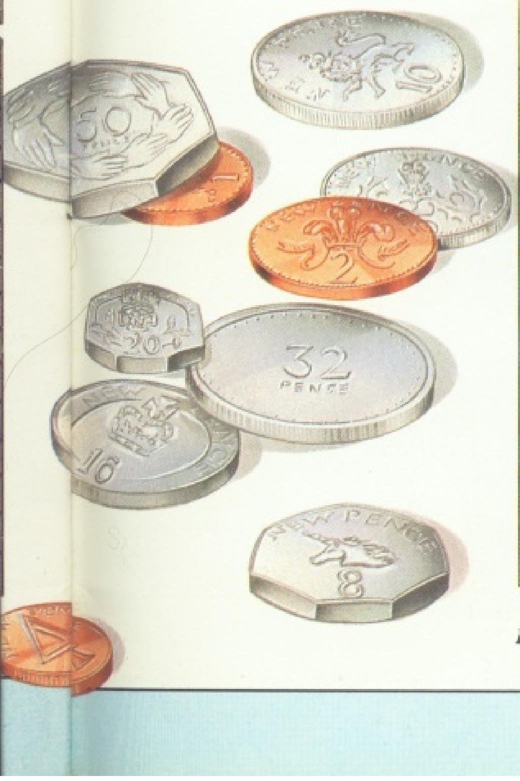
By now you should be able to work out for yourself why %100 represents four. From the table, or by using the addition method I've just illustrated, see if you can decode the denary values of the following binary numbers:

%1001
% 101
% 11
%1101
% 111

Actually the Spectrum provides an easy way for you to find the decimal, or denary, equivalent of a binary number, using the function BIN (for binary). If you replace the % sign with BIN, you can simply PRINT out the decimal value of a binary number. If you enter:

PRINT BIN 100

you'll see that the binary equivalent of %100 is indeed 4. That's enough for a first session. In the next part of *The Complete Spectrum* we'll meet bytes, groups of eight bits at a time, and see how to convert from denary to binary.



Denary Value	Column or Bit Values				Binary Value
	8	4	2	1	
1				1	%1
2			1	0	%10
3			1	1	%11
4		1	0	0	%100
5		1	0	1	%101
6		1	1	0	%110
7		1	1	1	%111
8	1	0	0	0	%1000
9	1	0	0	1	%1001
10	1	0	1	0	%1010
11	1	0	1	1	%1011
12	1	1	0	0	%1100
13	1	1	0	1	%1101
14	1	1	1	0	%1110
15	1	1	1	1	%1111

Figure II: Binary coding

It's all done by numbers

Making sense of machine code with MIKE BIBBY

MACHINE code is all about numbers – lots of them. More precisely, it's about lots of numbers, each of which is between 0 and 255 in value. Show me a machine code program and I'll show you a load of such numbers. For the moment forget about LD and JP or anything else you might have seen. Believe me, it's all done by numbers.

Let me explain. We're used to talking about a micro having memory, aren't we? Well a micro's memory is composed of lots of individual memory cells, as is our own brain. And, just like our memory cells, a micro's memory cell can only remember so much. In the case of the Z80, the cell can remember only one byte at a time – and a byte, you won't be surprised to learn, happens to be a number in the range 0 to 255.

If you haven't come across binary arithmetic an upper limit of 255 might seem a little arbitrary. A look at the article describing binary code on Page 30 of this issue should show you how it works.

Of course, we don't want to stop counting at 255. After all, if we were using binary to count the number of bytes in a micro's memory, we'd be limited to 256 (counting from 0 to 255) – not a very big computer.

So what the micro does is to have 65536 different memory locations, numbered from 0 to 65535 to store its data in. Why 65536? Well, in order to keep track of its memory bytes, the computer has to do some more wiring.

We've already seen that having eight wires would only allow us to keep tabs on 256 locations. What the Z80 does is to double up the number of wires to 16 – which then gives it 65535 as its largest number. Look at Table I if you don't believe me!

To get the value of these extra bits we just keep on doubling. 128 was the last one, so it goes 256, 512 and so on up to 32768. The top (higher valued) set of eight bits is called the high byte of the address – hi byte for short. The bottom (lower valued) set of eight is called the low byte of the address – lo byte for short.

If all the switches are on – that is, if all the bits were set at 1 – the number these two bytes would code is shown in Figure I.

