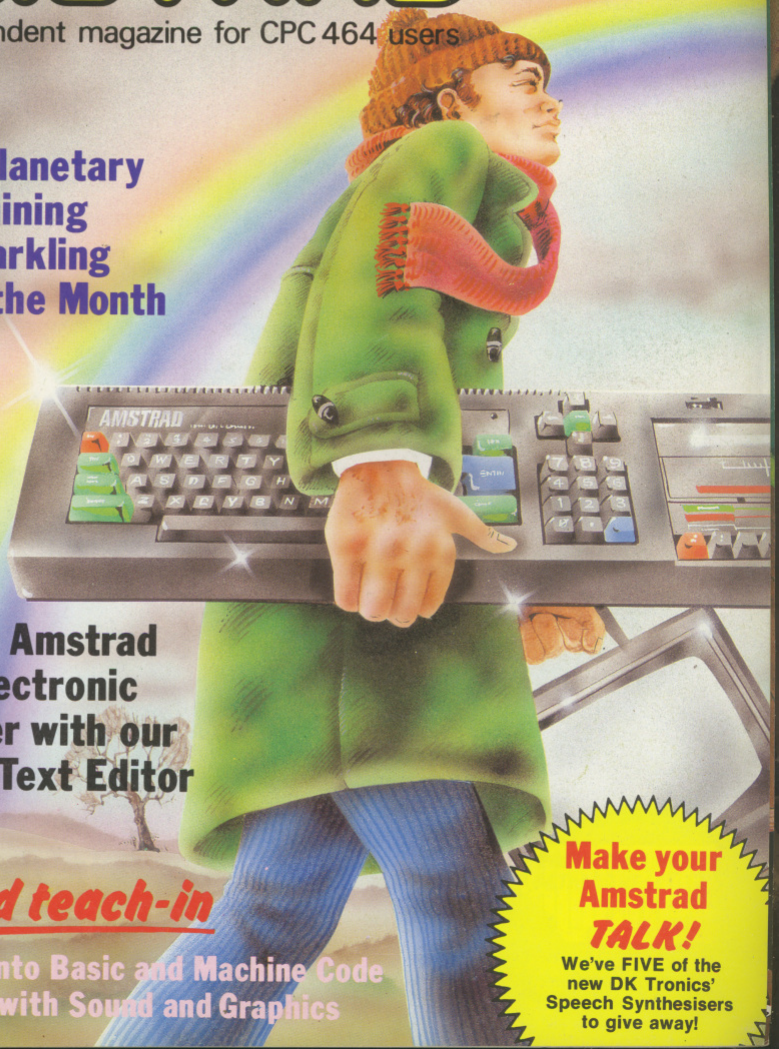
A Database Publication

Computing with the AMSTRAD

No. 2
February 1985
£1

The independent magazine for CPC 464 users



Go interplanetary crystal mining in our sparkling Game of the Month

Turn your Amstrad into an electronic typewriter with our powerful Text Editor

Amstrad teach-in

Delve deep into Basic and Machine Code
Get to grips with Sound and Graphics

Make your Amstrad TALK!

We've FIVE of the new DK Tronics' Speech Synthesisers to give away!

ANIROG

The Name For Quality And Innovation

Flight Path 737

ADVANCED PILOT TRAINER

AMSTRAD

FLIGHT PATH 737—3D TIME TREK

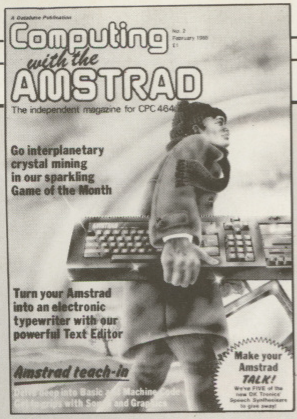
ZODIAC——HOUSE OF USHER

£6.95 Each

TRADE ENQUIRIES: ANIROG SOFTWARE LTD. 29 WEST HILL DARTFORD KENT (0322) 92513/8
MAIL ORDER: 8 HIGH STREET HORLEY SURREY 24 HOUR CREDIT CARD SALES HORLEY (02934) 6083
PAYMENT BY CHEQUE P.O. ACCESS/VISA 50p POSTAGE & PACKAGING

MICROPOWER MICROPOWER MICROPOWER

SUPERB SOFTWARE FOR THE AMSTRAD
SUPER-FAST LOADING TIME commodore 64
electron
B.B.C. MICRO
GHOULS
Run through the creepy mansion dodging ghostly ghouls and bouncing spiders. Leap over poison-smeared spikes, scamper along moving platforms and contracting floorboards, and use powerful springs to propel you onto overhanging ledges. Four screens.
Amstrad and Commodore versions £6.95
BBC and Electron versions £7.95
BBC and Commodore Disk price £9.95
Amstrad version
WATCH OUT FOR OUR NEW PACKAGING AND CATALOGUE
MICRO POWER
MICRO POWER LTD., NORTHWOOD HOUSE, NORTH STREET, LEEDS LS7 2AA TEL: (0532) 458800
SELECTIVE BRANCHES OF BOOTS, CO-OP, MENZIES, W. H. SMITH, WOOLWORTHS AND ALL GOOD DEALERS.
AUTHORS! WE PAY 20% ROYALTIES!



Vol. 1 No. 2 February 1985

Managing Editor: **Derek Meakin**
Features Editor: **Peter Bibby**
The A Team: **Mike Bibby**
Alan McLachlan
Kevin Edwards
Production Editor: **Peter Glover**
Layout Design: **Heather Sheldrick**
News editor: **Mike Cowley**
Advertisement Manager: **John Riding**
Advertising Sales: **Margaret Clarke**
Editor in Chief: **Peter Brameld**

Editorial: 061-456 8835
Administration: 061-456 8383
Advertising: 061-456 8500
Subscriptions: 061-480 0171
Telex: 667664 SHARET G
Prestel Mailbox: 614568383

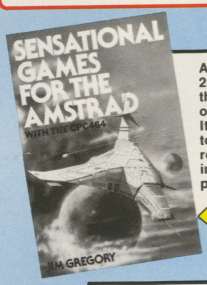
Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

Subscription rates for
12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

ABC Member of Audit Bureau of Circulations

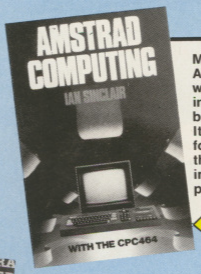
Contents

BOOKSHOP



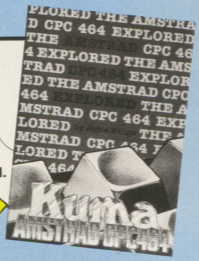
A choice selection of 24 rip roaring games that'll give you hours of fun.
It also shows you how to incorporate ready-made routines into your own programs.

£6.95



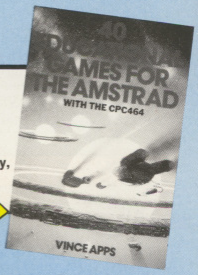
Make the most of your Amstrad's potential with this information-packed book.
It's full of easy to follow information that's certain to improve your programming.

£6.95



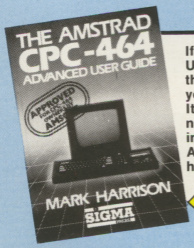
Starting where the Users Instructions leave off this invaluable book explores the Amstrad's sound, graphics and assembly language capabilities to the full.

£8.95



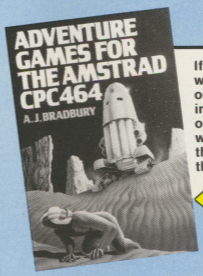
Put your Amstrad to good use with these forty games, full of educational value. Subjects include languages, geography, maths and science.

£6.95



If you're baffled by the Users Instructions, this is the book for you.
Its easy-to-follow no-nonsense introduction to Amstrad Basic is highly recommended.

£7.95



If you've ever wondered what goes on behind the scenes in adventure games, or thought about writing one, this well thought-out book is the one for you.

£7.95

ORDER FORM

All prices include post and packing

Please enter number required in box

		£
☐	**40 Educational Games for the Amstrad by Vince Apps** (Granada)	£6.95 .....
☐	**The Amstrad Program Book by Peter Goode** (Phoenix)	£6.95 .....
☐	**Adventure Games for the Amstrad CPC464 by A.J. Bradbury** (Collins)	£7.95 .....
☐	**Amstrad Computing by Ian Sinclair** (Granada)	£6.95 .....
☐	**Sensational Games for the Amstrad by Jim Gregory** (Granada)	£6.95 .....
☐	**The Amstrad CPC464 Explored** (Kuma)	£8.95 .....
☐	**The Amstrad CPC464 Advanced User Guide by Mark Harrison** (Sigma Press)	£7.95 .....

I enclose my cheque/P.O for £.....

Name .....

Address .....

.....

.....

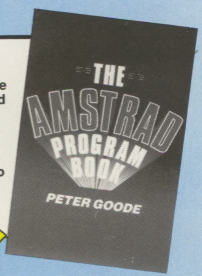
Post to: Database Publications, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

Cheques made payable to Database Publications Ltd.

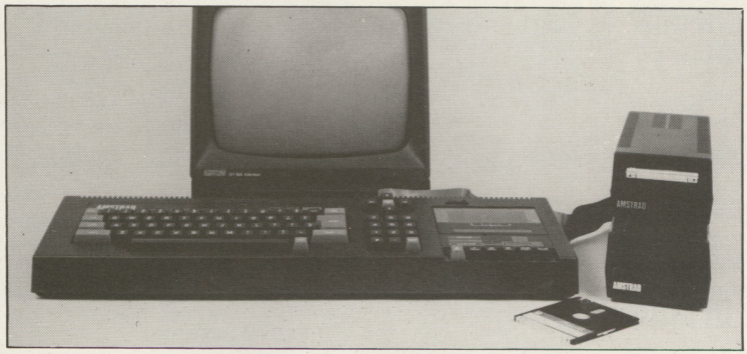
These prices apply to UK readers only. Overseas prices on request.

From adventures to asteroid storms, dungeons to dice, the 31 programs featured in this book are guaranteed to give you plenty of amusement – plus valuable insights into programming techniques.

£6.95



WATCH THIS SPACE FOR NEW BOOKS EVERY MONTH, TO DEVELOP YOUR KNOWLEDGE OF THIS EXCITING MACHINE STILL FURTHER!



Amstrad gets top game

TWO DISC DRIVE SYSTEMS ON WAY

AMSTRAD owners will soon be able to access a much wider range of software. The firm's own just released 3in disc drive is to be augmented by a plug-in 5¼in second drive from Timatic Systems.

ACORNSOFT's top selling game Elite is now being converted for the Amstrad.

Elite is a 3D spaceflight simulation within a nonstop galactic adventure. According to Acornsoft, it is "a fantastic voyage of discovery and adventure . . . the ultimate test".

Intergalactic trading exposes players to danger from all sides. The profits buy better defences which help the traders survive and eventually win the coveted rank of Elite.

The game took two years to develop in association with Cambridge undergraduates Ian Bell and David Braben.

But Acornsoft is not doing the Amstrad conversion itself. Instead, it has sold the rights to Firebird, the software arm of British Telecom, for an undisclosed sum.

The actual programming is being done once again by Barben and Bell who expect to finish around May.

Said a Firebird spokesman: "It's quite a complex job because the game was written for a 6502-based machine – the BBC Micro. Converting it to run on a Z80 machine is going to take a few months".

The companies have been cooperating on the project since Amstrad confirmed it intended producing only a 3in drive for the CPC464.

Timatic spokesman Nick Young said: "We now have available a 5¼in drive as an alternative second drive to the Amstrad 3in model.

"We've been waiting in the wings for Amstrad to bring out their disc drive. Ours is all ready to go".

As most computer software in the past has been produced on the 5¼in format, the Timatic drive means that users will soon have access to many more programs, particularly in the business area.

"A large range of CP/M programs has been made available ready for the launch of the Amstrad disc system, and can be supplied on either the 3in or 5¼in formats", Young told *Computing with the Amstrad.*

"All programs supplied by Timatic will be configured to run on the Amstrad. We also have the ability to transfer other programs to disc if required".

Timatic will issue free with any disc system bought from them a disc which contains various utilities and help files.

This is an on-disc tutorial which gives the user help in such subjects as Microsoft Basic, ED, ASM and others.

"Timatic can also put Amstrad owners in touch with the CP/M user group which has available to members more than 30 mbytes of public domain software", said Young.

These include JT Pascal, Forth, several Basic dialects, disassemblers and other CP/M utilities.

The firm also carries the full range of Amstrad cassette software.

"We're looking into CP/M modem software and hope to have a system available shortly to run with the disc system", said Young.

"And we welcome any ideas from Amstrad users for hardware and software add-ons which we could help to produce and market".

Policy pays off

AMSTRAD's policy of putting its CPC464 micro out to software houses well before the launch is now paying off handsomely.

Launched last summer with a good choice of games and software, the Amstrad is already established as a market leader.

Amsoft – the parent company's own software house – has 85 games on offer plus 10 educational programs and an assortment of business software.

And there are even more offerings from other publishers, many of whom have converted Spectrum games.

At least one graphics pad is on the way – from Touchmaster of Glamorgan. Says managing director Kevin Stephens: "We are working on an Amstrad interface now, and should have the product ready shortly".

Sorcerers

AMSTRAD owners are promised a version of Swords and Sorcery – an adventure that took PSS programmer Mike Simpson more than 18 months to devise.

The game contains 86 monsters, 635 locations and almost 2,000 objects – and, says Simpson, "in excess of four million ways to be insulted".

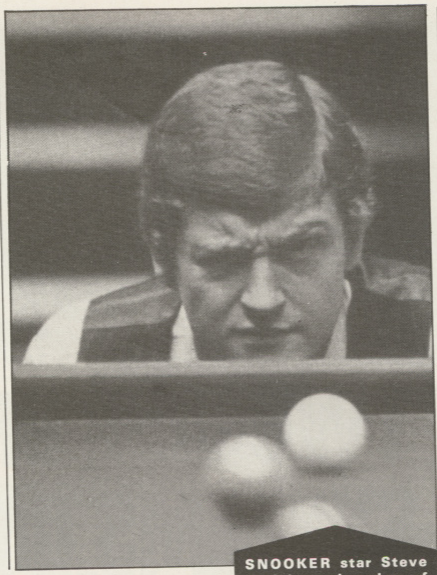
The program is run in real time and is fully menu driven with keyword classification.

and dragons

THE Saga of Erik the Viking by Terry Jones of Monty Python fame is now available for the CPC464 from Mosaic Publishing.

The illustrated adventure with dramatic text has more than 200 locations and features Viking settlements, wizards, dragons and giants.

Price is £9.95, including a booklet containing stories from the original book.



SNOOKER star Steve Davis has endorsed one of the first fully machine-coded games with Amstrad compatibility to be produced by an independent software house.

CDS devised its own Amstrad interface to produce Steve Davis Snooker, which retails at £7.95.

The company will also release its new adventure trilogy called The Artemis Quests in an Amstrad version. Part 1, called Castle Blackstar, will be available soon.

Your CPC can answer back

SAY-hello to the talking Amstrad – it can now answer you back.

Dk'tronics has brought out a speech synthesiser with an almost infinite vocabulary for the CPC464 using the SLO/256 chip.

A text-to-speech converter allows everything you want spoken to be entered in normal English without special control codes or characters.

The speech output from the SLO/256 is mono and is, directed to both speakers. The interface has a built in stereo amplifier which uses the stereo output on the back of the micro, improving sound quality and volume over the internal speaker.

The system can look at individual letters that make up each word and compare their relative position to each other before deciding on the appropriate sound.

"We've perfected what I consider to be the best synthesiser on the market, one which – within the limits of the allophones – produces realistic speech" says dk'tronics marketing manager Roger Barnard.

● *See Contest Page 61.*

Towering success..

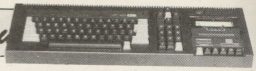
A FEW years ago Amstrad revolutionised the hi-fi market by putting a tuner unit, amplifier, tape deck and record player in one box.

It was the birth of the hi-fi tower and Amstrad went on to win a 45 per cent share of the market.

Now Amstrad has done a similar thing with its micro.

As Henry Ford once said, "you can have any colour you like as long as it's black". You can have any Amstrad complete micro system you like as long as it's either colour or green screen.

Amstrad says its policy is proving correct – the proof being the massive demand for the CPC464.



CBTV presenters Paul Henley (left) and Paul Shearer with "George", their Amstrad CPC464

Amstrad tots up the TV Top Ten

A CPC 464 has joined the team of presenters on the popular children's television programme CBTV.

And although there are no plans for it to take over completely from hosts Paul Henley and Paul Shearer it will play a major role in the Thames Television series.

Every Tuesday its colourful graphics are being used to announce the week's top ten best selling pop records.

And it screens short animated programs submitted by creative young viewers.

Christened George by Paul Shearer "in a flash of inspiration", the Amstrad earned its chance to achieve TV stardom by winning an audition at the recent PCW show.

Thames sent along two youngsters, 15-year-olds Vincent Graff and Matthew Gerry to run the rule over the range of home micros and select one for CBTV. They picked the CPC464.

Said Paul Shearer: "As well as screening the top ten and viewers' programs, George will probably be used for news headlines and our filing system.

"We also hope to get a modem so that George can interact with the outside world and link up with computer-owning viewers".

NOW CPC464 GETS FORTH

SKYWAVE Software has released two "firsts" for the Amstrad – Ams-Forth 1.11 and an expandable RS232 interface.

As a prelude to the release of the Multi-Forth 83 sideways ROM, Skywave's Amstrad version of Fig Forth on cassette features full support of the sound, colour and graphics.

It is fully compatible with the cassette interface, allowing both fast and slow recording speed, built-in editor and RAM disc simulation for easy software development, and built-in stack display utility for learning or debugging.

When the Multi-Forth 83 ROM for the CPC464 is released the tape can be traded in against the ROM.

Free with the Ams-Forth tape is Spell, a Forth decompiler allowing the user to understand and change Ams-Forth. Price including manual is £15.

Skywave's plug-in card contains a full RS232 interface using the 8251. It opens the way for lots of features, including a crystal-controlled baud rate generator giving a choice of eight different baud rates between 75 and 9600.

Separate transmit and receive baud rates allow Prestel modems to be driven, and a sideways ROM can be fitted as an upgrade to give extra RS232 features.

An expansion card can be added to allow the use of up to seven extra sideways ROMs or up to seven 8255 parallel ports, giving a total of 168 bits of parallel I/O.

Price of the interface, a cassette containing driver software and a manual is £59.

Help the birds

THE game voted Microdealer UK Educational Program of the Year for 1984 is now being produced for the Amstrad.

Osprey, from Bourne Educational Software, was devised in conjunction with the Royal Society for the Protection of Birds.

It encourages interest in bird watching and wildlife preservation by making the player responsible for protecting ospreys from poachers and other dangers.

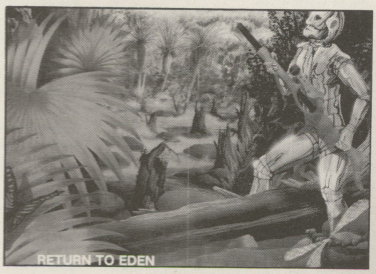
The program comes with a 32 page colour booklet outlining the history of the osprey.

Kim goes on

RETURN to Eden – the second part of the Silicon Dream trilogy – is now available for the Amstrad.

The colourful 240 screen program from Level 9 Computing picks up from where Snowball left off, continuing the adventures of Kim Kimberley and his interstar transport.

Price is £9.95 with a disc version promised soon at £11.95.



RETURN TO EDEN

Second in MIKE BIBBY's helpful series for beginners

THIS month we are going to begin writing our own programs. Nothing spectacular mind, but enough to give a quiet glow of satisfaction. Firstly, let's discuss what we did last month.

We learned that to "talk" to the micro we had to speak to it in a language it already understood, called Basic. We also learned how to get the Amstrad to do sums for us and to print out messages, or strings, as they are known.

One Basic word we used quite frequently was PRINT, which instructs the micro to write or print things out on the screen. For instance, to do the sum 4+4 we typed

PRINT 4+4 [Enter]

where [Enter] means you should press the key marked ENTER. This sends the message we have typed into the micro. Hopefully it then responds by printing the correct answer, 8.

Notice something – there's a space between PRINT and the first 4. Try doing the sum without that space. That is:

PRINT4+4 [Enter]

You'll find you get a "Syntax error" message. You see, the command PRINT is known as a keyword – Basic keeps it to itself and treats it specially. If you like, it's already in the micro's vocabulary.

However, to recognise you mean PRINT to be used as a keyword, you need to have a space after it – and before it if it's in the middle of a line. If not, the micro gets confused, as above. After all, it's never heard of PRINT4.

The technical term for the space marking the end of the PRINT is a delimiter – it shows the limit of the word PRINT.

While you're at it, try

print 4+4 [Enter]

You still get 8.

This may not surprise you, but it should – most other micros would not recognise the lower case "print". The CPC464 does.

Just as we can do addition, so we can do subtraction, multiplication and division – the symbols for which are –, * and / respectively. Notice, multiplication is *, not X. Also, divide isn't ÷, it's /. Although the Amstrad will print the more traditional ÷ symbol, it doesn't know that it means division: you have to use / for that.

We also learned last month that to print out messages, we had to surround them with quotes – as we do when we write what someone is actually saying.

So, to print out the message GOOD MORNING on the screen we type:

PRINT "GOOD MORNING" [Enter]

which, as you'll see if you try it, causes

GOOD MORNING

to be written on the screen.

Of course, on the CPC464 we have lower case, or small letters, as well so we could have entered the above as

print "good morning" [Enter]

This time:

good morning

will appear on the screen.

Notice the difference between the last two – the first was all upper case, the second all lower case.

In both instances the Amstrad recognised the command "print", proving once more that the Amstrad doesn't mind its keywords being in big or small letters – you can even mix them, as in PrInT.

It's different with the message in quotes, though – the first time the resulting print-out was in upper case, the second in lower case. You see, the Amstrad prints out exactly what's between the quotes: capitals and lower case letters are shown exactly as they are, so the two messages vary.

By the way, you don't need the space between the end of PRINT and the beginning quote marks of the message. This is because the quotes act as delimiters – that is, they mark the ends of keywords – just as spaces do.

In this series of articles we'll tend to keep to capital letters for our keywords so they stand out more clearly.

So far we have given the micro one instruction at a time, which it carried out immediately after we pressed Enter (assuming we'd typed it correctly).

Sometimes, though, we want to give the micro a series of instructions and then tell it to carry them out. For instance, suppose we want the message:

PROGRAMMING
IS
EASY

to appear on the screen. With our step-by-step method, we would have used:

PRINT "PROGRAMMING" [Enter]
PRINT "IS" [Enter]
PRINT "EASY" [Enter]

But, as you'll see if you try it, this doesn't produce the required effect, since each successive instruction spoils the layout.