Now do you believe me? The reason we've gone into so much detail is because, as I've said, machine code is all about numbers stored in the micro's memory. In fact machine code is mostly about moving those numbers (and hence the information encoded in them) around the memory of the computer – that is, from one memory location to another.

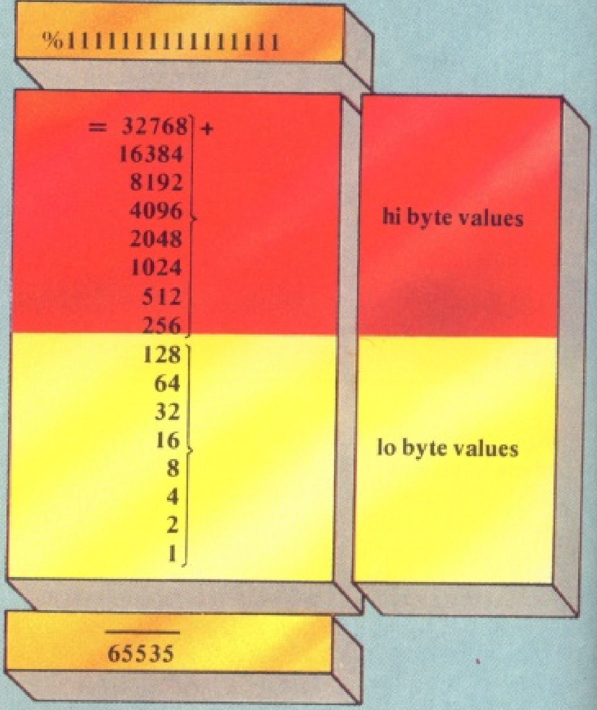


Figure I: Encoding the highest two byte number

For example, there's a large machine code program that actually runs your Spectrum. It's called the operating system, or firmware. One of its jobs is to print the familiar Sinclair Research message on the screen when you first turn on the machine. What happens is that the message is stored away in the micro's memory. When you switch on it copies the message from those locations into the memory reserved for the screen so you can see it. That is, the firmware machine code program moves the numbers that encode the message from one location to another.

This same sort of transfer of data occurs when you press a key. The firmware transfers the value of the key pressed from the location that remembers which key it was to the memory set aside for the screen. When you save a Basic program the firmware's own machine code program moves the contents of the memory where the Basic program is stored out to the cassette port.

It's all about moving bytes of data around. More formally, most of machine code is concerned with moving bytes of information from one memory location to another. If you're a realist, you'll probably

hi	bit	15	14	13	12	11	10	9	8
byte	value	32768	16384	8192	4096	2048	1024	512	256
lo	bit	7	6	5	4	3	2	1	0
byte	value	128	64	32	16	8	4	2	1

Table I: 16 bits explained

have guessed that there's a lot more to it than that. But have faith, most of what I'm telling you is true.

To investigate this movement of bytes further we need an analogy – in other words a meaningful lie. Suppose we have a tiny micro with only three memory locations. Figure II shows the sort of thing. It's fairly easy to wire them up so that the numbers can move from one location to another – just join each byte to every other byte (in the figure, by the way, I've only shown one of the eight wires for clarity). If you stretch your imagination you'll see that it looks a lot like a simple railway.

But suppose there were more locations, as in Figure III. You can see the layout's getting complicated. And when you consider that the Z80 addresses tens of thousands of such locations, you can see that we've got problems – the wiring's far too complex. Of course the answer is to stop giving each memory location its own direct lines to each and every other location. We'll do what railways do, and have junctions and branch lines.

Figure IV shows such a layout. Our six locations are

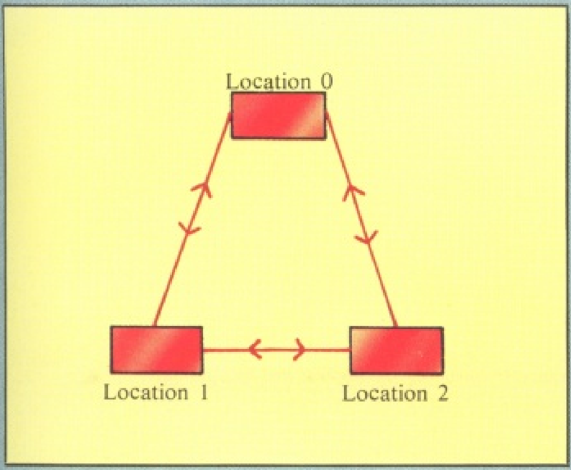


Figure II: Linking three memory locations

all connected via the major junction A. Everything passes through here. In fact there is such a memory location as A – a major junction through which traffic passes. It's deep in the heart of the Z80 and can hold one byte numbers. In computing jargon we call such a junction a register.

Now suppose you wanted to move a byte of information from memory location 0 to memory location 5. As you can see from the figure, you would go via register – that is, junction – A. You would do this by giving the machine two instructions:

1. Load register A with the number contained in memory location 0.
2. Load memory location 5 with the number that is in register A.

It's a sort of microelectronic pass the parcel. The data goes from location 0 to A, then from A to 5. It's always a two-stage journey. All traffic passes through A. In practice the actual layout is more like Figure V, but there's still no direct traffic. Everything goes to A first and then back out.

To stretch our analogy a little further, a major rail junction like A would have lots of facilities that other junctions haven't. It's the same with the Z80 – once you've got a number in A you can do all sorts of things with it! Also no rail designer worth his salt would depend on one major junction – there'd be too much congestion. He'd have other junctions. Similarly the Z80 has registers other than A for much the same reasons.

And as it stands our junction/register A doesn't have

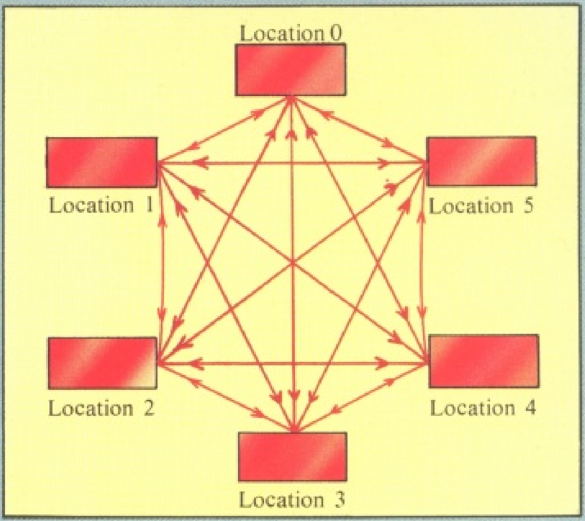


Figure III: The complexities of linking six memory locations

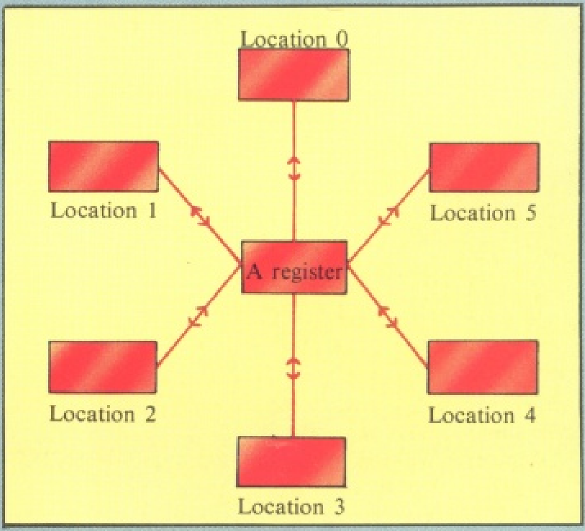


Figure IV: Memory locations linked by the A register

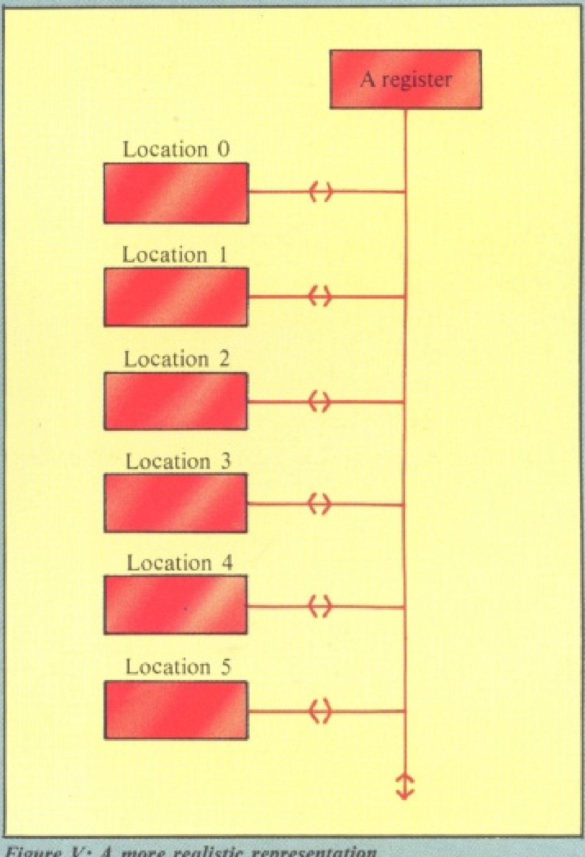
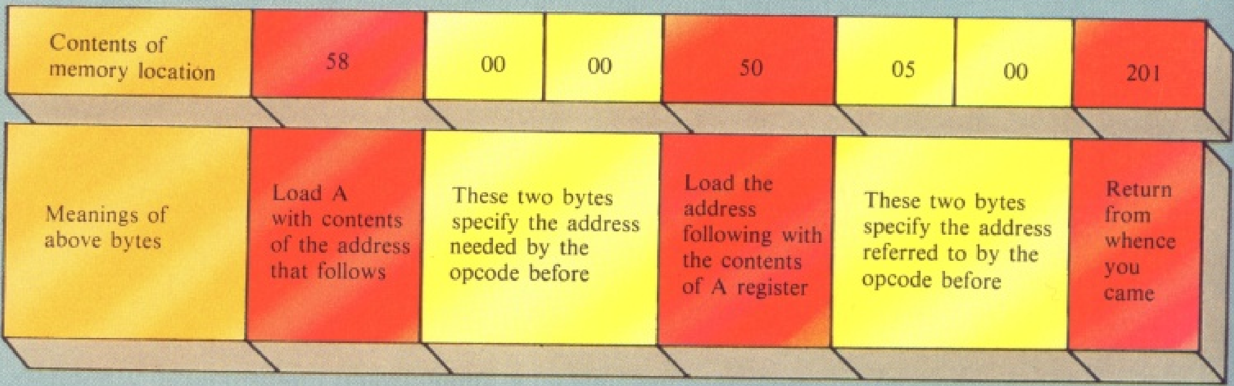


Figure V: A more realistic representation of memory/register linkage



Contents of memory location
58
00
00
50
05
00
201
Meanings of above bytes
Load A with contents of the address that follows
These two bytes specify the address needed by the opcode before
Load the address following with the contents of A register
These two bytes specify the address referred to by the opcode before
Return from whence you came

Figure VI: A simple machine code program explained

all that much capacity – just one eight bit number. This could be limiting if we wanted to deal with larger numbers, such as we use to label memory locations. Well the Z80 has got registers to handle these, too, as we'll see later on in the series.