We need to give the micro the instructions so that it:

1. Prints out PROGRAMMING
2. Prints out IS
3. Prints out EASY

in sequence, without stopping to ask us what to do next. Such a sequence of instructions is called a program. Notice also that the sequence is numbered – after all, the micro needs to know the order to carry them out in.

Now let's write a program to print out

PROGRAMMING
IS
EASY

We were on the right lines with the first attempt, but this time, let's try numbering our instructions as we enter them.

First of all, we shall enter Mode 1, a text-only mode with:

MODE 1 [Enter]

Now type:

NEW [Enter]

New is a Basic keyword that clears out the micro's memory. If you don't do this the program you are typing in might get jumbled up with a previous one – you'll see more clearly how this can happen later.

You probably think that you haven't got a program in at the moment, but use NEW anyway, because it is possible that you might have entered a line or two by chance.

'The Amstrad prints out exactly what's between the quotes'

Then type:

10 PRINT "PROGRAMMING" [Enter]

Notice two things:

- The first instruction is number 10, not number 1. In computing we tend to number our instructions in steps of ten for reasons that will become blindingly obvious later. We call the number of an instruction its line number.
- The micro didn't immediately carry out the instruction – it didn't print out PROGRAMMING after we pressed Enter. This is because of the line number. It tells the micro that what follows isn't to be done immediately but is to be remembered for later as it is just one in a series of instructions. I'll prove that the micro actually does remember it in a moment.

Now type:

20 PRINT "IS" [Enter]
30 PRINT "EASY" [Enter]

What I'm going to ask you to do next should test your faith in me! Clear the screen by typing:

CLS [Enter]

All your typing should have disappeared, but don't worry – your work hasn't been wasted. Because of the line numbers, the micro has kept a list of your instructions in its memory. To see the list, type:

LIST [Enter]

and your program should reappear. We'll call it Program I:

```
10 PRINT "PROGRAMMING"
20 PRINT "IS"
30 PRINT "EASY"
```

Program I

If you've typed your keywords in lower case, you'll notice when you LIST them, they appear in upper case. This can be quite useful in looking for mistakes – you know all your keywords should be in capitals. If they're not, you've done something wrong!

An important point coming up now. We have entered a program (a numbered sequence of Basic instructions) into the Amstrad CPC464's memory and have got the micro to display those instructions with LIST. We have not, however, told the micro to do these instructions.

It's like having written a shopping list – you still have to go to the shops and turn your list into reality.

So to get the micro to actually do, or as we say, run the program in its memory, we type:

RUN [Enter]

and, if we've typed it in properly, we should see printed out:

PROGRAMMING
IS
EASY

If you've managed it, congratulations on running your first program.

If not, don't worry, it's probably some simple error. List your program and look for the mistake. You might actually have a message telling you that there is an error in a particular line.

What we're about to do next, although it assumes that you have been successful so far, will in fact ▶

show you how to correct your mistakes.

Now let's try to alter our program so that it prints out:

PROGRAMMING
IS
SIMPLE

If you look back at the first program you will see that you need to alter line 30.

Changing line 30 couldn't be simpler – just type in the new line 30, remembering to start with the line number 30, then press Enter. The latest version will replace the old version in the micro's memory.

To demonstrate this, type:

30 PRINT "SIMPLE" [Enter]

and then:

LIST [Enter]

You should obtain Program II, which is:

```
10 PRINT "PROGRAMMING"
20 PRINT "IS"
30 PRINT "SIMPLE"
```

Program II

An examination of this listing should reveal that the new version of line 30 has indeed replaced the old one. (Notice also that we didn't give LIST a line number – we wanted the CPC464 to do it immediately.)

As a final proof that our amendment has been accepted, type:

RUN [Enter]

You should now get the revised message.

You can use this technique to correct mistakes in your programs. For example, if you accidentally typed line 10 as:

10 PINT "PROGRAMMING"

then, when you tried to run it you would get the message "Syntax error in 10". (Note that you don't receive this message when you first enter the line, only when you try to run it.)

To rectify such mistakes, simply retype the correct version of line 10 and press Enter to send it into the micro. The correct version will replace the faulty one.

There are more sophisticated ways of correcting, or editing, a line, but they can wait for a while. For the moment we shall simply retype the line, with its line number, and press Enter.

Of course, if you notice a mistake while you are entering a line, use the delete key to erase it, then continue typing from that point.

So far I have given you just two programs to run. However, using these as models, you can print out virtually any message you want on the screen.

Just use line numbers in increments of 10, each line printing out part of the message you want out on the screen, by enclosing it in quotes after PRINT.

An important point about this series is that I'm going to give you lots of example programs to type in. Virtually all of them have two things in common:

- *They make vital teaching points (otherwise they wouldn't be there in the first place).*
- *The output – that is, what appears on the screen – is trivial in content and in many cases there are far easier ways of doing it.*

Programming is a skill like driving – you can only improve by doing it, not reading about it. Please carry out the examples, however simple or obvious they may seem to you.

Also, and this is far more important, I want you to go beyond the programs – try to alter, adapt and extend them, just to see what happens.

Adopt an experimental approach and a healthy scepticism for my pronouncements. If you are wondering whether something will work, go ahead and try it – you can't hurt the micro from the keyboard, so let your imagination run riot.

You'll learn far more from your own examples than you will by merely echoing mine. And the good thing is that you get such prompt feedback from a micro. If what you write isn't acceptable you'll soon get an error message.

So what I'd like you to do now is to spend a good time writing simple "message" programs for the micro to run. For some reason, in my experience in computing classes the messages tend to become quite scurrilous. There's one thing I've never been too sure of – is it slander or libel when it appears on a VDU?

Remember, type NEW before each new program, and use line numbers for each instruction. It's also good policy to LIST your program before you RUN it, just to make sure that all is as you intend.

Now suppose we wanted to alter Program II so that it printed out the message:

PROGRAMMING
IS
RATHER
SIMPLE

We need a line in there between 20 and 30 to print out "RATHER". Well, 25 is a number between 20 and 30, so let's try:

25 PRINT "RATHER" [Enter]

If you list it you'll see that the program has now become Program III:

```
10 PRINT"PROGRAMMING"
20 PRINT "IS"
25 PRINT "RATHER"
30 PRINT "SIMPLE"
```

Program III

So line 25 has "crept in" between 20 and 30. Even though we entered it out of order, the Amstrad stores it in memory in its correct numerical position. Try running the program as final confirmation.

This ability to insert lines into programs is the reason our line numbers go up in steps of 10 when we are writing programs – it leaves us plenty of spare line numbers in between for when we are patching them up.

Now enter Program IV:

```
10 CLS
20 PRINT "COMPUTING"
30 PRINT "WITH THE"
40 PRINT "AMSTRAD"
```

Program IV

remembering to press Enter after typing each line.

Now LIST it. Is there a phantom line 25 in there?

If so, you didn't type NEW after the last program – the lines 10, 20 and 30 of the latest program have replaced those lines in the old program. But as the new program doesn't have a line 25, the old one remains to ruin your program.

The moral is to use NEW before entering a new program.

If you have got an unwanted line 25, don't worry – you can easily get rid of it by typing:

25 [Enter]

This will delete the line since you replace the old line 25 with a new line which contains nothing – which the micro then "forgets".

This method holds good for

deleting any line from a program – simply type out the line number, then press Enter.

Program IV contains the keyword CLS. And this, as you shall see when you run the program, clears the screen.

Now let's try to print out our message with blank lines between. We can use a line containing just PRINT to obtain a blank line, so Program V should do the trick.

```
10 CLS
15 PRINT
20 PRINT "COMPUTING"
25 PRINT
30 PRINT "WITH THE"
35 PRINT
40 PRINT "AMSTRAD"
45 PRINT
```

Program V

Now try Program VI:

```
10 CLS
20 PRINT "HELLO";
30 PRINT "OUT";
40 PRINT "THERE"
```

Program VI

The output you will get is:

HELLOOUTTHERE

that is, each successive string is printed after the preceding one. The semicolon stops the next string being printed on a new line, "gluing" it to the end of the previous string printed.

Notice that, since there are no spaces inside the strings, none appear between the words when they are printed out together.

Try to get the message to appear legibly by rewriting the program with appropriate spaces in the strings. Also notice that you can obtain the same output, far more simply, with Program VII:

```
10 CLS
20 PRINT "HELLO OUT THERE"
```

Program VII

However, as I said above, the programs I present to you are for making teaching points, which does not necessarily imply showing you the most efficient methods.

Experiment with joining up the output of successive PRINT statements with the use of the semicolon until you feel confident about it.

And now for something completely different. Try running Program VIII:

```
10 PRINT "I"
20 PRINT "FEEL"
30 PRINT "DIZZY"
40 GOTO 10
```

Program VIII

I think the effect is pretty impressive.

So far all our programs have merely copied back onto the screen what you have typed in. This program shows how, with the addition of one line (line 40), you can obtain a huge increase in the amount of output.

It is this ability, to repeat a simple operation rapidly, that gives the Amstrad much of its power.

If things are happening a little too fast for you, you can temporarily halt proceedings by pressing Esc once. If you then press the Space Bar or any other key things will continue at their normal rate.

If on the other hand, you press Esc once more, the program will stop running completely, and the message "Break" will appear on the screen.

What is happening is that the micro follows lines 10, 20 and 30 and prints out:

I *(line 10)*
FEEL *(line 20)*
DIZZY *(line 30)*

followed by three blank lines due to the apostrophes. It then encounters line 40, which tells it to go back to line 10. It duly does so and prints out:

I *(line 10)*
FEEL *(line 20)*

and so on until it reaches line 40, when it goes back to line 10 and so on ad infinitum. Notice that when the screen is full, it scrolls up to make more room.

Now the name for such a condition in a program, where you keep on repeating lines of code (as the program lines are known), is a loop.

We say here that we are in an unconditional loop because we haven't given the program any conditions for it to cease repeating itself.

Compulsively introspective micros are not useful machines!

This is bad programming practice – compulsively introspective micros are not useful machines!

To stop such unconditional loops, you have to interrupt them from "outside" by either pressing Esc (Escape), or pressing down (and keeping down) Ctrl followed by Shift followed by Esc. The latter causes what's known as a reset.

Of the two, Esc is to be preferred – you can compare it to stopping a car with the brakes. Using a reset is more akin to stopping your car by driving into a wall.

It won't actually damage your micro but it will cause you to return to the state of the micro on switch on, which is Mode 1, and with no program in when you LIST.

Esc on the other hand actually lets you continue if you wish, by simply pressing any key. The program will then continue. If not, press Esc again. The program will stop, but it's still there. Always use Esc if you can.

If you want to have some fun with an unconditional loop, try printing out repeatedly an arrow composed of asterisks such as:

```
*
***
*****
*******
***
***
***
***
```

which will scroll upwards off the screen.

Finally, apart from its being an unconditional loop, which is always naughty, can you see what else is going wrong with this program?

```
10 CLS
20 PRINT "THIS IS"
30 PRINT "VERY SILLY"
40 GOTO 10
```

Program IX

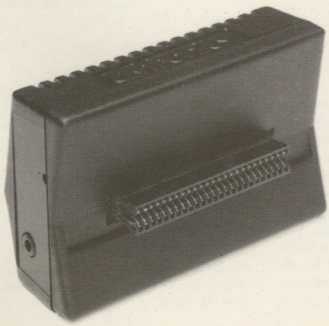
● *NEXT MONTH we'll use variables to give our programs even more power.*

Amstrad Speech Synthesizer!

The dk'tronics Amstrad speech synthesizer and powerful stereo amplifier uses the popular SLO/256 speech chip and has an almost infinite vocabulary. It is supplied with a text to speech converter for ease of speech output creation. Everything you wish to be spoken is entered in normal English, without special control codes or characters, it is therefore extremely easy to use. The voicing of the words is completely user transparent and the computer can carry on its normal running of a program while the speech chip is talking. The speech output from SLO/256 is mono and directed to both speakers.

To utilise the Amstrad stereo output on the back of the computer, the interface has a built in stereo amplifier, this gives all sound output a totally new dimension and greatly improves the sound quality and volume over the computer's internal speaker.

Although there are only 26 letters in the alphabet, letters have a totally different sound when used in different words. For example, The "a" in Hay is much longer and softer that in Hat. When you speak you automatically make adjustments because you know just how a word should sound. Not quite so easy with a computer. After looking at other speech synthesizers we decided that it was essential that the dk'tronics Amstrad Speech would offer a simple system that would enable the user to produce realistic speech that was instantly recognisable.



The solution to the problem was extremely complicated, it required hours of programming to enable the computer to look at the individual letters that make up each word and compare their relative position to each other before deciding on the appropriate sound.

I am delighted that we have now perfected what I consider to be the best Speech Synthesizer on the market, one which has achieved my aim, within the limitations of the allophones, of producing realistic speech.

At **only £39.95** the dk'tronics Amstrad Speech Synthesizer represents remarkable value for money.

Available from department stores and good computer shops everywhere or direct from

dk'tronics, Saffron Walden, Essex CB11 3AQ. 2CA Tel. (0799) 26350. Add £1.25 post and packing.

MICHAEL NOELS shows how easy – and how useful – it is to change the ink in your Amstrad's pen

Ink, ink everywhere nor any drop to spill

LAST month we investigated the use of colour on the Amstrad and saw that there were different ways, or modes, of using its screen.

Mode 0 gave us a screen of 25 lines, each line being 20 rather chunky characters across. The advantage of this was that we had the ability to display 16 different colours on the screen at once.

Mode 1 gave us rather more normal characters. There were still 25 lines on the screen, but this time they held 40 characters each. However, we paid for the increase in characters with a decrease in the number of colours displayed at once – only four at a time in this mode.

Mode 2 gave us rather thin characters, allowing 80 on each of our 25 lines. Again, there's a decrease in the number of colours available – we can only have two on the screen at the same time.

We also saw that to change the colour we write in, we use the command *pen*.

For example, if we had just switched on,

```
pen 2
```

would change the colour we're writing in to cyan, whereas:

```
pen 3
```

would make it red.

```
pen 1
```

would then restore our normal yellow writing and:

```
pen 0
```

would give us blue writing – not a good idea on a blue background.

We consider each pen to be filled with an ink, so *pen 2* has cyan ink in it and so on. That is, *pen 2* is *not* cyan – the ink in it is.

This idea of a pen being filled with an ink is very useful. Later on we'll see how to change the ink in *pen 2* to colours other than cyan.

To change the colour of the background we use the *paper* command. For example:

```
paper 3
```

will give us our characters on a red background.

You see, *paper 3* means "make the background (or paper) we write on the same colour as the ink presently in 3".

Lines scrolling onto the screen will henceforth be in the new background colour completely, and if you clear the screen with *cls* it will clear to that background.

We also saw last month that there was a border surrounding the screen – and we can change its colour.

For example:

```
border 0
```

will give us a black border.

We also saw that we could obtain all 27 colours available on the Amstrad by running a program to change the borders successively from *border 0* to *border 26*.

Notice that the border is independent of the mode we are in. As far as the screen's concerned, the mode we're in restricts the number of colours we can use. The border is free to adopt whichever of the 27 colours it chooses.

Take a closer look at:

```
border 0
```

Now before, if we did:

```
paper 0
```

the "0" meant "the colour of the ink in pen 0"

However, the "0" in:

```
border 0
```

doesn't have anything to do with *pen 0*. After all, it turns the border black, and so far, whichever mode you've been in, *pen 0* has been blue.

If *border 0* meant "make the border the same colour as the ink in pen 0" it would go blue, not black.

The "0" in *border 0* refers to ink number 0. Each of the Amstrad's 27 inks, or colours, has a unique number associated with it. Table I shows the complete list.

This number is always fixed: 0 is always black, 26 always bright white.

However, as we've seen, only a selection of the 27 inks can be on screen at any one time, because of the limitations on colour each mode poses.

So far we've acted as if this selection were fixed. For example, in Mode 1:

pen 0 has bright blue ink.
pen 1 has bright yellow ink.
pen 2 has bright cyan ink.
pen 3 has bright red ink.

These, though, are only the initial selection the micro makes for you at switch on, the default colours as they are known. Table II shows these for Mode 1.

You can, however, fill your pens with whatever inks you wish. You're still limited as to the number of pens – four in Mode 1 – but you can pick whichever four inks you want to fill them with.

In fact you can fill them all with the same colour if you wish – and you might want to do that, as we'll see.

For the moment, though, let's change it so that the ink in pen 2 is bright green. As we saw last month, the micro likes to do things by numbers. The ink number of bright green is 18.

Translated into the micro's language:

"fill pen 2 with bright green"

becomes

```
ink 2,18
```

Notice:

ink This command is short for "fill a pen with a new ink".

2 This is the number of the pen we want filling.

18 This is the number of the ink we want putting in the pen.

Let's try it.

Reset your micro by pressing Esc while holding down Shift and Ctrl, then clear the screen with CLS. You'll be in Mode 1.

You should now be left with just the Ready prompt and the cursor. Enter:

```
ink 2,18
```

Nothing much seems to have happened. Let's see what happens when we go to pen 2, though. Enter:

```
pen 2
```

and you'll see the effect immediately, as the Ready prompt will now appear in bright green, as will any other letters you type. Pen 2 really has been filled with a different ink.

Try typing a few words on the screen, so that you have at least a couple of green lines on the screen.

Previously *pen 2* gave us bright cyan – that is, it was filled with ink number 20. Go back to the original state by entering:

```
ink 2,20
```

and look carefully at the screen as you press Enter.

Everything that was written in ink 2 now appears in cyan. It doesn't matter if it was written before the ink in pen 2 was altered, it still changes. Ink is a retrospective command.

If you now try:

```
ink 2,18
```

you'll see that everything written with pen 2 immediately becomes bright green once more.

The reason is that you are limited to a fixed number of colours on the screen at once. If the new micro didn't change all of the writing in pen 2 immediately, you could cheat by choosing pen 2 then using:

ink 2,0 and writing . . .
then *ink 2,1* and writing . . .
then *ink 2,2* and writing . . .

and so on, until you'd assigned all 27 inks to pen 2 and written in each of them, breaking our rules.

The micro foils this by immediately changing all the writing in pen 2, old and new, to the colour specified by the ink command.

What actually happens is that the whole of your micro's screen is redrawn (or, "refreshed") 50 times a second, as is a normal television — that's how you get moving pictures.

Each time it starts to draw the screen, the micro asks itself, "What ink is in pen 2?" and then it draws everything it remembers as being written with pen 2 in that colour ink.

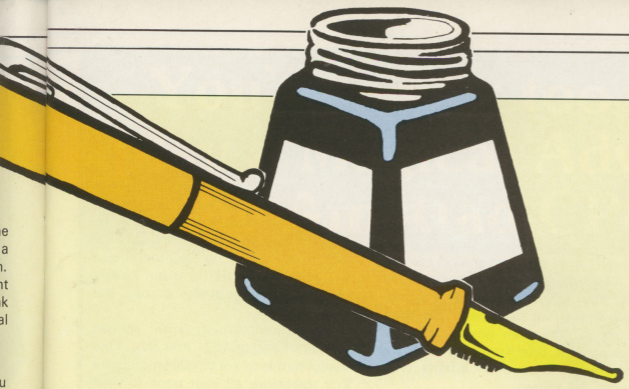
Of course, if you've just changed the ink in pen 2, it then redraws the screen using that new ink for pen 2 and – hey presto! – everything written

Ink number	Colour
0	Black
1	Blue
2	Bright blue
3	Red
4	Magenta
5	Mauve
6	Bright red
7	Purple
8	Bright magenta
9	Green
10	Cyan
11	Sky blue
12	Yellow
13	White
14	Pastel blue
15	Orange
16	Pink
17	Pastel magenta
18	Bright green
19	Sea green
20	Bright cyan
21	Lime green
22	Pastel green
23	Pastel cyan
24	Bright yellow
25	Pastel yellow
26	Bright white

Table I

Pen number	Ink number	Colour
0	1	Bright blue
1	24	Bright yellow
2	20	Bright cyan
3	6	Bright red

Table II: Default colours in Mode 1



in pen 2 changes colour.

Of course, this works for other pens as well – I've just used pen 2 as an example.

If you've been following this article at your micro – as you should be – you'll now be writing in bright green ink – since we're using pen 2 – on a blue background.

Change to pen 3 with:

```
pen 3
```

to get a bright red writing, or foreground colour. This hasn't altered since we haven't "refilled" the ink in pen 3.

Now enter:

```
paper 2 : cls
```

As you'll see, the screen clears to a bright green background with red writing in the foreground. This shouldn't be surprising – after all, *paper 2* means "change the background colour to the colour of the ink in pen 2", and pen 2 currently has bright green ink.

Let's sum up:

```
ink a,b
```

means refill *pen a* with *ink number b.*

Try:

```
ink 3,7
```

and, if you've been following so far, your writing should now be in purple. Can you see why?

So the state of play at the moment is shown in Table III and we're writing in pen 3 on paper 2.

Now let's see what happens to this set up, or configuration, if we change mode. Enter:

```
mode 0
```

You should still be writing in pen 3, which gives us purple letters, on a green background (paper 2).

However, as you'll have noticed, the rest of the screen is blue. When you change mode, the screen automatically clears to paper 0 and we haven't reassigned pen 0, which is blue. If you'd assigned pen 0 to pink with *ink 0,16,* the rest of the screen would be pink.

Generally, the Amstrad tries to preserve the status quo as far as possible over a mode change. For instance, you are still writing with pen 3 on paper 2 as before the mode change.

Also, pen 3 is still filled with ink 7 and so on.

Sometimes the Amstrad won't be so generous. For example, if you now go to Mode 2, since it only allows two pens, pens 2 and 3 won't work. However, if you've reassigned pen 0 and 1 with *ink,* these pens will appear in their new guise when you change to Mode 2.

Enough of this theory, let's try some programs to illustrate these ideas. Program I simply illustrates the use of INK, assigning all the different inks to pen 1 consecutively.

Line 40 sets up pen 1. We then use a FOR . . . NEXT loop to step us through all 27 colours (lines 50 to 100). Line 60 actually does the assigning, while lines 70,80 print out a simple message. Line 90 waits for a key to be pressed before continuing.

Program II is more complex. We're still assigning new inks but this time we're using pens 1,2 and 3, cycling the different inks through them.

Lines 50-80 form a FOR . . . NEXT loop that assigns a different, but consecutive ink number to each pen in line 60 (*biro* holds the pen number and *colour* is an increasing offset). Lines 70,80 pick pen *biro* and print out a simple message.