By now I think I've convinced you that machine code's all about moving numbers around in memory. But how does the micro know what to do? How do you tell it to move these numbers, and where you want them putting? The answer's simple – you give it a list of numbers stored in memory! I'm not joking, honest.

The program itself is just a sequence of bytes in memory. The bytes have meaning to the Z80, you see – it's a sort of code, machine code in fact. All you do is point your Z80 at the first byte and say go. It then moves along the list of bytes doing what it's told.

Let's have a look at what this means in the context of the little program we discussed earlier – the one that transferred one byte from location 0 to location 5. The actual string of bytes we need is:

```
58 0 0 50 5 0 201
```

I've written the numbers in decimal, as that's what we're used to – of course the micro reads them in binary. Figure VI explains what it's all about. When we point the Z80 at the location of the first byte of our program and tell it to go it knows that first byte is going to tell it to do something. The fancy name for this sort of "command" byte is an opcode – short for operation code.

Now 58 is an opcode that tells the Z80 to load register A with the contents of a particular location in memory. From the opcode itself, the Z80 knows that the address of this memory location will be contained in the two bytes directly following the opcode. (Remember you need two bytes to specify addresses.)

So having understood the meaning of the opcode, the Z80 moves its attention along to these two bytes and works out the address they refer to (in this case, location 0). It then copies the contents of that address into the A register. The Z80 has finished with the first three bytes and has done all the first opcode instructed it to. It now turns its attention to the fourth byte, which it knows must be an opcode since it has finished its previous task.

This time the byte is 50, which tells the micro to load the memory location specified in the next two bytes with the number in the A register. (In a sense, this is the mirror image of the last opcode. That loaded the A register from a memory location. This opcode loads a memory location from A.)

So having worked out what the opcode contained in the fourth byte wants it to do, the Z80 turns its attention to the next two bytes along, works out the address they store and copies the A register into that location.

Having finished that instruction, which used the fourth, fifth and sixth bytes, the Z80 then moves on to the seventh and last byte to find its next opcode. The seventh byte contains 201, the opcode for return – which tells the Z80 to go back to where it was before it started or, as the jargon has it, called this progam. This works in much the same way as RETURN does in a subroutine, causing the micro to rejoin the main flow of the program.

Notice that you don't need any extra bytes after this opcode to tell it where to return to. When this routine was called the Z80 carefully stored where it was up to for future reference – as does a Basic program when it meets a GOSUB.

By the way, you may have noticed that the two bytes specifying location five are not 0,5 as you might expect, but 5,0. I don't want to go into this too much yet. Suffice it to say that the Z80 likes to know the lo byte of an address before it receives the hi byte.

Let's have another look at the machine code program we've developed. I'm going to split each instruction – that is each opcode and its data bytes – onto a separate line.

```
58 0 0
50 5 0
201
```

Doesn't make immediate sense, does it? Our brain is much more adept at making sense of words than numbers. Have a look at the program in a new form, that uses "words":

```
LD A,(0) 58 0 0
LD (5),A  50 5 0
RET       201
```

The symbols on the left hand side are mnemonics. LD stands for LoaD and RET for RETurn.

The translation is as follows:

LD A,(0)	*LoaD the A register with the contents of memory location 0.*
LD (5),A	*LoaD memory location 5 with the contents of register A.*
RET	*RETurn to the program that called the machine code in the first place.*

You can get special programs called assemblers that let you type in your routines in these more meaningful mnemonics and then translate them into machine code, but they're a luxury we'll be doing without for a while.

● ***In the next part of this series we'll be looking at hexadecimal numbers and running our first machine code programs.***

The man himself

Sir Clive Sinclair chats to Mike Cowley

IT was 1940. Hitler was busy goosestepping his way across Europe in the direction of Britain. Yet for Thora and Bill Sinclair at home in Richmond, Surrey, it was still time to nurse high hopes for their recently arrived first born, Clive.

They weren't to be disappointed. For their son was to survive the Blitz – though the family home was bombed – to find fame and fortune as the man behind the Spectrum, the most successful home computer of all time.

Nicknamed "Bambino" by his parents – from the letters BAMB on his birth certificate – Clive soon showed signs of being an exceptional child. By the time he was three he was able to hold his own in conversations with the most articulate of adults.

At the age of 10, his teachers of the time simply threw in the towel confessing they had taught him all they knew. It may well have proved some consolation to them when he subsequently became chairman of Mensa, the organisation open to those with only the highest of IQs.

The young Clive displayed an amazing ability to solve the most complex of mathematical problems with ease, and apparently intuitively. For, more often than not, he was unable to explain how he had come up with the right answer.

Come his 17th birthday, he had been accepted by fellow pupils and masters alike at St. George's School, Weybridge, as an accomplished inventor, having developed a radio the size of a lighter.

Soon afterwards, a journalist was to correctly forecast that Clive Sinclair would become a household name. Yet, as far as his parents were concerned, he still wasn't anything special. After all, the family could already boast its own celebrity in the form of Clive's first cousin who fenced for England.

Not that Clive was without some athletic ability himself. He did go on to run – and finish – in both the New York and London marathons. But that was later. So for the time being one "name" in the family was enough.

In fact mother Thora, a feet on the ground Devonshire lass, refused to accept that Clive was "different" until he was knighted by the Queen for almost single-handedly establishing the home computer industry.

Up until that time on every occasion she had seen her son's name in the paper she told her husband: "I can't understand why they write about him".

Yet here was the creative genius who had gift wrapped home computing and unceremoniously handed it over to the man on the street. And for this he

had already been named both "The Guardian Young Businessman of the Year" and "Computing's Person of the Decade".

However for his mother the first day he became "special" was when he received his knighthood in the Queen's Birthday Honours List of 1983.

"It was as though when Her Majesty touched Clive on the shoulder, his mother suddenly realised he wasn't ordinary", recalls Bill Sinclair.

"But such is Clive's shy nature, we even had to hear about that honour from another relative who had read it in the paper". Today as the millionaire boss of Sinclair Research, Sir Clive has emerged as the dominant figure – if somewhat reluctantly – of the UK home computer industry.