Line 100 waits for a key to be pressed, then line 110 increases *colour.* The FOR . . . NEXT loop is itself wrapped up in an infinite WHILE . . . WEND, lines 40, 120, which causes the loop to be repeated indefinitely, though each time with an increased value of *colour.*

This in turn causes the colours to cycle through the pens as the ink number in line 60 will automatically be increased each time round.

Program III, in Mode 0, shows how we can use INK to perform a simple animation. The FOR . . . NEXT loop of 30-80 simply prints seven little men horizontally across the screen. Lines 60 and 70 locate and print the man, CHR$(249).

However, each man is drawn in the same colour as the background so you don't see him. More than that,

Pen number	Ink number	Colour
0	1	Bright blue
1	24	Bright yellow
2	18	Bright green
3	7	purple

Table III

```
10 REM PROGRAM I
20 MODE 1
30 LOCATE 8,12
40 PEN 1
50 FOR colour = 0 TO 27
60 INK 1,colour
70 LOCATE 10,12
80 PRINT "This is in pen 1"
90 WHILE INKEY$="": WEND
100 NEXT colour
```

Program I

```
10 REM PROGRAM II
20 MODE 1
30 colour = 0
40 WHILE -1
50 FOR biro = 1 TO 3
60 INK biro,(colour + biro) MOD 27
70 PEN biro
80 PRINT "This is in pen ";biro
90 NEXT biro
100 WHILE INKEY$="":WEND
110 colour = colour + 1
120 WEND
```

Program II

each man is drawn with a different pen.

To do this, line 40 sets the ink of the current pen (held in variable *biro*) to the background ink, 1, that is, bright blue. Line 50 ensures we write the little man in the current pen.

```
10 REM PROGRAM III
20 MODE 0
30 FOR biro = 1 TO 7
40 INK biro,1
50 PEN biro
60 LOCATE (3*biro-2),12
70 PRINT CHR$(249)
80 NEXT biro
90 WHILE -1
100 FOR man = 1 TO 7
110 INK man,24
120 FOR delay = 1 TO 500: NEXT
130 INK man,1
140 NEXT
150 WEND
```

Program III

After line 80, what we've got is seven little men, all in a line across the screen, all in the background colour.

Now what we're going to do is turn one man bright yellow for a while, then make him disappear. We then "turn on" the man next to him by making him bright yellow, then he disappears, and so on all the way across the screen.

The effect appears as one yellow man running all the way across the screen.

This is accomplished in FOR . . . NEXT loop 100, 140. *man* holds the number of the relevant man (and incidentally that of the pen used to draw him). Line 110 switches him on by filling the pen he was drawn with with yellow ink.

Line 120 causes us to wait for a while, then we send him back to oblivion with line 130, which switches his pen colour to the background colour, so he disappears.

The loop deals this way with all seven men, and, as it's wrapped in an infinite WHILE . . . WEND loop (lines 90,150), our little man keeps repeating his journey across the screen.

Try experimenting with various combinations of PEN and INK. One useful hint is to set up the small Enter key to restore all your pens to their default inks when you get lost or when your writing disappears against the background.

You can do this by entering:

```
KEY 139, "CALL &BC02 : PEN 1 " +
CHR$(13)
```

Now all you have to do to get back to normal, is to press Enter on the numeric pad. We call it the panic button here.

See you next month – we'll be looking at lines . . .

You're never too young to play a Magical Adventure on the Amstrad CPC464 . . .

Makes an ideal present

Based on the style of the classic computer adventures – but written so that even small children can learn to find their way around, encouraged by colourful graphics and exciting sound effects.

The pack contains a 48-page full colour storybook
PLUS
a full length multi-location adventure on cassette for only

£8.95! post free

Read the book – then play the game!

Please send me the complete Magic Sword pack for the Amstrad CPC464 containing storybook and cassette to:

Name _____

Address _____

_____

_____

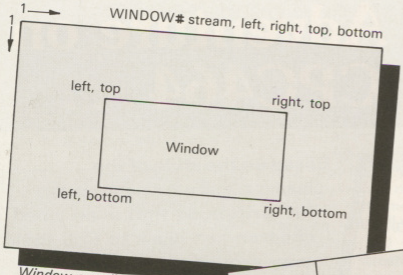
☐ I enclose my cheque for £8.95 payable to Database Publications
☐ Or debit my Access/Visa card:

No. _____

Signed _____

SEND TO: **Adventure offer, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY**

CA2



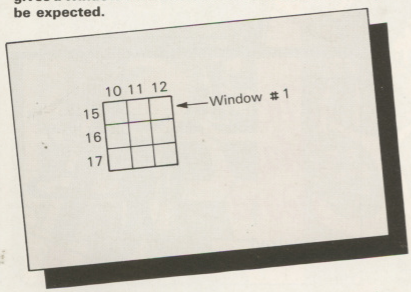
Window coordinates

Get the facts at your fingertips with the second of our ready reference charts

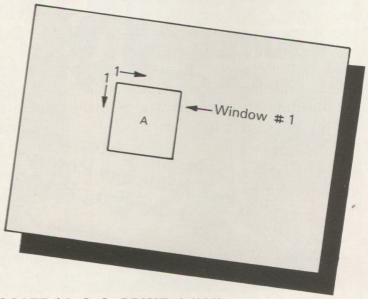
Mode	stream	left	right	top	bottom
0	0 - 7	1→20	1→20	1-25	1-25
1	0 - 7	1→40	1→40	1-25	1-25
2	0 - 7	1→80	1→80	1-25	1-25

Window parameter ranges

Beware the following trap: WINDOW#1, 10, 12, 15, 17 gives a window 3 x 3 characters wide not 2 x 2 as might be expected.



The LOCATE command works for each window in exactly the same way as it does for the whole screen measuring from the top left corner.



Each window is referred to by its stream number.

INPUT
CLS
PRINT
LOCATE
PAPER
PEN
TAG

All these can use # stream to specify a window.

LOCATE #1, 2, 2: PRINT#1 "A"

Each window acts as a miniature screen, working independently. They all scroll text upwards when the window is full, the top line disappearing.

Window#0 is SPECIAL:
1. It's the default window. If you don't put in a stream number or try to write to a window you haven't defined, it assumes window # 0.
2. It fills the whole screen (unless you change this).
3. The system messages all use window # 0.

WINDOW SWAP stream, stream
This exchanges the specified text windows. For example:
WINDOW #3, 5, 7, 9, 12: WINDOW SWAP 0, 3 means that all system messages will appear in the smaller window.

"An essential companion to the CPC-464 complete with Amsoft approval"

The **Amstrad CPC-464 Advanced User Guide** by top-selling author Mark Harrison has been produced with help from Amsoft, the computer products division of Amstrad. As a result of this liaison the book conforms to Amsoft's presentation conventions and accordingly carries their 'seal of approval'. With such fine pedigree the Advanced User Guide is a must for every 464 user.

BASIC advice

Clearly and concisely organised throughout, the book opens with a description of how the 464 works, moving on to communication with external devices and a summary of BASIC. A comprehensive reference section is included enabling you to find an explanation of any BASIC command or keyword in the 464's repertoire.

Programming techniques

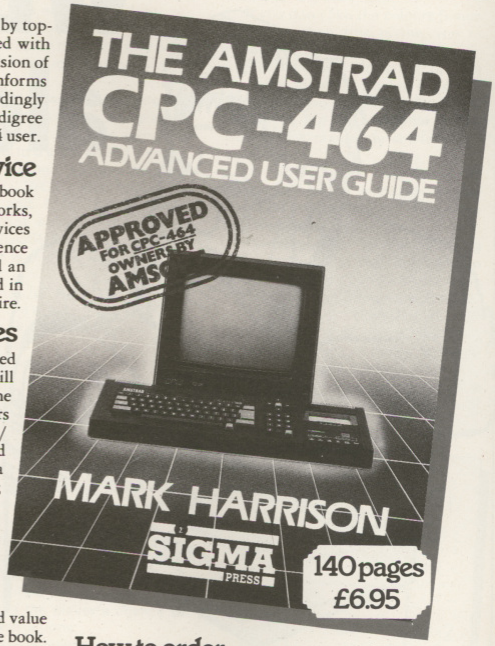
After getting to grips with BASIC you can proceed to the various programming techniques that will help you to get the most from your 464. The Advanced User Guide contains detailed chapters on Strings and Character Manipulation; Input/Output Techniques; Arithmetic; the Amstrad Memory Map; Time, Clocks and Interrupts; Data Structures; Data Processing; Amstrad Graphics; Sound and Synthesis.

Ready-to-run programs

You'll find no fewer than FORTY ready-to-run programs in the Advanced User Guide that demonstrate how the 464 works and illustrate useful programming techniques. The combined value of these alone far exceeds the cost of the book. Ranging in size they cover such topics as Code-breaking; Information Sorting; 3-D Graphics; Business Applications. There are also arcade-style space games and, most spectacular of all, a complete Sound Synthesiser program.

What Amsoft say

In the preface to the Advanced User Guide William Poel of Amsoft writes — " . . . we are particularly grateful to Mark Harrison for filling in most of the gaps that inevitably occurred in the original user handbook, as well as adding much to the general utility of the machine in the hands of the users . . . books such as this can do nothing but good for both the product and its users."
Amsoft clearly approve — we think that you will too.



How to order

The **Amstrad CPC-464 Advanced User Guide** costs only £6.95. You can obtain your copy through any good bookseller/computer store or by completing the order form and returning it to John Wiley & Sons Ltd.

To: Lesley Valentine, John Wiley & Sons Ltd, FREEPOST, Baffins Lane, CHICHESTER, West Sussex PO19 1YP

Please send me ..... copy(ies) of **The Amstrad CPC-464 Advanced User Guide** by Mark Harrison at £6.95 per copy
Postage and packing free — please allow 14 days for delivery
☐ I enclose a cheque /PO for £..... payable to John Wiley & Sons Ltd.
☐ Please debit my credit card account £.....
Card No. ..... Expiry date .....
Access ☐ American Express ☐ Barclaycard ☐ Visa ☐ Diners Club ☐
Telephone credit card orders - dial 100 and ask for FREEFONE 3477
NAME/ADDRESS .....

SIGNATURE .....
JOHN WILEY & SONS LTD REG. NO. 641192 ENGLAND

Σ SIGMA PRESS
MARKETED BY JOHN WILEY & SONS LTD, BAFFINS LANE, CHICHESTER, SUSSEX PO19 1UD, ENGLAND

WEND your
along the pa
programm

PETE BIBBY shares his first experiences in writing games for the Amstrad computer – and invites you to play with the result

ONE of the best ways to learn how a micro works *(or doesn't work)* is to have a go at writing a program for it. It's also a good way of finding out how little you know about it! So as soon as I'd got my Amstrad out of its box and onto the table in the spare bedroom I decided to write one.

But what to write? Having cut my computing teeth on the ZX81 and the BBC Micro I didn't know much about the PAPER and INK commands of the CPC464. I decided to write a program that would make use of them, and Simon was the result.

It uses the Amstrad's sound and graphics commands to produce an electronic version of the old children's game. Not only did it give me a lot of experience in writing Amstrad programs, I also ended up with a rather addictive game.

If you look at the listing, you'll see that the main structure of the program is in lines 30 to 110. As you can see, this *control module* as it's called, consists of five calls to subroutines set in two WHILE . . . WEND loops.

These are the key lines of the program, for while lines 130 to 1180 actually make up the subroutines it's lines 30 to 110 that decide when to call them. If they weren't there the program wouldn't work. Try leaving them out if you don't believe me!

If you look at the REM statements that comment on the subroutine calls, you'll get an idea of how I planned the program.

First of all I knew that I would have to have some instructions on the screen. So I decided to have a subroutine that printed them and also set up any initial variables that might be needed. Hence the subroutine starting in line 140 was born.

However at this stage I didn't bother with the actual coding of the routine. I just stuck in a PRINT "INSTRUCTIONS" and a RETURN and left it at that.

I could figure out the precise details of the instructions later. For the time being I was content with the simple message being printed every time the subroutine was called.

I'd decided that the game itself would consist of tunes made up of selection of five notes. Each note would also have a coloured rectangle linked with it. When a tune played you would hear the notes and see the flashing colours. The player would then try to reproduce these.

To do this I saw that I would have to set up the screen to display the colours, get the micro to decide on a tune, and then have it play the tune, flashing the necessary colours.

Next the player would have to enter his attempt at recreating the tune and the Amstrad would have to

r way path to mming

check whether it was correct and take the appropriate action.

If the player got it right then another tune, one note longer, should be produced. If the player is wrong the program should announce it and let the player have another go.

The next four subroutines do all this. The first (GOSUB 250) sets up the screen windows as needed while the second (GOSUB 610) decides on the sequence of notes to be played.

The following routine (GOSUB 700) plays the note sequence and the final call (GOSUB 880) takes in the answer from the keyboard, checks it and takes appropriate action.

Again I didn't bother figuring out what would go in these subroutines. For the time being all I was interested in was whether the logic of the program worked, so I just had the subroutines printing out messages to show me how the program was working.

The two WHILE . . . WEND loops are used to call the subroutines as necessary. The inner loop formed by lines 60 and 100 just repeats the enclosed subroutines while the variable *correct* is equal to −1.

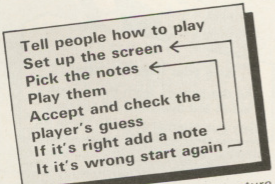


Figure I: Control module structure

These subroutines are the ones that actually play the game and while *correct* is equal to −1 they carry on going.

However the last subroutine (GOSUB 880) checks the player's answer and if it is wrong *correct* is made equal to 0 and the loop finishes.

Then the program enters the outer WHILE . . . WEND loop formed by lines 40 and 110. This calls the routine that sets up the screen windows (GOSUB 250) and then the inner loop is entered and the game carries on.

And that is how the control module of the program works. Figure I shows it all diagramatically.

Once I had this "skeleton" of the program up and working all I had to do was write each of the subroutines. By dividing the major task of writing the program into lots of little sub-tasks things were made much easier.

Let's take a look at the coding that makes up these subroutines.

As you can see from the REM, the first is the one that prints the instructions on the screen, using the LOCATE command to position them. The INKEY$ of line 210 just holds things up until a key is pressed.

This gives the player a chance to read the instructions.

Once a key is pressed the program continues to line 220 and gives *repeat* a value of −1. Since the value of *repeat* is never altered this means that the outer loop keeps on forever.

After setting *repeat* the Amstrad hits the RETURN of line 230 and the subroutine is ended. The program goes to the line following the one that called the subroutine. In this case it's line 40 and the program enters the eternal outer loop.

Immediately it comes to line 50, which sends it off to the subroutine that starts at line 250. As the REM statement tells us, this set of code sets up the screen windows.

First of all, however, it puts the micro into Mode 0, sets *correct* to minus 1 and the variable *tunelength* to 3.

Then line 290 assigns to the paper/pen numbers 10, 11, 12, 13 and 14 the ink numbers that produce the colours bright yellow, bright red, bright cyan, bright green and bright magenta respectively.

If you don't like the colours Simon produces then this is the line to change.

The next five lines set up five screen windows, one below the other. The next two lines set the paper colour of the first window to 10 (which was previously set to bright yellow).

It then clears this window, producing a bright yellow square and the number that refers to that window is printed next to it.

This is repeated for the remaining four windows and then lines 460 to 480 display the "carry on" message by the side of the coloured rectangles.

The WHILE . . . WEND of line 490 gets rid of any spurious input that might have built up while line 500 loops round until a key is pressed.

When this happens the screen is cleared. Line 520 sets the paper of

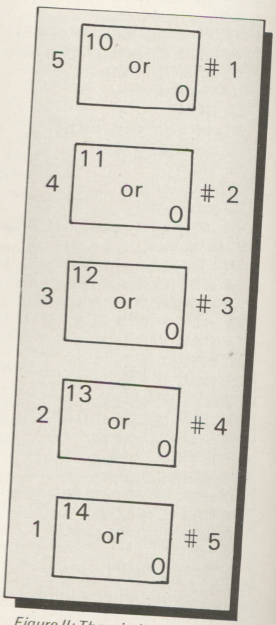


Figure II: The windows, streams and paper numbers

```
10 REM SIMON SAYS
20 REM PETE BIBBY
30 GOSUB 140:REM INSTRUCTIONS AND INI
TIALISATION
40 WHILE repeat
50 GOSUB 250:REM SET UP SCREEN WINDOW
S
60 WHILE correct=-1
70 GOSUB 610:REM SET UP NOTE SEQUENCE

80 GOSUB 700:REM PLAY NOTE SEQUENCE
90 GOSUB 880:REM ACCEPT AND CHECK ANS
WER
100 WEND
110 WEND
120 REM**********************
130 REM INSTRUCTIONS
140 MODE 1
150 LOCATE 15,4 :PRINT"INSTRUCTIONS"
160 LOCATE 15,5 :PRINT"************"
170 LOCATE 11,7:PRINT"USE THE NUMBER
KEYS"
180 LOCATE 11,9:PRINT"AND TRY TO RECR
EATE"
190 LOCATE 17,11:PRINT"THE TUNE."
200 LOCATE 15,23:PRINT"PRESS A KEY"
210 IF INKEY$="" GOTO 210
220 repeat=-1
230 RETURN
240 REM**********************
250 REM SCREEN WINDOWS
260 MODE 0
270 correct=-1
280 tunelength=3
290 INK 10,24:INK 11,6:INK 12,20:INK
13,18:INK 14,8
300 WINDOW #1,15,18,2,5
310 WINDOW #2,15,18,7,10
320 WINDOW #3,15,18,12,15
330 WINDOW #4,15,18,17,20
340 WINDOW #5,15,18,22,25
350 PAPER #1,10:CLS #1
360 LOCATE 10,4:PEN 10:PRINT 5
370 PAPER #2,11:CLS #2
380 LOCATE 10,9:PEN 11:PRINT 4
390 PAPER #3,12:CLS #3
400 LOCATE 10,14:PEN 12:PRINT 3
410 PAPER #4,13:CLS #4
420 LOCATE 10,19:PEN 13:PRINT 2
430 PAPER #5,14:CLS #5
440 LOCATE 10,24:PEN 14:PRINT 1
450 PEN 1
460 LOCATE 1,9:PRINT "PRESS A"
470 LOCATE 1,11:PRINT "KEY TO"
480 LOCATE 1,13:PRINT "CONTINUE"
490 WHILE INKEY$()"":WEND
500 IF INKEY$="" GOTO 500
510 CLS
520 PAPER #1,0:CLS#1
530 PAPER #2,0:CLS#2
540 PAPER #3,0:CLS#3
550 PAPER #4,0:CLS#4
560 PAPER #5,0:CLS#5
570 PEN 1
580 RETURN
590 REM**********************
600 REM SETS UP NOTES
610 sequence$=""
620 FOR loop=1 TO tunelength
630 trial=INT(RND*5+1)
640 trial$=STR$(trial)
650 trial$=MID$(trial$,2,1)
660 sequence$=sequence$+trial$
670 NEXT loop
680 RETURN
690 REM**********************
700 REM PLAYS NOTES
710 IF correct=-1 THEN LOCATE 2,11:PR
INT"MY"
720 IF correct=-1 THEN LOCATE 1,13:PR
INT"TURN"
730 IF correct=0 THEN LOCATE 2,11:PRI
NT"IT'S"
740 FOR loop=1 TO LEN(sequence$)
750 token$=MID$(sequence$,loop,1)
760 stream=VAL(token$)
770 IF stream=5 THEN SOUND 1,319,50,7
:PAPER #1,10:CLS #1:GOTO 820
780 IF stream=4 THEN SOUND 1,358,50,7
:PAPER #2,11:CLS #2:GOTO 820
790 IF stream=3 THEN SOUND 1,379,50,7
:PAPER #3,12:CLS #3:GOTO 820
800 IF stream=2 THEN SOUND 1,426,50,7
:PAPER #4,13:CLS #4:GOTO 820
810 IF stream=1 THEN SOUND 1,478,50,7
:PAPER #5,14:CLS #5:GOTO 820
820 IF SQ(1)=127 OR SQ(1)>127 GOTO 82
0
830 PAPER #(6-stream),0:CLS#(6-stream
)
840 NEXT
850 CLS
860 RETURN
870 REM**********************
880 REM YOUR TURN
890 LOCATE 1,11:PRINT"YOUR"
900 LOCATE 1,13:PRINT"TURN"
910 WHILE INKEY$()"":WEND
920 FOR loop=1 TO LEN(sequence$)
930 token$=INKEY$:IF INSTR("12345",to
ken$)=0 OR token$="" GOTO 930
940 stream=VAL(token$)
950 IF stream=5 THEN SOUND 1,319,50,7
:PAPER #1,10:CLS #1:GOTO 1000
960 IF stream=4 THEN SOUND 1,358,50,7
::PAPER #2,11:CLS #2:GOTO 1000
970 IF stream=3 THEN SOUND 1,379,50,7
:PAPER #3,12:CLS #3:GOTO 1000
980 IF stream=2 THEN SOUND 1,426,50,7
:PAPER #4,13:CLS #4:GOTO 1000
990 IF stream=1 THEN SOUND 1,478,50,7
:PAPER #5,14:CLS #5:GOTO 1000
1000 IF SQ(1)=127 OR SQ(1)>127 GOTO 1
000
1010 IF token$()MID$(sequence$,loop,1
) THEN GOSUB 1100 :RETURN
1020 PAPER #(6-stream),0:CLS#(6-strea
m)
1030 NEXT
1040 tunelength=tunelength+1
1050 CLS
1060 FOR delay= 1 TO 1000:NEXT delay

1070 RETURN
1080 REM**********************
1090 REM CORRECTION
1100 CLS
1110 SOUND 1,600,200,7
1120 FOR loop=1 TO 50:PRINT "WRONG! "
;:NEXT
1130 IF SQ(1)=127 OR SQ(1)>127 GOTO 1
130
1140 CLS
1150 correct=0
1160 GOSUB 700
1170 RETURN
1180 REM**********************
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

the first window to 0 and clears it with the result that the window now blends in with the rest of the screen.

The next four lines do this for each of the windows in turn.

Now that the windows have been set up and turned to the background colour, the program reaches the RETURN of line 580 and control returns to line 60.

It now enters the inner WHILE . . . WEND loop and hits line 70 which sends it off to the subroutine that starts at line 610.

This uses a FOR . . . NEXT loop and the RND function to produce numbers between one and five. These are stored in the string variable *sequence$* and are used to produce the sequence of notes and flashes that the player has to reproduce.

Once the Amstrad has picked the sequence of notes it wants played and has then stored the result in *sequence$*, the program meets the RETURN of line 680 which sends it back to line 80.

This calls the subroutine which plays the notes chosen. Lines 710 and 720 just print a simple message if *correct* is equal to −1.

For the time being we'll assume that it is −. This means line 730 is ignored as the IF condition isn't fulfilled.

The program now comes to the FOR . . . NEXT loop between lines 740 and 840. This loop cycles for as many times as there are notes in *sequence$*. As the loop cycles lines 750 and 760 detach one number at a time from *sequence$* and put it in a variable *stream*.

The next five lines all work in the same way. First of all *stream* is tested and if the condition isn't met then the rest of the line is ignored. If the condition is met then the program executes the rest of the line.

First of all a note is produced and then the paper of the corresponding window is changed to its original colour. The CLS clears the appropriate window to the new background colour and then the final GOTO jumps over the other lines to line 820. Figure II shows the windows and their paper colours.

Line 820 just holds up the program until the note is over. Try leaving it out and see what happens. The next line just switches off the window by returning its paper colour to the background colour.

In case you find that a little bit complicated, just suppose that when the subroutine was called *sequence$* was "153". What the loop does is to cycle three times, since *sequence$* is three characters long.

The first time round, *stream* is equal to 1 so line 810 is obeyed and the appropriate note played and window switched on.

When the note is over the window returns to the old background colour and the loop cycles again. This time *stream* is equal to 5 so line 770 is obeyed.

I leave it to you to figure out which line is obeyed the third time round the loop.

When the loop has finished cycling line 850 clears the screen and 860 ends the routine.

The final subroutine in the control module is now called by line 90. This routine accepts and checks the answer typed in by the player.

The first two lines tell the player that it's their go. Line 910 is used to get rid of any spurious input that may affect the program.

The FOR . . . NEXT loop between lines 920 and 1030 is reminiscent of the loop in the previous subroutine.

Each time round the loop the INKEY$ of line 930 accepts a number from the keyboard. The "mug trapping" in the line makes sure that only a number in the range 1 to 5 is accepted. Lines 950 to 990 work in exactly the same way as lines 770 to 810 and activates the requisite note and window.

Line 1010 checks that the player has got it right by comparing the number typed in with the number held in *sequence$*. If it's wrong then another subroutine is called to handle the matter.

For the moment let's ignore this negative case and assume that the player has succesfully recreated *sequence$*. When this happens, the program leaves the loop and goes onto line 1040.

This increases *tunelength* by one, ensuring that the next tune the CPC generates will be one note longer. The next line clears the screen while line 1060 just delays things for a while.

The RETURN in the next line sends the program back to line 100. This, in turn, sends the program back round the inner loop again, playing a tune and checking the input.

But what if the player made an error and the input was wrong? In this case the conditions of line 1010 would be fulfilled and the program goes to the correction subroutine at line 1100. This tells you in no uncertain terms that you've made a mistake and sets the variable *correct* to 0.

The next line calls the subroutine at line 700 which plays the correct sequence of notes. You'll notice that having *correct* equal to 0 means that a different message is printed on the screen.

When the correct tune has been played the subroutine finishes and the program goes back to 1170. This in turn ends the correction subroutine and the program comes to the WEND at the end of the inner loop.

Now, however, *correct* is 0, so the program doesn't cycle round the inner loop but goes onto line 110. This is the WEND of the outer loop, so now the program returns to line 50 and the whole sequence is repeated.

And that's how Simon works. It's hardly the world's most brilliant bit of programming but it is fun.

And if you've followed the above you should be well on the way to writing your own games.

Venture deep into the labyrinth of despair, beat the wizard and collect your reward in . . .

The Kingdom of Craal

FAR far away, beyond the Ice Mountains, lies the weird and wonderful Kingdom of Craal with its magnificent palace, crystal clear lake and enchanted forest.

It is a peaceful land, ruled for many happy years by King Meek who was respected by most of his subjects for his good nature and integrity – and his rather dishy, if inefficient, handmaiden Jajut.

There was one character however who was not party to this overwhelming admiration – Vadham the evil wizard.

Many years ago, he was banished to the castle dungeons for trying to nick the king's gold plated penknife – the one with a special blade for taking stones out of horses hooves.

In his spare time when he was not mixing spells or playing Ghouls, he turned the dungeon into a labyrinth of despair, where only the brave had the courage to enter. Having said that, we haven't seen any of them come back yet.

You were born in Craal the handsome son of a cobbler and many years ago you decided to seek a fortune in far off lands.

You returned many times over the years to visit your old mum and dad and spend the occasional happy hour in the company of the king's daughter Andrea.

What was once a childhood crush on his heir blossomed into love, and you have returned for good to claim your bride.

Alas on this last visit you found the king dead and the palace in uproar. The wizard had taken a heaven sent opportunity and, as the king opened the door down to the dungeons to let out the castle moggie, he grabbed his magnificent gold crown and disappeared into his hidey hole in the depths – better than a penknife no doubt, but not much use for getting things out of horses hooves.

You, in your typical youthful manner, were only interested in your future wife – Andrea. The palace guards, the footmen, the courtesans and even Ethel the cleaner, however, were not impressed with your infatuation and by a unanimous decision volunteered your services to retrieve the crown.

They threw you head-over-heels down the dungeon steps with a warning that should you return empty handed all your beloved possessions will be forfeit, even your subscription to *Computing with the Amstrad* – they'll stop at nothing some people.

By MIKE BIBBY

Well, you have your challenge and you don't really have much option but to accept it.

In this serious adventure you have at your disposal six single letter commands. These are n, s, e, w, l and i – for the four compass directions, look and inventory respectively.

The program will also accept other standard adventure commands such as take, drop, hit and say. These words are intelligent, which means that if you have a key and want it in a lock, all you need to say is 'drop key'. It will automatically go in the lock.

Now there's not much point in your typing in an adventure and finding, as you do, all the solutions within the listing. In order to conceal the clues therefore, I've written the important messages in code and they're all in the data statements at the end of the program.

There's nothing clever in what I've done, and I'm sure you'll soon spot that all the printed text has been offset by three letters. The subroutine starting at line 560 decodes it all and turns it into sensible English in the finished product.

It is imperative that great care is taken when entering these data lines if you are to enjoy the result of your toils.

Well, I think I've told you enough now. Any more hints and it wouldn't be much of an adventure, would it?

It only remains for me to wish you luck when you set out in your search for the crown – you're going to need it.

```
10 REM By Mike Bibby
20 REM (c) Computing With The Amstrad
30 MODE 1
40 CALL &BC02
50 INK 0,15:INK 1,3:INK 2,0:INK 3,1
60 BORDER 15
70 GOTO 220
80 DIM d%(10,4)
90 h$=SPACE$(25):c$=h$:o$=h$:h$="":c$
="":o$=""
100 hh$=SPACE$(255):oo$=h$:hh$="":oo$
=""
110 FOR I%= 1 TO 10: FOR J% = 1 TO 4
120 READ d%(I%,J%)
130 NEXT :NEXT
140 MX=12:TX=7:MX=51
150 DIM j$(MX):DIM o%(MX):DIM a$(MX)
160 FOR I% = 1 TO MX:READ hh$:READ cc
$:GOSUB 560:j$(I%)=oo$:hh$=cc$:GOSUB
560:o%(I%)=VAL(oo$):NEXT I%
170 FOR I% = 1 TO MX:READ a$(I%): NEX
T I%
180 h$="":c$="":o$=""
190 b%=-1:d%=-1:e%=-1:s%=0:l%=0:a%=-1
:g%=0:f%=-1
200 R%=2:x%=3
210 GOTO 270
220 PEN 1:PRINT:PRINT "On a visit to
the palace of Craal, you find the pl
ace in uproar. The king is dead and
his crown stolen by a wicked wizar
d who's fled to his den in the pa
lace dungeons."
230 PRINT "By paying rather too much
attention to the ex-king's daughter,
```

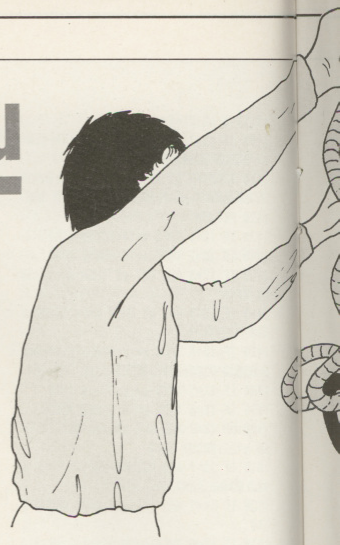
```
you find        yourself volunteered
 to recover it."
240 PRINT "You are thrown into the du
ngeons and    told not to come back w
ithout the crown."
250 PRINT "Here begins the adventure.
..."
260 GOTO 80
270 WHILE NOT g%
280 IF R% <> x% THEN GOSUB 640
290 x% = R%
300 C%=0: WHILE C%=0:GOSUB 410:WEND
310 ON C% GOSUB 840,880,910,960,1120,
640,1030,990,1060,1210
320 WEND
330 PRINT
340 IF o%(8)=1 THEN GOTO 360 ELSE a=3
1:GOSUB 2200
350 a=32:GOSUB 2200:PRINT:GOTO 380
360 a=29:GOSUB 2200:PRINT
370 a=30:GOSUB 2200:END
380 PRINT:a=51:GOSUB 2200
390 a$="":WHILE a$="":a$=INKEY$:WEND
400 a$=UPPER$(a$):IF a$="N" THEN END
ELSE RUN
410 PEN 2:PRINT:PRINT"What now?"
420 c$="":WHILE c$="":INPUT "===>",c$
: WEND :c$=LOWER$(c$)
430 IF LEN(c$)<> 1 GOTO 450
440 C%=INSTR("nsewil",c$): IF C%<>0 T
HEN RETURN ELSE PRINT"I don't recogni
se this single letter command - only
n,s,e,w,i,l.":RETURN
450 SX%=INSTR(c$," "): IF SX%=0 THEN
PRINT "I don't understand - put a spa
ce between command and object, please
.":C%=0:RETURN
460 verb$=LEFT$(c$,SX%-1):o$=" "+MID$
(c$,SX%+1):WHILE LEFT$(o$,1)=" ":o$=R
IGHT$(o$,LEN(o$)-1):WEND
470 C%=INSTR("droptakesayhit",verb$)
480 IF C%<>1 AND C%<>5 AND C%<>9 AND
C%<>12 THEN PRINT "I don't understand
 your command.":C%=0:RETURN
490 IF C%=1 THEN C%=7 ELSE IF C%=5 TH
EN C%=8 ELSE IF C%=12 THEN C%=10 ELSE
 IF C%=9 THEN RETURN
500 z%=0:I%=1:M%=0:WHILE M%<>1 AND I%
<>N%+1
510 IF LEFT$(o$,4)=LEFT$(j$(I%),4) TH
EN M%=1
520 I%=I%+1
530 WEND
540 IF M%=1 THEN z%=I%-1 ELSE PRINT"I
 don't understand the object you mean
.":C%=0:RETURN
550 RETURN
560 oo$=""
570 FOR JJ%= 1 TO LEN(hh$)
580 RR%=ASC(MID$(hh$,JJ%,1))-3
590 IF RR%=38 OR RR%=41 OR RR%=34 THE
N RR%=RR%+3
600 oo$=oo$+CHR$(RR%)
610 NEXT
620 RETURN
630 END
640 in=-1:a=R%-1:GOSUB 2140
650 PRINT:PRINT:PRINT"You can see aro
und you :-"
660 F%=0
670 FOR I%= 1 TO N%
680 IF o%(I%) <> R% THEN GOTO 700 ELS
E F%=-1
690 GOSUB 1430
700 NEXT I%
710 IF NOT F% THEN PRINT"Nothing of i
nterest."
720 PRINT
730 RETURN
740 DATA 0,0,0,0
750 DATA 3,9,0,0
760 DATA 0,2,4,0
770 DATA 0,0,5,3
780 DATA 0,0,0,4
790 DATA 0,7,0,0
800 DATA 6,0,0,8
810 DATA 0,0,7,9
820 DATA 2,0,8,0
830 DATA 0,0,0,0
840 IF d%(R%,1)=0 THEN PRINT" Not all
owed":PRINT:RETURN
850 IF R%=7 AND a% THEN a=33:GOSUB 22
00:PRINT:RETURN
860 R%=d%(R%,1)
870 RETURN
880 IF d%(R%,2)=0 THEN PRINT" Not all
owed":PRINT:RETURN
890 R%=d%(R%,2)
900 RETURN
910 IF d%(R%,3)=0 THEN PRINT" Not all
owed":PRINT:RETURN
920 IF d%(R%,3)=8 AND ff% THEN a=34:G
OSUB 2200:ff%=0
930 IF d%(R%,3)=7 AND o%(7)=0 THEN o%
(7)=8:a=35:GOSUB 2200:PRINT
940 R%=d%(R%,3)
950 RETURN
960 IF d%(R%,4)=0 THEN PRINT " Not al
lowed":PRINT:RETURN
970 R%=d%(R%,4)
980 RETURN
990 IF o%(z%)=1 THEN PRINT"You alread
y have it":RETURN
1000 IF o%(z%)<> R% THEN PRINT"It's n
ot here":RETURN
1010 IF z%(T% THEN PRINT"You can't ta
ke that" ELSE o%(z%)=1
1020 RETURN
1030 IF o%(z%)=1 THEN o%(z%)=R% ELSE
PRINT "You don't have it":RETURN
1040 ON (R%-1) GOSUB 1280,1300,1290,1
340,1290,1290,1360,1290,1290
1050 RETURN
1060 h$="":FOR I%= LEN(o$) TO 1 STEP
-1
1070 h$=h$+MID$(o$,I%,1)
1080 NEXT I%
1090 IF o$=h$ THEN g%=-1:a=50:GOSUB 2
200:RETURN
1100 PRINT"Okay,"+CHR$(34)+o$+CHR$(34
)
1110 RETURN
1120 PRINT"Your inventory contains:-"
1130 F%=0
1140 FOR I%=T% TO N%
1150 IF o%(I%)<> 1 THEN GOTO 1170 ELS
E F%=-1
1160 GOSUB 1430
1170 NEXT I%
1180 IF NOT F% THEN PRINT"Nothing at
all."
1190 PRINT
1200 RETURN
1210 IF o%(z%)<>R% THEN PRINT"It wasn
't here to hit":RETURN
1220 IF o%(11)<>1 THEN a=36:GOSUB 220
0:RETURN
1230 IF z%<>6 AND z%<>3 THEN PRINT "T
his has no effect whatsoever."
1240 IF z%=6 AND a% THEN a%=0:a=37:GO
SUB 2200:PRINT:o%(6)=0:RETURN
1250 IF z%=3 AND s%=0 THEN a=38:GOSUB
 2200:s%=-1:o%(8)=2:RETURN
1260 IF z%=3 AND s%=-1 THEN a=39:GOSU
B 2200:RETURN
1270 RETURN
1280 IF z%=8 AND e% THEN a=40:GOSUB 2
200:o%(z%)=0:e%=NOT e%:o%(11)=1
1290 RETURN
1300 IF z%=7 AND NOT b% AND l% THEN a
=41:GOSUB 2200:PRINT:l%=0
1310 IF z%=9 THEN a=42:GOSUB 2200:o%(
z%)=0:o%(12)=3
1320 IF z%=10 THEN d%=0
1330 RETURN
1340 IF o%(7)=5 AND o%(12)=5 THEN a=4
3:GOSUB 2200:b%=0:o%(12)=0:o%(7)=1
1350 RETURN
1360 IF z%(>10 THEN GOTO 1400
1370 IF d% THEN a=44:GOSUB 2200:GOTO
380
1380 IF (NOT l%) OR (o%(7)<>8 AND o%(
7)<>1) THEN a=45:GOSUB 2200:GOTO 380
1390 IF o%(2)=0 THEN a=46:GOSUB 2200:
```

```
o%(1)=0:o%(2)=8:o%(8)=8
1400 IF z%=7 AND NOT b% THEN l%=-1:a=
47:GOSUB 2200:PRINT
1410 IF z%=9 OR z%=12 THEN a=48:GOSUB
 2200:PRINT:a=49:GOSUB 2200:GOTO 380
1420 RETURN
1430 IF I%=1 THEN a=10:GOSUB 2140
1440 IF I%=2 THEN a=11:GOSUB 2140
1450 IF I%=3 AND NOT s% THEN a=12:GOS
UB 2140
1460 IF I%=3 AND s% THEN a=14:GOSUB 2
140
1470 IF I%=4 THEN a=15:GOSUB 2140:a=1
6:GOSUB 2200:a=17:GOSUB 2200
1480 IF I%=5 THEN a=18:GOSUB 2140
1490 IF I%=6 THEN a=19:GOSUB 2140
1500 IF I%=7 AND NOT b% AND NOT l% TH
EN a=21:GOSUB 2140
1510 IF I%=7 AND NOT b% AND l% THEN a
=22:GOSUB 2140
1520 IF I%=7 AND b% THEN a=23:GOSUB 2
140
1530 IF I%=8 THEN a=20:GOSUB 2140
1540 IF I%=9 THEN a=24:GOSUB 2140
1550 IF I%=10 AND d% THEN a=25:GOSUB
2140
1560 IF I%=10 AND NOT d% THEN a=28:GO
SUB 2140
1570 IF I%=11 AND NOT a% THEN a=26:GO
SUB 2140
1580 IF I%=11 AND a% THEN crip$="A "+
j$(I%):PEN 3:PRINT crip$
1590 IF I%=12 THEN a=27:GOSUB 2140
1600 PRINT
1610 RETURN
1620 DATA iluh,;,,sloh@ri@dvkhv,3,yhqg
lqj@pdfklqh,5,pluuru,9,wuroo,8,gzdui,
":",odps,3,frlq,3,sdufkphqw,7,fxuwdlq
,<,vzrug,3,yrxfkhu,3
1630 DATA "d@udwkhu@vsduvh,@xqiulhqgo
!@@urrp@zlwk@wkh@dssduhqwo!@lqh(sol@d
eoh@@@vljq@+Duprxu!+@rq@wkh@zdoo!
1640 DATA "d@ydvw@fdyhuq@zlwk@d@odujh
@@@srro@ri@zdwhu1@D@sdwk@ohdgv@dorqj@
lwv@@@hgjh!
1650 DATA "dq@h(wuhpho!@wljkw,@Hdvw@@
@@@Zhvw@wxqqho!
1660 DATA "d@vkrs1@Ryhu@wkh@frxqwhu@l
v@dvljq=@Hehqhh}hu+v@Hqwhusulvhv!
1670 DATA "d@vpdoo@fkdpehu,@lwv@zdoov
@@@lulghvfhqw@zlwk@vsdunolqj@fulvwdov
,vdyh@rqh!
1680 DATA "d@udwkhu@jorrp!@fdyh,@@@@@
@@@uhplqlvfhqw@ri@rog@plqh@zrunlqjvl"
1690 DATA "zkdw@orrnv@olnh@d@@@@@@@@@
@@@khuplw+v@fhoo!"
1700 DATA "dq@rog@gluw!@fkdpehu@wkdw@
@@@arrnv@wr@kdyh@ehhq@sloadjhg@orqj@d
jr1"
1710 DATA "@wkh@zl}dug+v@fkdpehu!@Dv@
qr@rqh@kdv@hyhu@vxuylyhg@wklv@h(shulh
qfh,@@wkhuh@lv@qr@h(wdqw@ghvfulswlrq1
111"
1720 DATA "exuqlqj@eulooldqwo!@lq&d@f
ruqhu!"
1730 DATA "qrz@txlwh@frro!"
1740 DATA "zlwk@d@qrwlfh@+qr@krw@prqh
!@ru@frxqwhuihlw@frlqv+!"
1750 DATA "zlwk@d@qrwlfh@+hpsw!+@rq@l
w!"
1760 DATA "lq@elwv@rq@wkh@iorru!"
1770 DATA "hqjudyhg="
1780 DATA "*Vd!@pluuru@zrug@wr@zl}dug
@phhw,"
1790 DATA "Eh@vxuh@!rx@kdyh@wkh@jrog@
wr@juhhw!*"
1800 DATA "zlwk@d@exvlqhvv@olnh@dlu!"
1810 DATA "zhdulqj@gdun@jodvvhv!"
1820 DATA "zruwk@lwv@zhljkw@lq@jrog!"
1830 DATA "qrw@!hw@olw!"
1840 DATA "exuqlqj@euljkwo!i"
1850 DATA "lq&d@udwkhu@edwwhuhg@frqgl
wlrq!"
1860 DATA "gu!@dv@d@erqh@dqg@frpsohwh
o!eodqn!"
1870 DATA "lq&txlwh@jrrg@frqglwlrq@@@
dqg@qrw@dw@doo@gdps!"
1880 DATA "fryhuhg@lq@eorrg!"
1890 DATA "uhdglqj=@H(fkdqjh@wklv@yrx
fkhuwrjhwkhu@zlwk@dq@rog@odps@dqg@!rx
+oo@jhwd@qhz@odps@iuhh$"
1900 DATA ",@zulqljlqj@zhwl"
1910 DATA "Wkh@zl}dug+v@h!hv@oljkw@xs
1@Kh@wdnhv@@@@!rxu@frlq&wr@ex!@klpvho
i@d@gulqn@dqg@@@@glvdsshduv!"
1920 DATA "\rx+@sxw@rq@wkh@furzq@kh+v@
ohiw@ehklqg@@@dqg@ilqg@!rxuvhoi@ed+n@
lq@wkh@sdodfh,@@@Nlqj@ri@Fuddo,@dqg@k
dss!@hyhu@diwhu!"
1930 DATA "Rk@ghdu,@!rx+yh@jrw@qr@prq
h!@@@dqg@wkh@@zl}dug@zdv@@krslqj@@!rx
+g@ex!@klp@d@@@@@slqw@ru@wzr!"
1940 DATA "Qdwxudoo!@kh@nloov@!rx1"
1950 DATA "Wkh@gzdui@uhixvhv@wr@ohw@!
rx+@sdvw!"
1960 DATA "D@zrug@wr@wkh@zlvh=@\rx+uh
@jrlqj@wr@wkh@rqo!@urrp@lq@wkh@sodfh@
zlwkrxw@lwv@rzq@@qdwxudo@skrvskruhvfh
qfh!"
1970 DATA "Dq@ludwh@gzdui,@lqixuldwhg
@e!@wkh@oljkw@!rx+yh@ohw@lq,@wkurzv@k
lv@xqolw@odps@dw@!rx!@Lw@odqqv,@lq@wk
h@urrp@!rx+yh@axvw@@ahiw,@vkdwwhuhg!"
1980 DATA "\rx@kdyhq+w@jrw@dq!wklqj@g
hfhqw@wr@klw@@zlwk!@Jhw@d@vzrug!"
1990 DATA "Wkh@gzdui@idoov@ghdg,@wkhq
@glvdsshduv@lqiurqw@ri@!rxu@hihv1"
2000 DATA "Wkh@yhqglqj@pdfklqh@vkdwwh
uv!@D@frlq@@@@gursv@rxw1"
2010 DATA "Judwxlwrxv@ylrohqfh@grhvq+
w@khos@dq!irqh1"
2020 DATA "Wkh@frlq@idoov@lqwr@wkh@vo
rw@lq@wkh@@@@@yhqglqj@pdfklqh1@\rx@du
h@jlyhq@d@vzrugl@Wkdw+v@zki@lw+v@fdoo
hg@wkh@Duprxu!$"
2030 DATA "Xqiruwxqdwho!,@!rxu@odps@j
rhv@rxw1"
2040 DATA "Lqwhuhvwlqj$@Vrphwklqj+v@k
dsshqlqj@wr@@@wkh@sdufkphqw1@Wdnh@d@o
rrn1111"
2050 DATA "\rx+yh@jrw@d@qhz@odps$"
2060 DATA "Wkh@fxuwdlq@idoduhv@xsl@\rx
@glh@lq@wkh@@@eod}h!"
2070 DATA "Wdnlqj@dgydqwdjh@ri@wkh@vx
gghq@soxqjh@@@lqwr@gdunqhvv,@wkh@gzdu
i@iurp@qh(w@grru@frphv@lq@dqg@srolvkh
v@!rx@riil"
2080 DATA "Wkh@iluh@jrhv@rxw1@\rx@vhh
@d@frlq@lq@lwvdvkhv!"
2090 DATA "\rxu@odps@oljkwv!"
2100 DATA "Lw@exuvwv@lqwr@iodph$"
2110 DATA "\rx@vxgghqo!@uhdolvh@wkdw@
lw@zdv@ylwdo@@wr@wkh@jdph!@Ryhu&rph@z
lwk@julhi,@!rx@@@nloo@!rxuvhoi"
2120 DATA "Wkh@pluuru@glvvroyhv@dqg@!
rx@ilqg@@@@@@!rxuvhoi@xsoohg@wkurxjk
@wkh@jds@lw+v@@@@ohiw@lqwr@wkh@zl}dug
+v@urrp!"
2130 DATA "Zkdw@d@vkdph@@@dqg@!rx@zhu
h@grlqj@vr@@@@zhoo@wrr!@Gr@!rx@zdqw@d
qrwkhu@jrB@+\2Q)"
2140 hh$=m$(m):GOSUB 560
2150 PEN 1
2160 crip$="You are in "+oo$:IF in TH
EN PRINT:PRINT crip$;
2170 IF NOT in THEN crip$="A "+j$(I%)
+" "+oo$:PEN 3:PRINT crip$
2180 in=0
2190 RETURN
2200 hh$=m$(m):GOSUB 560:PEN 1:PRINT:
PRINT oo$
2210 RETURN
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

Once you get the knack binary numbers are easy t



LAST month we had a look at binary numbers, the sort the micro uses. As we saw, a binary number consists of a lot of 0s and 1s together, each 1 or 0 being known as a bit. These bits normally come in groups of eight at a time, called bytes.