For so far he is the only person within that sector to have been elevated to celebrity status by the media. Whether it be related to the booming sales of the Spectrum or the ill fated C5 electric car, he is rarely out of the headlines.

Even the redoubtable Chris Curry, co-founder of Acorn and a man who reputedly enjoys the high life, only made the front pages courtesy of a heated exchange of words – some say blows – with Sir Clive in a hostelry.

Those who are closest to Sir Clive – particularly his family – insist that the cloak of publicity sits uneasily at best on his shoulders. It has been reported that he has been seen to wince more than once when reading of his supposed exploits in certain gossip columns.

The fact of the matter seems that he would still opt for the role of back room boffin any day.

Balding, bespectacled and bearded, with piercing eyes, he could have well been culled from Central Casting in Hollywood to play the role of a mad scientist in a B movie.

Mad he is not. Scientist he certainly is. An innovator who has been carried kicking and screaming into the limelight, yet shrewd enough to capitalise on it for the good of his company.

Sir Clive holds aloft the one millionth Spectrum during celebrations with the workers at the Scottish assembly plant

At his London office – not a stone's throw from Harrods and furnished like a Habitat window display – Sir Clive reflected on his career to date.

In particular, he took time out to discuss his reasons for being convinced there was a home computer market where one hadn't existed before.

"It was always my belief – and still is – that there is a very clear hobbyist market out there", he says. "To my mind, at a price, you could sell to the man in the street who had never dreamed of buying a computer before.

"With it he could learn to program if he wanted or simply have a bit of fun with it. That was the case when we brought out the Z80 which cost £100 when there was nothing else around under £500.

"And this was the first time such a concept had been dreamed up anywhere in the world. Fortunately it worked".

Why were the early ZX machines so successful? "Apart from the fact that the man on the street could afford them, a lot of people felt, rightly or wrongly that computers would be a very big feature in their lives and those of their children. So they wanted to get to know about them without being afraid of them.

"There is also a great deal of excitement of having a machine at home and seeing it do marvellous things".

But has the home computer boom ended? Despite the doom and gloom reports carried by many newspapers of late, Sir Clive insists this is not the case. "The industry will remain healthy as long as it provides the man on the street with what he wants at a price he can afford", he says.

"You will discover when they have totalled up the sales figures for 1985 they will be very similar to the previous year – if not better.

"So where is the problem? Certainly not for the Spectrum, which will have 40 per cent of the market, more than that of the next three machines combined together".

Nor does he accept the claim that there is a high fall out rate among Spectrum or any other micro users for that matter. He finds the idea of micros gathering dust under the stairs as being, in the main, farfetched.

"Our surveys actually show that the drop out rate is very low. Not only do people use their machines when they buy – or have them bought for them – but they continue using them – very much so".

And Sir Clive is keen to point out that the Spectrum is far more than being just a games machine. "Naturally because a lot of people play games on them – and why shouldn't they? – people tend not to think of them as serious machines. But a great number of people buy them to learn and remain serious users.

"Nor should anyone overlook the fact that 10 per cent of all Spectrums sold are used for business purposes. And this makes it by far the best selling business machine in the UK. This is something I am very conscious of and the company is going to pursue this erea".

What of the future for the Spectrum? "The machine has come out of the recent problems that have beset the computer market better than any other", he says. "It has in fact pushed up its market share.

"Now a number of manufacturers have announced that they intend to pull out of this section of the market. Once this happens our sales will rise further.

"The British public's love affair with the home computer is far from over. Sinclair Research and the Spectrum will make sure of that.

"As a result, we will continue to lead the world in home computing. What could be more fitting for the country where it all started?"

NO FAULTS

THE SHOW JUMPING SIMULATION GAME THAT'S A CLEAR WINNER

Horsemanship...Computermanship...It's got the lot.

The stadium stills, the crowds wait, quietened as the tension builds to breaking point — only 15 seconds left on the clock. Rider releases horse in a flurry of movement. Has he set the right angle? Are they approaching too fast? Out over the first, pops the second and eases over the third. A superb treble! A magnificent finale to a nail biting competition.

Never before has the instinctive relationship between man and horse, the skill and tension of competitive riding been so well represented on computer. An outstanding test of skill, an interesting setting for individual challenge or multi-player competition.

1st choice for Spectrum 48K and CBM 64/128

£7.95

Alligata Software Ltd.
1 Orange Street, Sheffield S1 4DW.
Telephone: (0742) 755796