%10101100 is a typical binary number. Notice the % sign at the front – to distinguish it from our normal numbers. After all, we don't want anyone mistaking it for ten million, one hundred and one thousand, one hundred, do we?

How do we interpret it? Well the values meant by the 1s and 0s depend on the column they're in. We're used to this from the old htu (hundreds, tens and units) days of primary school – a 1 in the tens' column meant "one ten", whereas a one in the hundreds' column meant 100. So:

```
h t u
1 0 0
```

was very different from:

```
h t u
  1 0
```

It's much the same in binary. Every column has its value. Figure I shows these values for each column of a byte. Notice how the first column is column 0 – micros start counting at 0.

Column number	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

Figure I: Binary column values

To arrive at the value of a binary number, you just add together the values of the columns containing ones. Figure II shows how the %10101100 above translates into 172.

Once you get the knack, binary numbers are quite easy to handle. Bits and Bytes discusses some more binary techniques this month.

One drawback to using binary from a human's point of view is that it's extremely easy to make mistakes with. All those 1s and 0s can be very confusing, and you can end up writing:

%10110101

when you mean:

%10101101

It can be very difficult to spot where you've gone wrong. So when I deal with binary I always split the bits up into groups of four by putting a wiggly line down the middle. For example, I would write %10110101 as:

%1011≀0101

Value	128	64	32	16	8	4	2	1
Bit	1	0	1	0	1	1	0	0
=	128	+	32	+	8	+	4	
=	172							

Figure II: Translating binary

It's far easier for my eyes to see the pattern of the two sets of four than the whole eight at once.

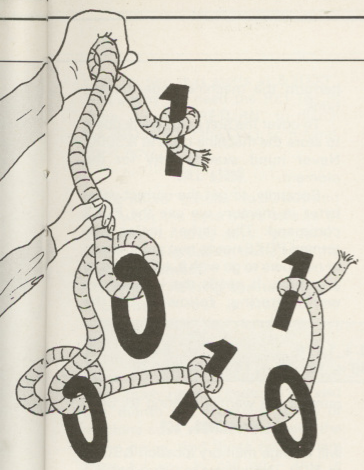
These sets of four are called nybbles. The left hand nybble, which contains the larger column values, is called the most significant nybble (MSN). The right hand one is called, not surprisingly, the least significant nybble (LSN). Figure III show what goes on with the least significant nybble.

Column	3	2	1	0
Value	8	4	2	1

Figure III: LSN values

I think that you'll immediately see that the biggest number it can hold is 15, when all the bits are 1. And, of course, the smallest number is 0. Figure IV shows all the patterns.

You've probably thought to yourself: "So what? Why should I write %1111 when it's easier to write 15?" And you've got a point. In fact we're going to make it easier than that – we're going to give each of our nybble



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

Figure V: Nybble patterns plus code

y to handle

Part II of MIKE BIBBY's guide to machine code

nybble pattern	value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Figure IV: Nybble patterns plus values

patterns a code, consisting of a single character.

The code for %0000 will be 0
The code for %0001 will be 1
The code for %0010 will be 2

Yes, you've guessed it, the code is the value of the pattern! So the code for %1001 will be 9.

But what about when we come to %1010? This has the value 10, but we can't use that for the code, since we can only use one digit.

All right then, it's our code, so let's decree that we use A as the code for %1010. Okay, I know it's really 10 and you know it's really 10, but our rule says only one character in the code, so A it is.

You won't be surprised to learn that %1011, which has value 11, will have code B, %1100, which has value 12, has code C, and so on up to %1111 value 15, which has code F.

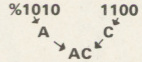
Figure V shows all the nybble patterns, this time with their codes and values.

What we've really done is to invent a rule that says, instead of counting 1, 2, 3 . . . 9, 10, 11, 12, 13, 14, 15 we count 1, 2, 3 . . . 9, A, B, C, D, E, F.

For the moment ignore what happens when we get to 16!

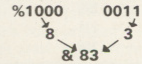
Of course the micro deals in bytes, but we can use our new code for these, too. After all, the most significant nybble is a pattern of four bytes, just like the least significant nybble, so all we do is code each nybble separately, but in the same manner.

Here's how:



If you look back to Figure II you'll see that %10101100 is 172, so AC is really another way of coding 172. To show that AC is really a number coded in our new way, we'd better prefix it with &, as we did with % for binary coding.

Here's another example:



Here %10000011 is 131, and coded in our new way this is &83. Notice how vital the "&" is. If we didn't have it we would mean 83 in our ordinary system of numbering (called denary) – that is, 8 tens and 3 units – and not 131 as we had intended.

So with this new method you can code any byte in two characters, one for each nybble. For example 255 would be &FF. See if you can work out why.

This form of coding is called hexadecimal – hex for short. You'll understand why in a minute.

Just as each column has a value in binary coding, so each has a value in hexadecimal. Let's see what they are.

The binary for 15 is %00001111 ----> hexadecimal &0F.

The binary for 16 is %00010000 ----> hexadecimal &10.

What's going on? Well, to obtain 16 from 15, we have to add one, so our hex sum is

```
  &0F +
  &01
 &10
```

What's happening is that we "put 0 down, carry one" when we get to 16. In normal sums we do this when we get to 10 – and the 1 goes in the tens column.

In hexadecimal we carry when we get to 16 – each unit in the second column is worth 16. That's why I said don't worry about what happens when we get to 16. When we get there, we carry.

This, incidentally, is why it's called hexadecimal – the hex is for six, the decimal for ten – hexadecimal, the

Hex Digit	Value in 16's column	Value in 1's column
0	0	0
1	16	1
2	32	2
3	48	3
4	64	4
5	80	5
6	96	6
7	112	7
8	128	8
9	144	9
A	160	10
B	176	11
C	192	12
D	208	13
E	224	14
F	240	15

Figure VI: Hexadecimal Ready Reckoner

number code based on sixteen.

This means that:

&10 is 1 sixteen + 0 units = 16
&17 is 1 sixteen + 7 units = 23
&1B is 1 sixteen + 11 units = 27
&20 is 2 sixteens + 0 units = 32
&3C is 3 sixteens + 12 units = 60
&A3 is 10 sixteens + 3 units = 163

and so on . . .

That 16s column can present you with some hard sums, so Figure VI provides a sort of ready reckoner, telling you the value of a hex digit for each column.

Fine, we can now write the value of a byte as a two digit hexadecimal number. This however is limited to a maximum of &FF, or 255.

As we saw last month, we need larger numbers than this to specify the 65536 memory locations the Z80 addresses. What we did was to specify our memory address with two bytes, a hi byte and a lo byte. In hexadecimal it's two digits to a byte, so we can specify any address with four digits (instead of the cumbersome 16 bits we used last month). Figure VII shows how it works.

Just as in binary we kept doubling our column values, so in hex we keep multiplying by 16. Hence the following values apply:

column 1 1 = 1
column 2 1x16 = 16
column 3 16x16 = 256
column 4 256x16 = 4096

This is quite sensible when you think about it. If the lo byte were &FF (255), that is as big as a single byte can hold, and we added one, we'd have to show somehow that we'd arrived at 256. We can do this by adding a one to the first column of the hi byte (which is worth 256) and setting the lo byte to &00 – in other words, put 0 down and carry 1.

The sums might get a bit hard if your mental arithmetic is a bit rusty, but you can translate from four digit hex addresses quite straightforwardly.

For instance:

&0202 = 2x256 + 2 = 514
&1A10 = 1x4096 + 10x256 + 1x16 = 6672
&FFFF = 15x4096 + 15x256 + 15x16 + 15x1 = 65535

If you want to translate from our normal numbering system (denary) to binary or hex, the Amstrad makes it easy for you. To see the hex equivalent of 6672, enter:

```
PRINT HEX$(6672)
```

and you'll see displayed 1A10 (regrettably without the "&").

To see the binary equivalent of 25, enter:

```
PRINT BIN$(25,8)
```

and you'll see 00011001. The reason for the eight after the comma is so that you get all eight bits of the byte shown. If you're using numbers bigger than 255 – that need two bytes – change the 8 to 16.

Unfortunately the Amstrad isn't as useful at translating from binary or hex to denary, but that's fine – I want you to start thinking in hex (and to some extent binary), and not keep referring to familiar old denary. It soon becomes second nature, and is extremely useful.

Anyway, enough of this number theory. Let's do some machine code. Last month we discovered that all machine code was a sequence of bytes in memory that gave the micro certain tasks. All you did was point the micro at the first byte of the machine code and say go.

That gives us three problems. We have to decide where in memory to put our machine code, we have to actually put the correct sequence of bytes into those memory locations and finally we have to tell the micro to perform the machine code instructions.

To cover the first point we're going to store the machine code at &3000. Never mind exactly why for the moment.

Secondly, to get the correct value bytes in memory we use the POKE command. The rather inelegantly termed POKE needs two parameters, or numbers to go with it, separated by a comma. It needs the location we want altering, followed by the number we want it altering to. So:

```
POKE &3000, &C9
```

will change memory location &3000 to &C9. (Notice we're using hex.)

Try it. As nothing seems to have happened, the more sceptical of you might be wondering if I've got it right. Has the value of the contents of memory location &3000 really been altered to &C9 by that POKE?

Well we can use a command called PEEK to examine the contents of memory locations. Try:

```
PRINT PEEK( &3000 )
```

You'll get the answer 201 – but remember, this is in denary, not hex. To really check up on me you'll have to use the more cumbersome:

```
PRINT HEX$( PEEK( &3000) )
```

You should now see C9 displayed. So the POKE really has worked!

Now, believe it or not, we've just entered a one byte machine code program into memory. You see &C9 is the instruction, or opcode, that tells the Z80 to "return from whence you came". Its mnemonic (an easily remembered "codeword") is RET. We met it last month in its denary form, 201.

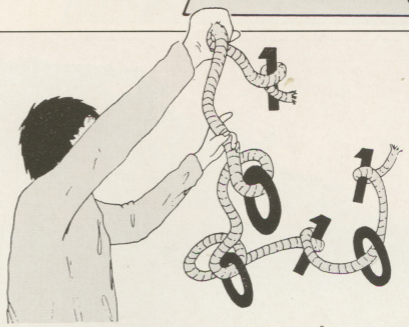
Right, we've put our machine code program at &3000. How do we get our Amstrad to do it? We use:

```
CALL &3000
```

This, if you like, is a sort of "GOSUB from Basic to the machine

Column	hi byte		lo byte	
Number	3	2	1	0
Value	4096	256	16	1

Figure VII: Four digit hex column values



code routine that starts at &3000". That's why we need the RET (&C9) there. It's the "RETURN" to match the "GOSUB" of the CALL command. Try it now. Enter:

```
CALL &3000
```

The micro "gosubs" to &3000 and does what the byte there tells it. In this case it merely says "go back", and you end up back in Basic, with the Ready statement.

All right, nothing spectacular has happened – but it worked, so congratulations! You've just run your first machine code program.

Let's try something that does a bit more than this, though, to prove we're really running code. As we said last month the firmware is full of machine code routines that keep the micro going. And those routines are there for you to call on – they're already written!

Let's use one. There's a routine that starts at &BBDB that clears the graphics screen. To demonstrate it, let's call it directly from Basic by entering:

```
CALL &BBDB
```

As you'll see, it works. What I want to do, though, is to call this routine from one of my own machine code programs. After all, I might want to clear the screen in the middle of one of my magnificent machine code creations, and I don't want to have to go into Basic to do it!

The Z80 has a useful instruction that lets us call (or "gosub to") another machine code routine – the code for it is &CD. You give the Z80 the address you want it to "gosub to" in the two bytes directly following the &CD in memory.

So to clear the screen with the routine at &BBDB you might think we want the following sequence of bytes in memory:

```
CD  BB  DB
```

You'd be wrong, though. The Z80 expects memory addresses to be given to it "back to front" – that is, lo byte first, followed by hi byte.

So to call the routine at &BBDB the sequence is:

```
CD DB BB
```

The mnemonic for &CD is CALL, by the way. Don't confuse it with the Basic call. However we've missed out something very important in our program – the RET (&C9). So our complete sequence of bytes, starting at memory location &3000, would be:

```
CD DB BB C9
```

Figure VIII illustrates the program. Now let's get the program into memory. Enter:

```
POKE &3000, &CD
POKE &3001, &DB
POKE &3002, &BB
POKE &3003, &C9
```

If you like, you can check up on it with PEEK to see if it's there.

When you're ready, call your routine with:

```
CALL &3000
```

and your screen should clear. Well done – your first machine code program that actually has a visible effect!

Type some rubbish onto the screen then:

```
CALL &3000
```

again. As you'll see, your routine is still there.

You must take care with your machine code programs – it's all too easy to crash your machine or wipe out your program. For instance, try CALL &0 and you'll see what I mean.

If you now call &3000 you'll see that your code has gone. It's often tiny errors that cause such disasters. The solution is simple – check and recheck before each Enter.

As I mentioned last month, you can get special programs called assemblers that let you type in your programs in the more meaningful mnemonics rather than as a list of hex bytes. If you wrote the above program on an assembler, and then listed it, it would look something like Figure IX.

address	opcodes	mnemonics
3000	CD DB BB	CALL &BBDB
3003	C9	RET

Figure IX: Assembler listing

The mnemonics are what would be typed into the assembler. The address column shows the address of the starting byte for each instruction, and the opcodes column shows the opcode and associated data bytes for each instruction.

Well that's all for this month. Next month we'll be looking at better ways of entering machine code – and learning new instructions to enter.

<table>
<tr><td>Memory address</td><td>&3000</td><td>&3001</td><td>&3002</td><td>&3003</td></tr>
<tr><td>Contents of memory location</td><td>&CD</td><td>&DB</td><td>&BB</td><td>&C9</td></tr>
<tr><td rowspan="2">Meaning of above bytes</td><td rowspan="2">CALL the subroutine at address specified by next two bytes</td><td>lo byte</td><td>hi byte</td><td rowspan="2">RETurn from whence you came</td></tr>
<tr><td colspan="2">Address of subroutine required by previous byte (lo byte first)</td></tr>
</table>

Figure VIII: Our simple clear screen program

DIGGER!

By PAUL BENEDICT

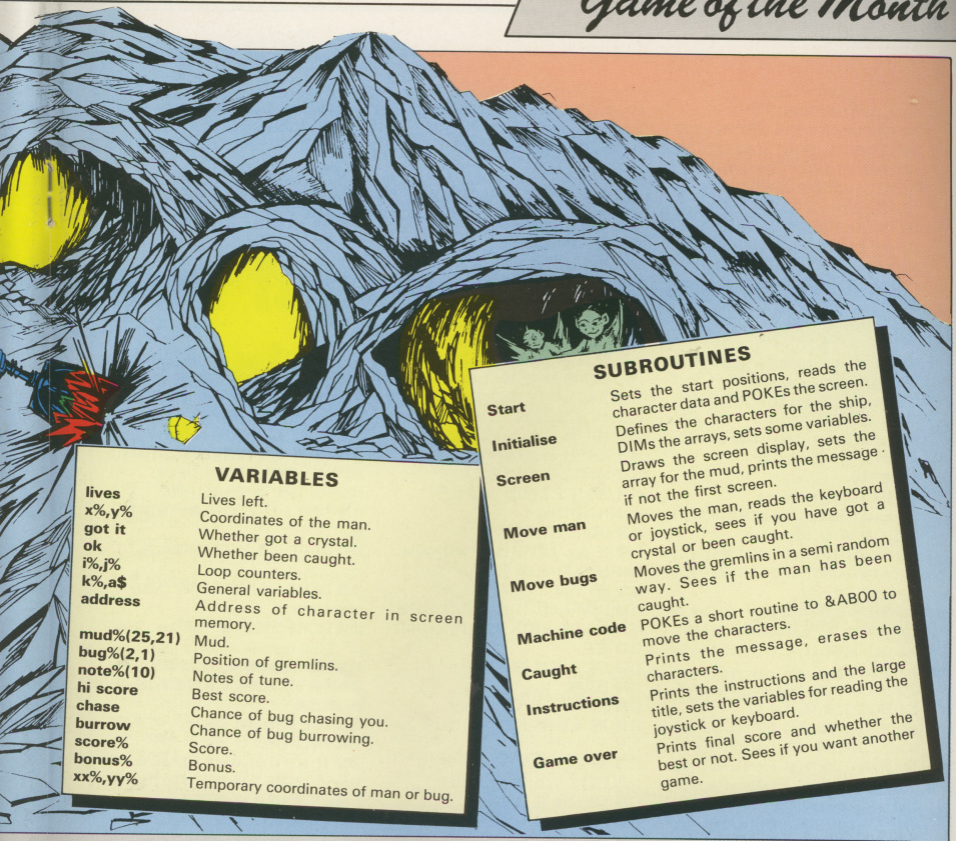
YOU are on your way back to Starbase 11 when your engines begin to splutter. After frantically searching the star charts you head for the nearest planet, Delta, which has the necessary dilithium crystals to recharge the ship's warp drive.

The touchdown was a bit bumpy but you survive. Your task is now to collect those crystals. They are buried beneath the mud in a cavern inhabited by gremlins.

They don't mind you taking a few as long as you keep out of their way, but the more you take, the more they chase you.

The game has been broken down into a series of subroutines, each with a specific task. All the subroutines have a title in a REM statement at the start.

```
10 REM ***** Digger *****
20 REM *By Paul Benedict*
30 MEMORY &AAFF
40 MODE 1
50 GOSUB 1200 :REM instructions
60 GOSUB 1020 :REM machine code
70 GOSUB 430 :REM initialise
80 MODE 0
90 WHILE a$="Y"
100 WHILE lives
110 GOSUB 630 :REM screen
120 WHILE collected%<8 AND lives
130 GOSUB 250 :REM start
140 WHILE ok AND collected%<8
150 GOSUB 820 :REM move man
160 GOSUB 910 :REM move bugs
170 WEND
180 IF NOT ok THEN GOSUB 1090 :REM ca
ught
190 WEND
200 WEND
210 GOSUB 1440 :REM game over
220 WEND
230 MODE 1:PEN 1
240 END
250 REM *** start ***
260 LOCATE 12,25:PRINT "Lives:";USING
 "#";lives
270 RESTORE 280
280 DATA 0,20,40,0,0,60,60,0,0,20,40,
0,84,252,252,168,168,252,252,84,40,84
,168,20,0,168,84,0,84,168,84,168
290 DATA 3,2,1,3,2,2,1,1,0,3,3,0,1,0,
0,2,2,130,65,1,2,0,0,1,2,65,130,1,1,3
,3,2
300 DATA 5,15,15,10,79,143,79,143,79,
5,10,143,15,15,15,15,5,79,143,10,5,14
3,79,10,15,10,5,15,10,0,0,5
310 DATA 179,51,51,115,115,162,81,179
,115,0,0,179,34,81,162,17,17,115,179,
34,17,17,34,34,17,0,0,34,179,0,0,115
320 address=&C000+&50+8:GOSUB 400 :RE
M man
330 address=&C000+20+&50+16:GOSUB 400
 :REM bug 1
340 address=&C000+20+&50+36:GOSUB 400
 :REM bug 2
350 address=&C000+20+&50+56:GOSUB 400
 :REM bug 3
360 bug%(0,0)=5:bug%(0,1)=21:bug%(1,0
)=10:bug%(1,1)=21:bug%(2,0)=15:bug%(2
,1)=21
370 x%=3:y%=2:got.it=0:ok=-1
380 LOCATE 8,2:PRINT SPC(10)
```



VARIABLES

lives	Lives left.
x%,y%	Coordinates of the man.
got it	Whether got a crystal.
ok	Whether been caught.
i%,j%	Loop counters.
k%,a$	General variables.
address	Address of character in screen memory.
mud%(25,21)	Mud.
bug%(2,1)	Position of gremlins.
note%(10)	Notes of tune.
hi score	Best score.
chase	Chance of bug chasing you.
burrow	Chance of bug burrowing.
score%	Score.
bonus%	Bonus.
xx%,yy%	Temporary coordinates of man or bug.

SUBROUTINES

Start	Sets the start positions, reads the character data and POKEs the screen.
Initialise	Defines the characters for the ship, DIMs the arrays, sets some variables.
Screen	Draws the screen display, sets the array for the mud, prints the message if not the first screen.
Move man	Moves the man, reads the keyboard or joystick, sees if you have got a crystal or been caught.
Move bugs	Moves the gremlins in a semi random way. Sees if the man has been caught.
Machine code	POKEs a short routine to &AB00 to move the characters.
Caught	Prints the message, erases the characters.
Instructions	Prints the instructions and the large title, sets the variables for reading the joystick or keyboard.
Game over	Prints final score and whether the best or not. Sees if you want another game.

```
390 RETURN
400 REM *** poke character data ***
410 FOR i%=0 TO 7:FOR j%=0 TO 3:READ
k%:POKE address+j%+&800*i%,k%:NEXT:NE
XT
420 RETURN
430 REM *** initialise ***
440 ENT -1,50,10,5
450 SYMBOL 248,3,14,117,255,146,146,2
55,255
460 SYMBOL 249,192,112,174,255,73,73,
255,254
470 SYMBOL 250,255,170,85,42,63,32,32
,80
480 SYMBOL 251,248,168,88,172,252,22,
19,41
490 SYMBOL 252,0,1,10,0,189,189,0,0
500 SYMBOL 253,0,128,80,0,182,182,0,0
510 SYMBOL 254,0,85,42,21,0,0,0,0
520 SYMBOL 255,0,80,160,80,0,0,0,0
530 DATA 0,9,12,0,3,10,16,2,6,15,8,20
,24,13,23,26
540 RESTORE 530:FOR i%=0 TO 15:READ j
%:INK i%,j%:NEXT
550 INK 3,13,26:BORDER 0
560 DIM mud%(25,21),bug%(2,1),note%(1
0)
570 DATA 379,319,239,213,190,478,159,
142,379,119,319
580 FOR i%=0 TO 10:READ note%(i%):NEXT
590 FOR j%=3 TO 20:mud%(2,j%)=32:NEXT
600 hi.score=10:score%=0:lives=3:chas
e=1:burrow=0
610 a$="Y"
620 RETURN
630 REM *** screen ***
640 IF score%>0 THEN LOCATE 4,10:PRIN
T CHR$(22);CHR$(1);"CONGRATULATIONS";
CHR$(22);CHR$(0):FOR i%=0 TO 3000:NEX
T:score%=score%+bonus%
650 CLS:RESTORE 660
660 DATA 248,249,8,8,10,250,251,8,8,1
1,252,253,8,8,10,254,255
670 LOCATE 1,1:PEN 14:PRINT CHR$(22);
CHR$(1);
680 FOR i%=1 TO 17
690 IF i%=10 THEN PEN 2
700 READ j%:PRINT CHR$(j%);
710 NEXT
720 PEN 1
730 FOR i%=3 TO 20:LOCATE 1,i%:PRINT
STRING$(20,CHR$(207));:FOR j%=1 TO 20:
mud%(i%,j%)=207:NEXT:NEXT
740 FOR i%=21 TO 23:FOR j%=1 TO 20:mu
```

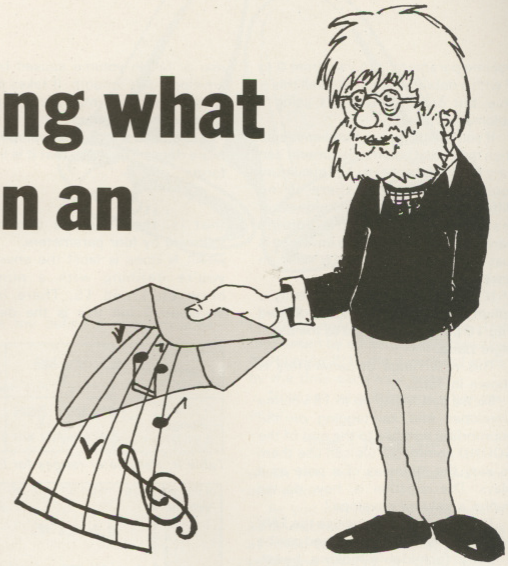
```
d%(i%,j%)=32:NEXT:NEXT
750 FOR j%=2 TO 8 STEP 2:LOCATE j%,23
:PRINT CHR$(15);CHR$(3);CHR$(233);CHR
$(8);CHR$(15);CHR$(4);CHR$(202);CHR$(
8);CHR$(15);CHR$(5);CHR$(148):mud%(23
,j%)=202:NEXT
760 FOR j%=13 TO 19 STEP 2:LOCATE j%,
23:PRINT CHR$(15);CHR$(3);CHR$(233);C
HR$(8);CHR$(15);CHR$(4);CHR$(202);CHR
$(8);CHR$(15);CHR$(5);CHR$(148):mud%(
23,j%)=202:NEXT
770 PEN 1:PRINT STRING$(20,CHR$(208))
;CHR$(22);CHR$(0);
780 score%=score%+bonus%:PEN 15:PRINT
 "Score:";score%
790 PEN #1,2:LOCATE #1,10,1:PRINT #1,
"Bonus 500"
800 collected%=0:chase=chase-0.1:burr
ow=burrow+0.1:bonus%=500
810 RETURN
820 REM *** move man ***
830 bonus%=bonus%+(bonus%>0):LOCATE #
1,15,1:PRINT #1,bonus%
840 xx%=x%+(INKEY(c%)>-1)-(INKEY(d%)>
-1):yy%=y%+(INKEY(a%)>-1)-(INKEY(b%)>
-1)
850 char%=mud%(yy%,xx%):IF char%=0 OR
 (char%=202 AND got.it) THEN RETURN
860 CALL &AB00,x%,y%,xx%,yy%:x%=xx%:y
%=yy%:mud%(y%,x%)=32
870 IF char%=202 THEN got.it=1:INK 7,
2,1:SOUND 2,30,40,15,0,1
880 IF x%=3 AND y%=2 AND got.it THEN
SOUND 2,200,200,15,0,1:score%=score%+
10:LOCATE 7,25:PRINT score%:mud%(y%,x
%)=32:got.it=0:collected%=collected%+
1:INK 7,2
890 IF (x%=bug%(0,0) AND y%=bug%(0,1)
) OR (x%=bug%(1,0) AND y%=bug%(1,1))
OR (x%=bug%(2,0) AND y%=bug%(2,1)) TH
EN ok=0
900 RETURN
910 REM *** move bugs ***
920 SOUND 132,note%(INT(RND*11)),20,1
4:SOUND 129,478,200,12
930 FOR i%=0 TO 2
940 IF RND>chase THEN xx%=bug%(i%,0)+
(x%(bug%(i%,0))-(x%>bug%(i%,0))):yy%=b
ug%(i%,1)+(y%(bug%(i%,1))-(y%>bug%(i%
,1)) ELSE xx%=bug%(i%,0)+INT(RND*3)-1
:yy%=bug%(i%,1)+INT(RND*3)-1
950 char%=mud%(yy%,xx%)
960 IF char%=202 OR char%=0 OR char%=
255 OR (char%=207 AND RND>burrow) THE
N RETURN
970 CALL &AB00,bug%(i%,0),bug%(i%,1),
xx%,yy%
980 mud%(bug%(i%,1),bug%(i%,0))=32:mu
d%(yy%,xx%)=255:bug%(i%,0)=xx%:bug%(i
%,1)=yy%
990 IF xx%=x% AND yy%=y% THEN ok=0
1000 NEXT
1010 RETURN
1020 REM *** machine code ***
1030 RESTORE 1070
1040 FOR i%=0 TO 64
1050 READ a$:POKE &AB00+i%,VAL("&"+a$
)
1060 NEXT
1070 DATA DD,46,00,DD,5E,02,CD,2D,AB,
E5,DD,46,04,DD,5E,06,CD,2D,AB,D1,01,0
0,08,C5,06,04,7E,36,00,12,23,13,10,F8
,01,FC,07,09,EB,09,EB,C1,10,EB,C9,1D,
21,00,C0,55,A7,CB,13,CB,13,19,05,C8,1
1,50,00,19,10,FD,C9
1080 RETURN
1090 REM *** caught ***
1100 SOUND 2,90,50,15,0,1,1
1110 LOCATE 8,2:PRINT "G U L P"
1120 lives=lives-1
1130 LOCATE x%,y%:PRINT " "
1140 IF got.it THEN collected%=collec
ted%+1:got.it=0:INK 7,2
1150 FOR i%=0 TO 2
1160 LOCATE bug%(i%,0),bug%(i%,1):PRI
NT " ":mud%(bug%(i%,1),bug%(i%,0))=32
1170 NEXT
1180 FOR i%=0 TO 2000:NEXT
1190 RETURN
1200 REM *** instructions ***
1210 INK 0,0:INK 1,0:INK 2,15:INK 3,1
1
1220 BORDER 0:PLOT -5,-5,3:PEN 1
1230 LOCATE 1,25:PRINT "DIGGER"
1240 FOR i%=0 TO 96 STEP 2
1250 FOR j%=0 TO 16 STEP 2
1260 IF TEST(i%,j%) THEN PLOT 200+i%*
2,365+j%*2:PLOT 200+i%*2,367+j%*2:PLO
T 202+i%*2,365+j%*2:PLOT 202+i%*2,367
+j%*2
1270 NEXT
1280 NEXT
1290 LOCATE 1,25:PRINT SPC(10):INK 1,
6
1300 LOCATE 1,5:PEN 2
1310 PRINT "On your way back to star
base 11 you    find yourself running
short of fuel.    Fortunately a nearb
y planet, Delta 2,   has the necessar
y dilithium crystals."
1320 PRINT:PRINT "Collect the crystal
s and take them back to your ship, bu
t watch out for the    gremlins who
inhabit the planet,they    are deadly
."
1330 PRINT:PRINT "At first they ignor
e you, but on later  screens they wil
l start to chase you."
1340 PEN 3:PRINT:PRINT:PRINT "U
se joystick or keyboard."
1350 PEN 2:PRINT:PRINT "A...up    Z...
down   <...left   >...right"
1360 PEN 1:LOCATE 1,25:PRINT "Press S
PACE BAR or FIRE to start..."
1370 a$="":WHILE INKEY$<>"":WEND
1380 WHILE a$<>" " AND JOY(0)<>16
1390 a$=INKEY$
1400 WEND
1410 CLS
1420 IF a$=" " THEN a%=69:b%=71:c%=39
:d%=31 ELSE a%=72:b%=73:c%=74:d%=75
1430 RETURN
1440 REM *** game over ***
1450 LOCATE 12,25:PRINT "Lives:0"
1460 LOCATE 6,10:PRINT CHR$(22);CHR$(
1);"GAME  OVER";CHR$(22);CHR$(0)
1470 FOR i%=0 TO 5000:NEXT:CLS
1480 PAPER 8:PRINT CHR$(30);STRING$(1
00," ")
1490 LOCATE 1,25:PRINT STRING$(20," "
);
1500 LOCATE 3,3:PEN 12:PRINT "Final s
core:";score%
1510 LOCATE 3,25:PRINT "Another  Game
 ?"
1520 MOVE 0,0:DRAW 0,399,15:DRAW 639,
399:DRAW 639,0:DRAW 0,0:MOVE 0,320:DR
AW 639,320
1530 MOVE 0,16:DRAW 639,16
1540 PAPER 0:PEN 11
1550 IF score%>hi.score THEN hi.score
=score%:LOCATE 3,10:PRINT "This is th
e best":LOCATE 7,15:PRINT "so far !"
ELSE LOCATE 5,10:PRINT "The best is":
LOCATE 8,15:PRINT hi.score
1560 a$=""
1570 WHILE INSTR(" YN",a$)<2
1580 a$=UPPER$(INKEY$)
1590 WEND
1600 score%=0:lives=3:chase=1:burrow=
0
1610 RETURN
```

Give your fingers a rest...

All the listings from this month's issue are available on cassette.

See our special offer on Page 55.

It's amazing what you find in an envelope



Part II of our series on putting sound on the Amstrad by NIGEL PETERS

LAST month we had a look at how to get the Amstrad to produce simple sounds using a very basic version of the SOUND command. If you understood what I said then you should be able to see why:

```
SOUND 1,200,100,5
```

produces a note of volume 5, on channel A with pitch 200, lasting for one second.

Make sure you follow this because we'll be using this particular note as the basis of our exploration of the Amstrad's volume envelope.

And what, you may ask, is a volume envelope? When we enter:

```
SOUND 1,200,100,5
```

we're rewarded with a fairly loud note. Notice that the loudness doesn't vary, it stays the same through the length of the note.

The trouble is that in real life notes don't always stay at the same loudness level. They fade away or grow louder as time goes on. We can't do this with a simple SOUND command we've used so far.

The note starts at loudness 5 and stays at that level until it stops a second later. However, by using a previously defined volume envelope we can make a note vary in loudness as it plays.

So, let's define a volume envelope with:

```
ENV 1,5,2,20
```

Don't worry just yet about what that means, just accept that this command defines a volume envelope that we can refer to as envelope number 1.

Now let's hear what effect this has on the note by entering:

```
SOUND 1,200,100,5,1
```

You should be able to hear the note getting louder and louder. It still plays for one second but the volume is changing.

The command is the same one we used previously except that ,1 is stuck on its end to tell the micro that the volume envelope previously defined as number 1 is to be used.

You can define up to 15 of these envelopes, numbered from 1 to 15. Once defined, you call them by attaching the appropriate number to the end of our simple sound command. The Amstrad then produces the note, varying its loudness in line with the volume envelope specified.

Before we go into the mechanics of how the volume envelope is defined, I must first admit that last month I told you what might be considered a small lie.

You'll remember I wrote that the

	Channel	Pitch	Duration	Volume		Volume Envelope
				without envelope	with envelope	
Range	1=A 2=B 4=C	0 to 4095	1 to 32767	0 to 7	0 to 15	0 to 15
Default				4	12	0

Table I: Parameter ranges for SOUND command

volume parameter can vary from 0 to 7 with a default value of 4. Well this is true, but only if you aren't using a volume envelope.

If you *are* using a volume envelope then the actual volume parameter can vary from 0 (silence) to 15 (maximum loudness). Actually there is no difference in the absolute loudness of maximum volume, for a volume parameter of 7 without an envelope is just as loud as one of 15 with an envelope.

It's just that when you specify a volume envelope the range is divided into 16 parts as opposed to the usual eight parts.

This new range of parameters is shown in Table I.

So we can define up to 15 volume envelopes and, by tagging on the appropriate number to the end of the SOUND command, we can use them to vary the loudness of a note as it plays. The question is, how do we define a volume envelope?

The answer is that we use the ENV command. Now this command can be a fairly formidable-looking beast, having up to 16 parameters following it.

When you get to know it, however, it's not all that bad. The secret is not to let it see that you're afraid of it.

Let's listen to the effects of envelope 1 again. Unless you've switched off, or redefined it, it should still be lurking in your micro, but just to make sure type in:

```
ENV 1,5,2,20
```

Notice that the ENV command doesn't make a noise itself. You can type them in until you're blue in the face but the Amstrad will stay mute. All a volume envelope does is (when asked) affect a SOUND command. The SOUND command is the one that makes the noise. If you don't believe me, enter:

```
SOUND 1,200,100,5,1
```

and you'll hear a note being affected by a sound envelope.

If you listen carefully, you should hear five distinct changes of loudness during the second that the note is playing. The note gets louder in steps of two and each step lasts for 20 hundredths of a second.

It's important to realise that the rise or fall in volume caused by an envelope isn't smooth. It takes place in steps. To be formal, the change is discrete, not continuous.

Let's have a look at envelope 1 in more detail. You'll see that it is in the form:

```
ENV N,P,Q,R
```

that is, the envelope command followed by four parameters.

All N does is label the envelope you're defining with a number between 1 and 15. There is an envelope 0 but this is the default volume envelope, fixed at two seconds at the same volume level as the normal volume parameter.

The P parameter just tells the micro how many steps the envelope is going to take. There can be up to 127 of these steps. In our volume envelope 1 there are five of these steps.

The Q parameter tells the Amstrad the increase or decrease in volume that is to occur with each step. the initial volume is taken from the volume parameter of the SOUND command. In the case of envelope 1

Parameter	Number N	Number of steps in section P	Volume change per step Q	Time length of each step R
Range	0 to 15	0 to 127	−128 to 127	0 to 255

Table II: Parameter ranges for ENV command

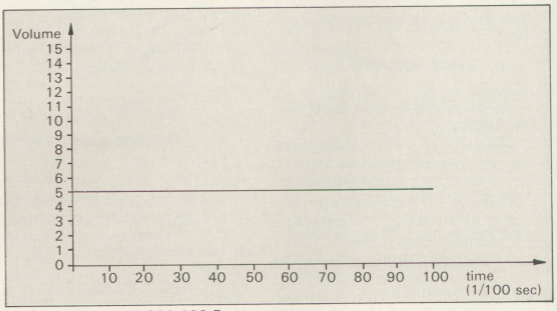


Figure I: SOUND 1,200,100,5

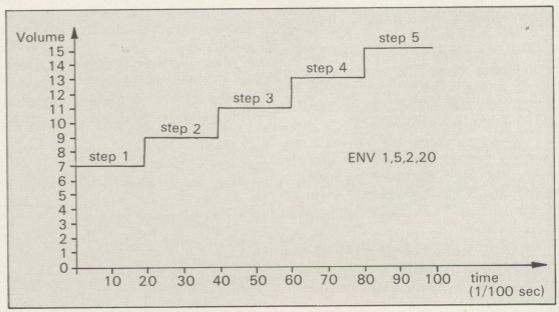
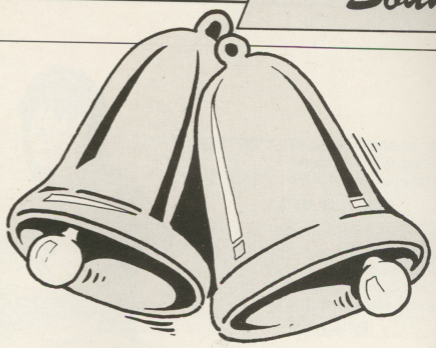


Figure II: SOUND 1,200,100,5,1



the volume increases by 2 volume units for each step of the envelope. The Q parameter can vary from –128 to 127 in value.

Finally the R parameter decides how long each step of the envelope takes. It is measured in hundredths of a second and can range in value from 0 to 255.

Table II sums up these parameters and their ranges.

Figure I shows the effects of:

```
SOUND 1,200,100,5
```

in graphic form.

As you can see, the volume level stays the same throughout the second that the note plays.

Now try our old favourites:

```
ENV 1,5,2,20
```

(if you've switched off or redefined the envelope) and:

```
SOUND 1,200,100,5,1
```

Figure II shows what happens. The envelope causes the volume of the note to increase during the second that it plays.

There are five steps, each lasting 20 hundredths of a second. Each step causes the volume to increase by two, the result of all the steps being that the volume goes from 7 to 15 in the second the note is playing.

Now have a change and define another volume envelope with:

```
ENV 2,5,1,20
```

You can hear this in action by entering:

```
SOUND 1,200,100,5,2
```

Volume envelope 2 is called by the parameter at the end of the SOUND command. Now the note increases in loudness again, but since the Q parameter is 1, it doesn't get as loud as before.

Incidentally, volume envelope number 1 is still lurking around the Amstrad's memory. You can hear this for yourself by entering:

```
SOUND 1,200,100,5,1
```

and getting the same result as before.

The volume envelope doesn't always have to increase the volume. It can decrease it as well. If you don't believe me define a new envelope with:

```
ENV 3,5,-1,20
```

and then call it with:

```
SOUND 1,200,100,5,3
```

Now the volume is decreased by a factor of 1 for each step forward in time.

While we're decreasing the volume, try:

```
ENV 3,5,-2,20
```

and:

```
SOUND 1,200,100,14,4
```

Bell-like, isn't it? Notice that the volume parameter of the SOUND command – the fourth figure – is 14. This means that the volume envelope starts just below maximum volume and decreases in steps of 2.

Try defining a few envelopes of your own and seeing the effects you can create. It's great fun.

While you're playing you may come across a few weird effects.

You may have noticed how, in the examples above, the duration of the SOUND command has always been one second. Similarly, there have always been five steps, each of which lasted 20 hundredths of a second.

In other words, the five steps of the volume envelope added up to one second, the duration of the SOUND command. You might wonder what happens if the SOUND duration is longer or shorter than the time taken by all the steps of the volume envelope. Let's see.

Define a new version of envelope 1 with:

```
ENV 1,5,2,10
```

There are now five steps, each of 10 hundredths of a second so the envelope span is 0.5 seconds. The time taken by an envelope is given by multiplying P*R, the number of steps by the time each step lasts.

Now see what happens when you enter:

```
SOUND 1,200,100,5,1
```

which tells the Amstrad to play for one second.

As you can hear the note quickly achieves maximum volume and stays at that level until the SOUND command finishes. In other words, the envelope only effects the note for the duration of the envelope span (P*R). The note carries on for its full duration, playing at the volume reached when the envelope finishes.

To see what happens in the opposite case, redefine envelope 2 with:

```
ENV 2,5,1,20
```

which gives five steps lasting a total of one second.

Now let's have a sound lasting half a second and hear what happens. Enter:

```
SOUND 1,200,50,5,2
```

and you'll hear the poor old envelope being cut off in its prime. If you want it to have its full effect use:

```
SOUND 1,200,100,5,2
```

Apart from discrepancies between the duration of volume envelopes and the SOUND commands that call them, there are other problems that can arise. What if there are too many steps in the envelope?

Try it and see. While:

```
ENV 1,10,2,20
```

expects 10 steps, each lasting a fifth ▶

of a second and needs a full two seconds to do its work.

Unfortunately:

```
SOUND 1,200,100,5,1
```

only lasts one second, so the envelope gets cut off in its prime, only having five steps.

The opposite case, where there are two few steps is illustrated by:

```
ENV 1,2,2,20
```

and:

```
SOUND 1,200,100,5,1
```

As you can hear, you get two steps – taking up 40 hundredths of a second – and the note remains at the final volume for the remaining 60 hundredths.

Now have a look at Table I again. You'll notice when an envelope is used, the volume parameters can range between 0 and 15 in value. So far, all the examples have been picked so that the parameters stay in this range.

What happens if a volume envelope tries to go out of range?

Define an envelope with:

```
ENV 5,5,3,20
```

This increases the volume in steps of three each time. Now consider the effect of this envelope on a SOUND command such as:

```
SOUND 1,200,100,5,5
```

The initial volume is 5. The first step of the envelope will make this 8, the second 11, and the third 14. But what happens when the fourth step tries to give it a value of 17 which is out of range? Try it and see.

As you can hear, the Amstrad is a crafty beast and, seeing that it is being asked to do something naughty, it just takes away 15. As you can hear the volume "wraps around", going suddenly from very loud to quiet again.

It can happen the other way as well. The envelope defined by:

```
ENV 6,5,-3,20
```

decreases the volume by 3 for each step of the envelope. When it works its wicked way on:

```
SOUND 1,200,100,5,6
```

the volume goes from 5 to 2 and then,

rather than go out of range, wraps around to 14 and carries on decreasing again. Try it and see.

By now you should be fairly comfortable with the volume envelope. You might also be aware of some of its limitations. One of them is that so far we can only use a volume envelope to make a note increase or decrease in volume.

In real life, however, notes sometimes do both, growing gradually louder, then fading away. In its present form of:

```
ENV N,P,Q,R
```

our volume envelope can't handle this. And once again I must confess to telling you rather less than the truth.

You see, the actual definition of a volume envelope isn't:

```
ENV N,P,Q,R
```

it is really:

```
ENV N,P1,Q1,R1,P2,Q2,R2,
    P3,Q3,R3,P4,Q4,R4,
    P5,Q5,R5
```

Instead of our friendly little ENV with four parameters, we've got a huge beast with 16 numbers after it. However, don't give up in disgust or fear, it's quite simple really.

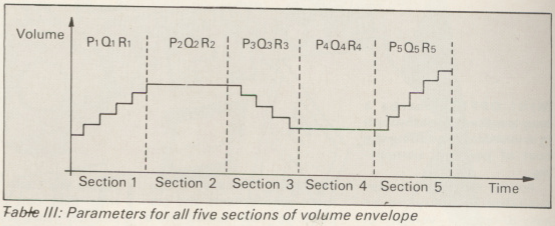


Table III: Parameters for all five sections of volume envelope

Up until now we've only been using a fraction of the volume envelope's parameters. Each envelope can have up to five sections, each section affecting the same SOUND command in a different way.

So far, we've only used the first section as a kindness to those of a nervous disposition.

Each of the five sections of the volume envelope works in exactly the same way as the first one that we've been concentrating on. All you have to do when figuring out an envelope is to treat each of its five sections as we did before.

The only difference is that instead of P, Q and R, the first section has parameters P1, Q1, R1, the second P2, Q2, R2 and so on. Figure III shows how the parameters relate to the sections.

Although you can have five sections in a volume envelope, as should be obvious from the above you don't have to have all five sections in use.

Let's define an envelope that will have the volume increasing, then decreasing, then increasing again.

We do this with:

```
ENV 1,5,2,20,5,-2,20,5,2,20
```

Here the first section of the envelope has the volume increasing by two for five steps. The second section has the volume decreasing by two for five seconds. I leave it to you to figure out what the third section does.

```
SOUND 1,200,300,5,1
```

will let you hear this three-sectioned envelope. Notice I've increased the duration of the SOUND command to three seconds. This ensures that all the sections get a chance to play.

And really that is all there is to the volume envelope. Although it looks difficult it's not that hard so long as you take it section by section. All it needs is a little practice to provide familiarity. and that's where Program I comes in.

Run it and you'll find it will allow you to create your own volume envelopes and hear the results. After an hour or so you'll find that you'll understand them completely.

And you may even be looking forward to next month's article on pitch envelopes.

```
10 REM PROGRAM I
20 REM VOLUME ENVELOPE
30 DIM P(5),Q(5),R(5)
40 WHILE -1
50 MODE 1
60 INPUT "How many sections in volume
 envelope?", sections
70 IF sections<1 OR sections >5 THEN
CLS:GOTO 60
80 CLS
90 FOR loop=1 TO sections
100 LOCATE 3,5:PRINT "Section" loop
110 LOCATE 3,8:PRINT "Number of steps
?"
120 LOCATE 30,8:INPUT P(loop)
130 IF P(loop)<0 OR P(loop)>127 THEN
LOCATE 30,8:PRINT SPACE$(8):GOTO 120
140 LOCATE 3,13:PRINT "Size of each s
tep?"
150 LOCATE 30,13:INPUT Q(loop)
160 IF Q(loop)<-128 OR Q(loop)>127 TH
EN LOCATE 30,13:PRINT SPACE$(8):GOTO
150
170 LOCATE 3,18:PRINT "Duration of st
ep?"
180 LOCATE 30,18:INPUT R(loop)
190 IF R(loop)<0 OR R(loop)>255 THEN
LOCATE 30,18:PRINT SPACE$(8):GOTO 180
200 LOCATE 14,23:PRINT "PRESS SPACE"
210 WHILE INKEY(47)=-1:WEND:CLS
220 WHILE INKEY$<>"":WEND
230 NEXT loop
240 ENV 1,P(1),Q(1),R(1),P(2),Q(2),R(
2),P(3),Q(3),R(3),P(4),Q(4),R(4),P(5)
,Q(5),R(5)
250 duration=P(1)*R(1)+P(2)*R(2)+P(3)
*R(3)+P(4)*R(4)+P(5)*R(5)
260 SOUND 1,200,duration,5,1
270 CLS
280 duration$=RIGHT$(STR$(duration),L
EN(STR$(duration))-1)
290 PRINT "SOUND 1,200,";duration$;",
5,1
300 FOR loop=1 TO sections
310 loop$=RIGHT$(STR$(loop),1)
320 PRINT "P(";loop$;")  ";P(loop)
330 PRINT "Q(";loop$;")  ";Q(loop)
340 PRINT "R(";loop$;")  ";R(loop)
350 NEXT
360 LOCATE 14,23:PRINT "PRESS SPACE"
370 WHILE INKEY(47)=-1:WEND:CLS
380 WEND
```

Program I

Now you can use your Amstrad CPC646 to write a letter or a report, to compile a mailing list or classify your record collection, to check your bank statement or sort out your family finances (and then translate them into colourful graphics) . . . all for just £5.95.

Quick to learn, easy to use, that's..

MINI OFFICE marks a long-awaited breakthrough in dramatically reducing the cost of personal computing.

For the first time it makes available to everyone an easy-to-operate version of four of the most popular business computing applications – and at a price anyone can afford.

Never before has a word processor been sold for anything as low as £5.95. Nor a database manager. Nor a spreadsheet. Nor a graphics program.

Yet Mini Office contains them all.

So how was it done?

It all started with a suggestion that we should prepare a package to give readers a gentle introduction to the kind of software that businesses were running on their computers.

At that stage there was no intention that it should be an ambitious package. Just a simple program that could be sold at a very low price.

We called in experts in business software programming, told them what we wanted and sat back to await results.

What happened next was totally unexpected. For they all came back with ideas that were to considerably expand our original brief.

In the end what had been planned as little more than a beginners' guide to word processing, database management, spreadsheets and graphics had been turned into a full scale suite of programs covering all four applications.

In fact the only part of the brief that remained was our original insistence that the package should be quick to learn and easy to use.

And despite all the extra sophistication that has been written into it, we decided that, as a service to our readers, the price should still be kept at the very low figure originally fixed.

How does Mini Office operate?

Using the **Word Processor** is simplicity itself. There are none of the cryptic coded instructions that had to be mastered by people learning the early word processors.

You start by selecting the size of type you prefer – either normal or double-size. The latter is a feature that you cannot find on any other word processor.

It is particularly suitable for the partially sighted – in many cases giving them their very first opportunity to use a word processor.

This means they can use an Amstrad to compose a letter, using the double-size mode, and then print it out using normal size type. For many people this could be the first time they can send out a perfectly typed letter without outside help.

Primary school teachers are also expected to make great use of the double-size function, both on the screen and on hard copy printouts.

```
A unique feature is
the double size text
option in both
printer and edit
mode - perfect for
young children and
people with poor
vision.
```

The word processor – with double size characters

While you are using the word processor three useful pieces of information are displayed across the top of the screen.

They tell you how much time has elapsed since you started using it, the number of words you have written so far, and how many characters you can key in before the Amstrad's memory in full.

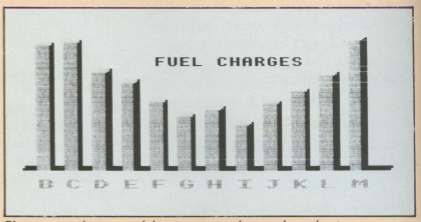
At any time you can press a key which tells you your typing speed. This is a most useful function, and can play an important part in increasing your efficiency at the keyboard.

You can also decide the size of the margin, the line length and the tab positions. Text can be moved from one part of the document to another.

At any time you can preview the text to see how it would look when printed out.

As with all the other programs in Mini Office, your work can be saved to tape and loaded when you want to use it again. It can also be printed out.

The **Database** program can be used to store a mass of information. It can be retrieved, in its entirety or just the parts you require for a particular purpose, whenever you need it.



Figures on the spreadsheet can produce a bar chart . . .

The operation is so simple that a useful database can be created in minutes rather than days – and you certainly don't need any computer experience to set it up.

The search facility is very easy to use. You can search for a particular word or part of a word. Or you can order a numeric search – such as telling the computer to find all the numbers greater or less than the one you provide.

You can carry out multiple sorts. For instance, if you have built up a mailing list containing a list of names, addresses, telephone numbers, occupations and ages you can ask the database to provide you with a list of teachers living in Liverpool whose ages range from 25 to 30.

One powerful option allows you to replace anything on the database without having to go through the whole lot making amendments yourself. You could, for instance, instruct it

Contains 32-page booklet giving clear, easy-to-follow instructions about all four programs

to find each reference to "teacher" and replace it with "lecturer".

The **Spreadsheet** is our version of the program that marked a milestone in business computing – Visicalc.

It is often pointed out that this one program alone has helped to sell more personal computers than any other.

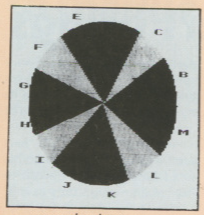
Certainly Visicalc and its derivitives have never been shaken from their position at the top of the list of best-selling business programs.

Yet the concept is very simple – a giant worksheet of rows and columns, only part of which can be seen on your screen at any one time. Into any position on the sheet you can put numbers, labels and mathematical formula.

And when you alter any figure its effect ripples through the rest of the sheet, changing any totals as may be necessary.

The Mini Office version is ideal for home finance, providing you with an effortless means of keeping tabs on your income and expenditure – and enabling you to work out your own budget.

In our Spreadsheet program – as well as in the Database – we have provided



. . . or a pie chart

a sample file so that you can experiment with it before entering your own data.

One feature we have included which to our knowledge does not exist in any other spreadsheet is a warning device to prevent you accidentally erasing formula – a very useful precaution.

The **Graphics** program uses the standard business graphics – line, bar and pie charts – in full colour. Which is something not always available on far more expensive graphics packages.

The program uses data you have already prepared on the spreadsheet. You have to identify which set of information you require to see in graph form – such as by indicating which row or column – and then which of the graphs you require.

The graph is then automatically configured exactly as you require it. If you have an Amstrad DMP-1 or other suitable printer capable of producing graphics you can also print out hard copies for a permanent record.

Because our original intention was to produce a package for people new to all these applications, we have produced a fully-detailed, easy to understand manual.

This 32 page free booklet gives clear instructions about how to use all four programs and in itself forms a concise introduction for first-time users.

If you want to start doing more with your Amstrad than just playing games, this package is your ideal introduction to the four most popular applications for professional computers.

Send for it today

Please send me _____ copy/copies of Mini Office for my Amstrad CPC464

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

ONLY £5.95

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.

Thieving...with no risk to your liberty

PLAYING **Blagger** by Alligata is the only way I have so far found of breaking, entering and thieving from a house without running any risk of being put inside.

If you have acquisitive tendencies and don't want to spend the rest of your life doing porridge then this ladder and levels type game is for you.

Your objective on each screen is the safe. All you have to do is collect the golden keys that are lying – perhaps stashed might be a better word – around the room.

However, there are various problems you will encounter during your escapades. These come in all types and varieties from falling too far down the screen, to the whirling motorised guardians.

These wander backwards and forwards over a set path which makes it possible to get away from them but only with great skill or luck.

That is what really marks this game out from some of the others. It is not simply a matter of collecting the keys but tends to revolve around the actual routes that you develop.

Inside each building there are collapsing floors, conveyor belts and even poisoned spikes so you have to keep your wits around you at every moment.

There is a great deal of strategy involved in solving each of the screens and as there are 20 of them you might end up playing it all night.

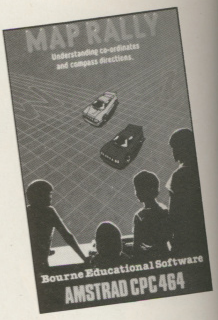
The graphics are good and the excellent use of colour makes the type of floor you are about to stand on easier to identify.

A remarkable feature of this version is the excellent stereo sound. If you weren't aware of it the Amstrad's three sound channels form a left, middle and right system allowing stereo output.

In order to hear this you have to amplify the output but if you do the results are excellent. This is the only game that I have yet seen to use this feature.

A good, almost classic game likely to keep you quiet for months instead of days.

Dave Carlos



The right direction

MAP RALLY is another program from the excellent Bourne stable which has been tried and tested on the BBC and Electron machines. This is perhaps a good thing, because the error-trapping has all been done already.

The idea is fairly simple with two players trying to drive around a grid reaching a number of predetermined checkpoints on the way. The routes of the two players are

The sight of 'Er*bert brings delight!

'ER*BERT has been available for some time for the Acorn micros which is perhaps why the cassette inlay tells the user to CHAIN "", rather than the LOAD or RUN which is used on the CPC464.

It is written in machine code which ensures speed of action although there are a few features which make me wonder.

The game itself seems to be rather too hard for young children yet the graphics seem to be designed for just that age-range. Perhaps the idea is to make us old-stagers seem younger at heart.

The graphic figures are rather fun and very well drawn with a coily snake and a gorilla called Boris making life awkward for 'Er*bert in his cubic domain. It is, of course a Q-Bert lookalike in which the

little figure has to visit each of a pyramid of cubes, changing their colours as he goes.

The magic discs are there from the original, which allow the figure to escape pursuit and perhaps to lure the pursuers over the edge. There is also a Rota-Hat which allows the long-nosed hero to fly up or down by two levels of cubes.

The sight of the 'Er*bert with a propeller stuck onto his head brought great delight to my children when they saw it.

There is also a banana, which allows double points to be scored, but also ensures the attentions of Boris.

I played it from the keyboard, and found the controls fairly easy to position, but I'm sure a joystick would greatly improve matters.

One feature I enjoyed when comparing it with a BBC Micro version from another house was the variety of layouts on the various screens. The screens reveal more difficult yet more interesting structures to negotiate.

There are also 10 skill levels with additional problems occurring on the harder stages. On the debit side, however, was the total lack of any kind of hi-score table.

Surely every self-respecting game demands one, yet this doesn't even allow for the input of the name of the highest scorer.

Indeed, on my copy, there was a bug on the score system, as I constantly went from 9999 to zero – hardly good for the ego.

Overall, this is an entertaining version with good 3-D graphics and reasonable use of sound. For many players, the lack of a hi-score table will be a definite drawback but it is after all a very inexpensive cassette.

With all its problems, it is still a fair buy.

Phil Tayler

different, avoiding any follow-my-leader tactics, although there are a similar number of checkpoints for each.

The size of the grid is determined by the difficulty level selected from 6 x 6 to 18 x 18 squares. Each driver is given a clue to the direction he should drive in. Selection of whether East or West is required, and how many squares (kilometres) to move are then made.

This is followed by similar inputs to determine the North/South movement.

Checkpoints are announced with a fanfare, while an incomplete leg brings a further clue from the new position.

There are three winners determined by the number of separate moves made, another resulting from the time taken for the players' moves, and a third result from the number of extra kilometres driven above the optimum.

Thus, it is perhaps educationally rather valid that the same child might not win on all three counts.

Following this program is a very similar one, but this time the player has to enter the specific coordinates which he/she wishes to move to. This is entered as Eastings first, followed by Northings.

This standard approach will be beneficial later on when O.S. maps are encountered.

These two programs would make this tape a very useful buy for the home, or indeed even schools should the Amstrad make the inroads it deserves.

In addition however, there is a facility to set a time limit so that more able players may be encouraged to compete against their previous performances.

There is even a monitor program which could allow a busy teacher or parent to look up on the screen just what has been happening while the children have been using the program.

An excellent program for practising and testing the idea of coordinates, compass bearings and so on which are presented in an entertaining manner while still retaining a real educational content.

Phil Tayler

Learn your letters with a hungry croc!

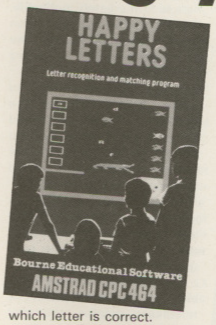
HAPPY Letters, another of Bourne Educational Software's products for the early learner, aims to give children in the three to six-year-old age range practice in simple letter recognition.

In addition it tests the child's ability to pair lower case letters with their upper case equivalents and teaches keyboard layout.

At first the child simply matches lower case letters. A column of letters appears on the screen, and a separate letter moves past them.

The child presses the Enter key when the letters match up. If he gets it right a tune plays, a fish swims out from the side, eating up both letters, then swims back.

If he is wrong he is given another go, then finally shown



which letter is correct.

When all the letters have been covered a voracious crocodile appears. It devours the fish belonging to the incorrectly matched letters while the correctly matched fish escape.

By far the most exciting part is when the croc eats the fish, so some children will be tempted to make deliberate mistakes to see the ensuing carnage.

The second stage is similar, though in this case the child has to match the moving letter with the initial letter of a simple word, in lower case at first and then in capitals.

Finally the child is given capital then lower case letters to match with the corresponding capitals on the keyboard.

The program worked well, maintaining the child's interest even if for the wrong reason. One minor niggle though – the shape of "a" used is not the one early learners are accustomed to.

Carole Sillers

Gems fail to sparkle

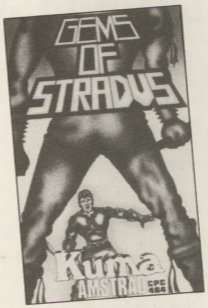
I'M feeling rather diffident about putting word processor – Amsoft's Easyword, and very nice too – to paper about **Gems of Stradus,** from Kuma Computers.

Had I really been that thick or insensitive? Piqued perhaps because I hadn't got so far into the adventure? The fact remains that the gems sparkled not at all for me.

To start with, I found myself getting bored rather rapidly with wandering through rooms devoid of interest apart from the eye-straining ever changing hues of walls and doors.

I did, admittedly, pick up keys galore, enough blazers to fit out the Brigade of Guards when in mufti, a pair of stilts and the best part of a box of Swan Vestas.

Incidentally, what in blazes are blazers doing in a macho



sword swinging (see cassette cover) adventure like this?

Even falling foul of hordes of nasties, the occasional homicidal ghost and repulsive and deadly reptiles did not get the blood tingling.

No, Kuma, you didn't offer enough early on to make me want to persevere. Doubtless things hot up later, but I'll leave it to a more determined adventurer.

Peter Gee

An old favourite

HUNCHBACK from Ocean is a well known game which has appeared on several micros and is one of the old favourites.

This version, for joysticks only, is pretty standard. It isn't the best I have seen – but is reasonable and quite enjoyable.

For those who haven't played it before, the time is the middle ages with knights, castles and damsels in distress.

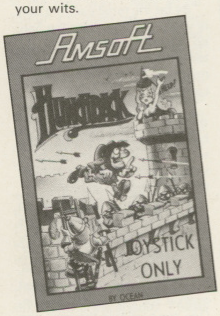
Esmerelda has been locked in the castle stronghold and you, as Quasimodo, must attempt to rescue her. You have to run along the ramparts of the castle, dodge flying arrows and fireballs, swing over pits on ropes and leap guards who attempt to thrust their spears into your nether regions.

As if this wasn't enough, there is a knight chasing you to ►

make sure that you don't hang around too long.

There are 15 screens, each successive one slightly harder than the previous with a different combination of obstacles to be negotiated.

At first there is only one type of hazard, but later on there are two or even three – you need a cool head and all your wits.



Every time you complete a screen you are awarded a bell. If you complete five screens on the trot without losing a life there is a large bonus.

There are a couple of bugs however which should have been corrected before the program was released. The first is immediately noticable. Quasimodo is not drawn on the first screen at the start, he only appears when you move him – wherever he is.

The other fault is when you enter your name into the hi-score table. You are asked to enter your initials but the problem is that you must enter five characters, no more and no less.

If your name or initials are less than five characters then you have to keep on pressing the Space Bar or Enter until the difference is made up.

And Delete doesn't work either, so you can't rub out any mistakes: graphic character 127 is printed which totally messes you up.

Apart from these minor gripes I quite like playing it. Overall it is simple but enjoyable with reasonable graphics, nice sound effects and many different screens to master.

It isn't very difficult and so would appeal to the slightly younger user rather than the hardened arcade addict looking for a new challenge.

Roland Waddilove

Not just another Pacman

FANCY yourself as a space fighter? Well get your silver space suit and your thermal underwear as you might be in your craft for a number of days if you start playing **Astro Attack** by Amsoft (Romik).

It is a rather strange combination of a grid game and a space game that involves a little forethought and superb gut reactions to be played well.

The nearest arcade game that I can liken it to is Pacman – but it is much more than this.

You are the pilot of a space fighter that is engulfed by

radioactivity and transported to another time where the Rigonites rule the land. They like to play with any spacecraft that gets trapped in their game grids by stalking them just for fun.

If they catch you, you end up dead, but not to worry – reincarnation is readily available and not as painful as you might expect.

Your weapons comprise a photon blaster that propels square – yes square – photons around the grid in an attempt to kill the nasties.

Unfortunately they usually turn a corner just before the photon hits them!

Your other major enemy is time, ticking away on the display at the side of the screen.

If you fail in your mission you are killed by the Matter Zapper that the Rigonites activate when they are bored with you.

The only way to avoid such a death is to kill all their craft before time disappears, literally, and you then go onto the next sheet.

Each sheet is made up of a number of blocked passages as well as a number of force fields that you must avoid at all costs.

They are deadly to you, but not to the enemy or to your photon missiles so don't be afraid to shoot through them.

Bonus points come from the spinning satellites, probably stolen from NASA's last flight, and you can have another chance to live if you get 10,000 points.

If you want to have any chance to get to this level I suggest you use a joystick mode. I couldn't get my fingers to be fast enough at all.

The sound and graphics are both good and I think it will be attractive to the more thoughtful arcade game player.

Dave Carlos

A little smasher!

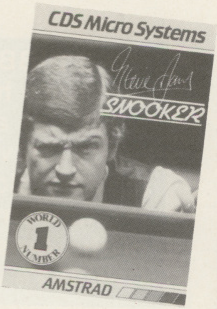
SNOOKER is easy. Watch the box any night, and apart from someone throwing a wobbly over an obviously simple shot which you or I could manage blindfold, there is nothing to it.

There again, snooker is darned difficult, as anyone who has peered down the ginormous length of a full sized table at the local Conservative or Working-men's Club – the equivalent of your Yankee pool hall – can bear witness to.

So you can bet your last piece of chalk that I approached CDS Micro Systems' **Snooker** with a great deal of respect – especially as it bore the magic name of snooker superstar Steve Davis.

And Steve has not snookered us. This is a little smasher.

For starters, you have full control of the white ball. Angle, speed, touch of top perhaps, or a searingly decelerating prod at the bottom of the cue ball – it's all there.



And the ensuing cannoning to and fro as the table takes the brunt of your onslaught is most realistic.

The speed control bears watching. I took no little time finding out how to judge the gentle kiss, the sturdy prod, the blazing "the hell with it" blast.

Mind you, it doesn't make it any easier than playing the real thing. I found laying a ruler on the screen helped to judge angles. But I would dearly have loved to play this game on a four foot flat screen monitor.

However these are quibbles about my incompetence. If I spent as much time playing Steve Davis Snooker as I did eating hot dinners I'd probably be showing H. Higgins Esq a thing or two on the Coral circuit.

As it is, I must try harder. This game deserves it. Anyone got a ruddy rubber ruler?

Peter Gee

Sheer fun and excitement!

I HOLD Interceptor Software personally responsible, and I'm going to tell the boss that in the morning . . .

I mean, cigarettes have to carry warnings, and so should Interceptor's **Message from Andromeda.** It should read something like, "This program can cause sleepless nights and subsequent loss of performance at work. Only to be played at weekends or holidays".

As it is, it's 4.30am and I'm exhausted. I'm also totally elated, as my wife will testify – my whoop of joy has just woken her up.

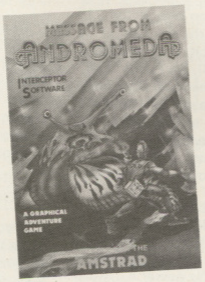
You see, I've just completed my mission, and I'm "over the moon, Jimmy".

Message from Andromeda is accurately described as a graphical adventure game – and the graphics are superb.

Captain of a space patrol cruiser, you receive a distress signal from a previously unknown planet. Your task is to investigate its source.

Emerging from your craft, you find yourself among buildings.

It goes without saying that one should make a map but you'll also find it useful to EXAMINE each object. And although you may be unaware of it, you're wearing a laser pistol.



The program understands the standard adventure commands: N, S, W, E, I, TAKE, DROP and so on. However, it's much more sophisticated than that, accepting commands such as "Shoot pistol at soldier".

Pretty soon you'll encounter your first stumbling block – the mirror room, which has a square plate beside a panel. Persevere and you'll succeed – the HELP command is useful here.

Once you've cracked that, the way leads through the armoury with its resident android, and downwards.

You're then in a typical underground adventure – note I said typical, not run of the mill.

Of course there's the obligatory twisty maze, this time in the form of a limestone cavern. It's well worth investigating this thoroughly – and you'll need all the objects you've found to do it.

The LOAD and SAVE features of the game come in handy around this time.

The game continues full of action right up to its dramatic climax. You'll meet a rather nasty slug, a ferocious scaled monster and the alien commander as you explore further. And I won't even hint at why the skeleton's there.

In the end you'll have to flee for your life before . . . no – I've told you enough already.

Let's hope I've managed to convey some of the excitement and sheer fun of the game.

Andrew Spencer

REVIEWED THIS MONTH

Astro Attack	Amsoft
Blagger	Alligata
Er*bert	Microbyte
Gems of Stradus	Kuma
Happy Letters	B.E.S.
Hunchback	Ocean
Map Rally	B.E.S.
Message from Andromeda	Interceptor
Snooker	CDS

There are two schools of thought on subtraction – do you borrow

MIKE BIBBY continues his series of articles aimed at lifting the veil of mystery cloaking the fundamentals of the Amstrad CPC464's workings

WE have seen that we can code our numbers in ways other than our usual denary, or decimal, system. We also looked last month at a way of coding known as the binary system, which uses the digits 0 to 1 to represent any number – unlike the denary system which uses the digits 0 to 9.

To distinguish the two systems, we decided to prefix binary numbers with the symbol "%".

The number "one hundred and sixty two" is encoded in each system as follows:

In denary,

162 i.e. 100+60+2

In binary,

```
   128 64 32 16 8 4 2 1
%    1  0  1  0 0 0 1 0
   i.e. 128+32+2
```

Each column in the binary system, known as a "bit", contains either a one or a zero.

Although the binary representation of a number is rather cumbersome to write, this simple two-state system is easily represented by electrical circuits – which are either ***on*** or ***off.***

We saw that the computer handles bits in groups of eight at a time. Such a group is called a byte. Thus a byte contains eight bits labelled, somewhat peversely, bits 0 to 7. (See Figure I.)

Bit 0, as you can see, is the "1" column. As this is the smallest value bit we say that bit 0 is the least significant bit (LSB). Bit 7, the "128" column, is called the most significant bit (MSB). The reason for using the numbers 0 to 7 to label the bits instead of the more logical 1 to 8 has to do with powers, a subject you almost certainly covered at school.

"2 to the power 2" is 2*2 = 4
"2 to the power 3" is 2*2*2 = 8
"2 to the power 4" is 2*2*2*2 = 16

and so on. "2 to the power 8" would be eight twos all multiplied together.

Notice as the powers of two increase – that is, as we multiply more twos together – the answers are doubling, just as our column or bit values do.

Also, 2 to the power of 2 is 4, the value of bit 2, while 2 to the power of 3 is 8, the value of bit 3. It shouldn't come as any surprise to you to find that 2 to the power of 7 is 128, the value of bit 7.

You can verify this on the Amstrad by using the symbol "↑" which stands for "to the power of". It shares a key with the £ sign.

Try:

```
PRINT 2↑4
PRINT 2↑7
```

Be sure to try 2↑1, which will show you why bit 1 has the value 2. Also try 2↑0. The answer may surprise you. The fact is that any number to the power 0 is 1! Hence bit zero has the column value of one. Figure II illustrates this.

Look at this sum:

```
    % 1
+ % 1
  %10
```

If you think about it, that is correct, since the sum adds one and one, and the answer %10 is binary for two. One way of relating this to our usual way of doing sums is to say that we carry when we get to two, instead of ten as we do in our normal, denary, sums.

Another way to look at it is that we have to carry when we get to two because we aren't allowed to use the digit '2'. If you remember, last month we had a rule forbiding two "coins" of the same value.

Try this sum:

```
      4 2 1
    % 1 1 in      3
+ % 1 0 denary  + 2
  % 1 0 1         5
```

Here we carry from the second

Bit number	7	6	5	4	3	2	1	0
	1	0	0	0	1	1	0	1
Bit value	128	64	32	16	8	4	2	1

Figure I: The bit pattern for 141

Bit number	7	6	5	4	3	2	1	0
Bit value	2↑7	2↑6	2↑5	2↑4	2↑3	2↑2	2↑1	2↑0
	128	64	32	16	8	4	2	1
	1	1	0	0	1	1	0	0

Figure II: The bit pattern for 204

w or just decompose?

column to the third.

Addition is not very hard at all – just make sure that you always "put 0 down and carry 1" when you get a two.

If you get a three then "carry one for two and put one down".

For example:

```
     8 4 2 1
  %    1 1 1  in       7
+ %      1 1  decimal +3
  % 1 0 1 0           10
```

Subtraction is a little more complicated, and depends on whether you borrow or decompose! The latter phrase doesn't describe the current economic climate, it's just that there are two schools of thought on the way subtraction should be taught – the borrowers and the decomposers.

Fortunately, we can ignore binary subtraction since we can manage without it – as does the microprocessor inside your machine. If you want to do some binary subtraction it is straightforward enough provided that you remember that it is two you're borrowing or taking, not ten. Figure III illustrates the process – without any attempt to explain it.

Before we leave the realm of simple sums, look what happens if we shift everything in a binary number over to the left, putting a zero into bit 0, which would be left vacant otherwise. For example:

```
  8 4 2 1
%   1 0 1    which is 5
```

becomes

```
  8 4 2 1
% 1 0 1 0    which is 10
```

This shifting to the left doubles the number automatically. This isn't too hard to visualise, because the value of each bit is transferred to the next higher bit, which is of course double in value – so the end result is that the whole number is doubled in value.

Similarly, we can do the binary equivalent of \2 by shifting to the right. For example:

```
  8 4 2 1
% 1 1 0 1    which is 13
```

becomes

```
  8 4 2 1
%   1 1 0    which is 6
```

and, of course, 13\2 gives you 6. The "\" operator – which we read as "div", deals with integer division. That is, it does division but only tells you the "wholes" and ignores the remainders.

As each bit is moved to the right, it occupies a column exactly one half lower in value, thus the sum total of all the bits is one half lower, save for the original bit 0 which has disappeared altogether (hence the ignored remainder).

Well, that's enough binary for one month. Next month it's hexadecimal.

```
 %   °1̸  ¹1̸°  ¹0̸          %  1  ¹1  ¹0̸                    6
-%         1    1   OR   -%  ¹  ¹1̸°  1     In decimal  -3
 %         1    1         %       1   1                  3

Decomposition             Borrowing
```

Figure III: Binary subtraction

ALAN McLACHLAN continues to prowl the whacky, whimsical world of the Amstrad CPC464

EVENING all. I hope you enjoyed the first issue of our new magazine. The features editor told me I can have a regular slot if I can stay awake long enough to write the articles. *(And the readers can stay awake long enough to read them. – Ed.)*

You've got to hand it to these computer technical bods, haven't you? They knew what they were doing when they launched the home micro on us unsuspecting 'inquisitive-but-very-soon-to-be-hooked' non computer persons.

I've lost more beauty sleep over the last few weeks than I've ever lost before – yes, even when the cat went missing for three days. I later found out she'd been on an all expenses paid binge with the ginger tom from next door – but that's another story.

These late nights, early mornings, call them what you will, are of course due to an inexplicable desire to get to grips with my Amstrad as soon as possible.

I told you last time that I was forever trying to use abbreviated commands that I'd been used to on the BBC Micro. Well you'll be pleased to know that to L. for LIST I now regulary add P. for PRINT.

I've no doubt that I'll soon get it right, but with switching from one machine to the other at regular intervals these errors occur quite naturally. It must be a problem that dogs all programmers who learn on a micro with a different Basic.

Even so, they can be quite infuriating and must add hours to what would normally be acceptable time spent programming.

At the moment I'm trying to translate some straightforward BBC Basic into Amstrad Basic. Now, as long as things remain simple, such as translating:

```
PRINT TAB(X,Y)"Hello"
```

to

```
LOCATE (X,Y):PRINT"Hello"
```

things go quite smoothly.

The FOR . . . NEXT loop is another expression which I seem to be able to handle without tearing my sparse hair out by its roots. But the conditional statement IF . . . THEN is a different problem altogether.

In BBC Basic the following THEN was optional. Its only use was cosmetic and assisted in making a program legible. Try leaving it out on the Amstrad and, except on very rare occasions such as when it precedes a GOTO, its omission will result in that blood curdling "syntax error" message.

I must admit I've chickened out and from now on all my IF statements will be followed by a THEN – it makes life so much easier.

And what about those WHILE . . . WENDS? These are completely new to me and to be honest I struggled for a while (sorry). They are in place of those very useful REPEAT . . . UNTIL loops on the Beeb.

For those that have not come across them – everything following the REPEAT is executed to the point where the condition following the UNTIL is met.

Translating them is not quite as straightforward as I first imagined. Some are simple enough though. Take this example on the BBC Micro:

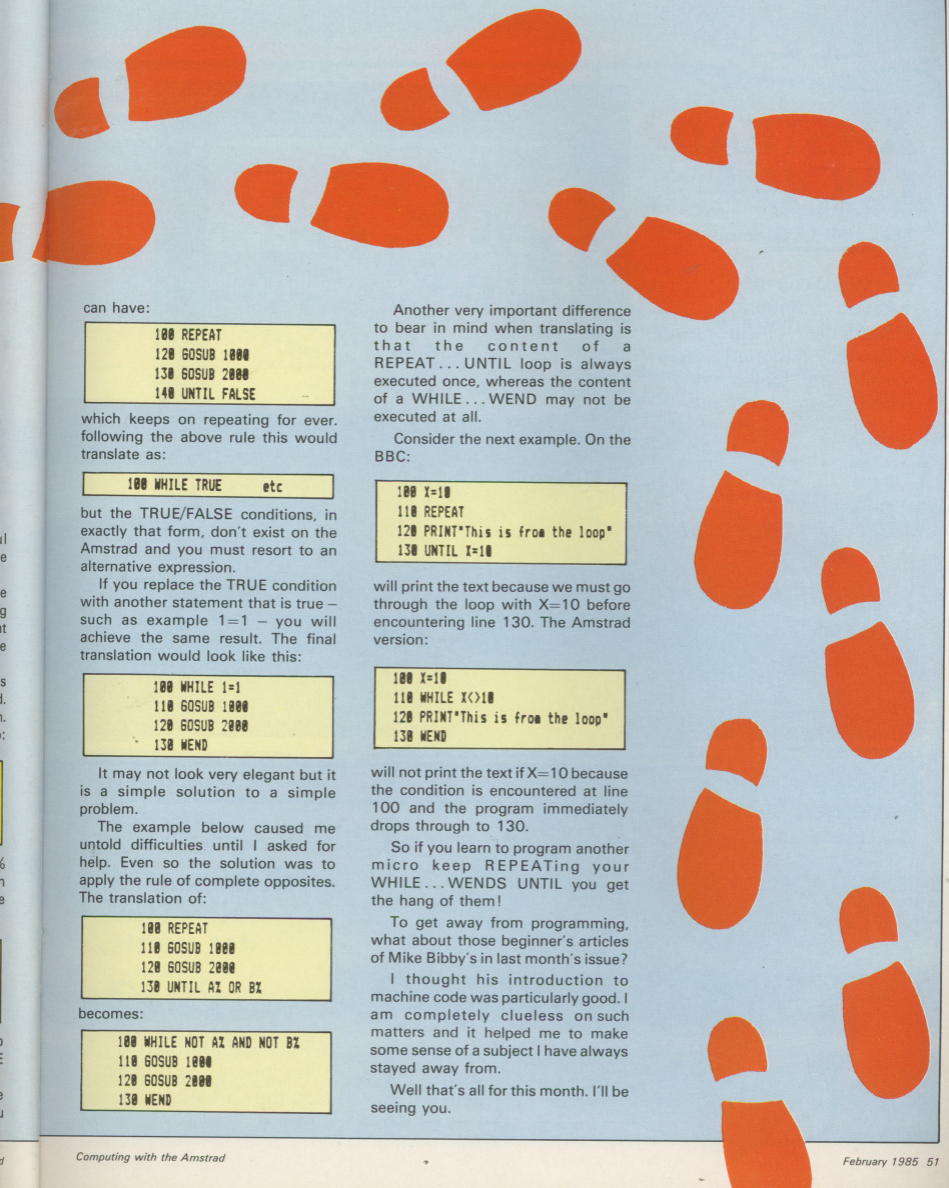
```
100 xX=0
110 REPEAT
120 xX=xX+1
130 UNTIL xX=20
```

This REPEATedly adds 1 to x% UNTIL x%=20 and then carries on with the rest of the program. On the Amstrad this translates easily as:

```
100 xX=0
120 WHILE xX<20
130 xX=xX+1
140 WEND
```

Instead of repeatedly adding 1 to x% until x%=20, you add it WHILE x%<20.

The next one is slightly more complicated. On the BBC Micro you



can have:

```
100 REPEAT
120 GOSUB 1000
130 GOSUB 2000
140 UNTIL FALSE
```

which keeps on repeating for ever. following the above rule this would translate as:

```
100 WHILE TRUE      etc
```

but the TRUE/FALSE conditions, in exactly that form, don't exist on the Amstrad and you must resort to an alternative expression.

If you replace the TRUE condition with another statement that is true – such as example 1=1 – you will achieve the same result. The final translation would look like this:

```
100 WHILE 1=1
110 GOSUB 1000
120 GOSUB 2000
130 WEND
```

It may not look very elegant but it is a simple solution to a simple problem.

The example below caused me untold difficulties until I asked for help. Even so the solution was to apply the rule of complete opposites. The translation of:

```
100 REPEAT
110 GOSUB 1000
120 GOSUB 2000
130 UNTIL A% OR B%
```

becomes:

```
100 WHILE NOT A% AND NOT B%
110 GOSUB 1000
120 GOSUB 2000
130 WEND
```

Another very important difference to bear in mind when translating is that the content of a REPEAT... UNTIL loop is always executed once, whereas the content of a WHILE... WEND may not be executed at all.

Consider the next example. On the BBC:

```
100 X=10
110 REPEAT
120 PRINT"This is from the loop"
130 UNTIL X=10
```

will print the text because we must go through the loop with X=10 before encountering line 130. The Amstrad version:

```
100 X=10
110 WHILE X<>10
120 PRINT"This is from the loop"
130 WEND
```

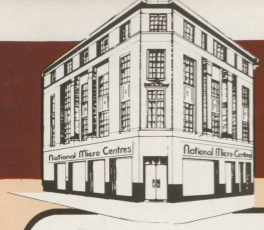
will not print the text if X=10 because the condition is encountered at line 100 and the program immediately drops through to 130.

So if you learn to program another micro keep REPEATing your WHILE... WENDS UNTIL you get the hang of them!

To get away from programming, what about those beginner's articles of Mike Bibby's in last month's issue?

I thought his introduction to machine code was particularly good. I am completely clueless on such matters and it helped me to make some sense of a subject I have always stayed away from.

Well that's all for this month. I'll be seeing you.



National Micro

AMSTRAD CPC464

The COMPLETE computer

Everything you need to start you computing immediately – includes a specially designed monitor and built-in data cassette recorder.

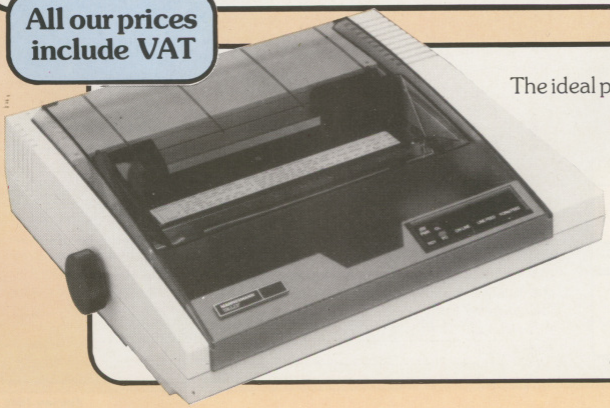
Also available

Printe lea

Powe mo

Fanta tic

All our prices include VAT



The ideal printer for the CPC464

Mannesmann Tally MT80 dot matrix

£217

Price includes lead for interfacing printer to the CPC464

Carriage £7

Centres

MAIL ORDER DIVISION

Phone your order on **061-429 8080**

Answering service outside normal office hours
Or use the order form below

lead for non-Amstrad printers . . . **£9.95**
Carriage FREE

modulator – links up the keyboard to your domestic colour TV set . . . **£29.95**
Carriage £3

tic top quality joystick only **£14.95**
Carriage £1

Blank data cassettes
Bargain offer –
11 for only **£5!**
Carriage 50p

Shoppers welcome at our retail stores:

National Micro Centre
36-38 St. Petersgate
Stockport SK1 1HL
Tel: 061-429 8080/8074

Wilmslow Micro Centre
62 Grove Street
Wilmslow, Cheshire
Tel: 0625 530891

Top 10

1. **Ghouls** (Micro Power) Rescue the jewels from Horrid Hall **£6.95**
2. **Manic Miner** (Software Projects) Aid Bill's search for the keys **£9.95**
3. **Dungeon Adventure** (Level 9) Seek the Demon Lord in his dungeon **£9.95**
4. **Galaxia** (Kuma) Defend yourself against the space attackers **£5.95**
5. **Hunchback** (Amsoft/BES) Rescue Esmerelda from the tower **£8.95**
6. **ER*BERT** (Microbyte) Paint the pyramid, avoid the nasties **£5.95**
7. **Happy Letters** (Amsoft/BES) Letter recognition for infants **£8.95**
8. **Timeman 1/2** (Amsoft/BES) Learn to tell and set the time **£8.95**
9. **Football Manager** (Addictive) Take your team to success **£7.95**
10. **Flightpath 737** (Anirog) Can you land the jet airliner **£6.95**

Carriage FREE

ORDER FORM

Post to:
NATIONAL MICRO CENTRES,
36 St. Petersgate,
Stockport SK1 1HL

Please supply the following:

Item	Qty	Total £	p

Attractive credit terms
Phone for details

Carriage .....
TOTAL .....

Please indicate method of payment:

☐ Cheque payable to National Micro Centres

☐ Access/Barclaycard No.

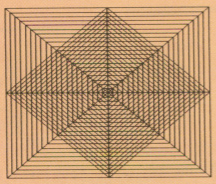
☐

Name .....
Address .....
.....
.....
Tel. No. .....
Signed ..... CA 2

Amstrad Analysis

By Trevor Roberts

THIS month Analysis takes a look at a short but impressive graphics program, exploring how the ORIGIN, MOVE and DRAW commands work.



```
10 REM Origins
20 MODE 1
30 ORIGIN 319,199
40 FOR unit=0 TO 190 STEP 10
50 GOSUB 100:REM draw square
60 GOSUB 170:REM draw diamond
70 NEXT unit
80 GOSUB 240:REM final lines
90 WHILE 1=1:WEND
100 REM makes square
110 MOVE -unit,unit
120 DRAW -unit,-unit
130 DRAW unit,-unit
140 DRAW unit,unit
150 DRAW -unit,unit
160 RETURN
170 REM makes diamonds
180 MOVE 0,unit
190 DRAW -unit,0
200 DRAW 0,-unit
210 DRAW unit,0
220 DRAW 0,unit
230 RETURN
240 REM draws final lines
250 MOVE -190,190
260 DRAW 190,-190
270 MOVE -190,-190
280 DRAW 190,190
290 MOVE 0,-190
300 DRAW 0,190
310 MOVE 190,0
320 DRAW -190,0
330 RETURN
```

Shifts the origin

Loop repeatedly calls subroutines

Eternal loop

Square drawing subroutine

Diamond subroutine

Axes Subroutine

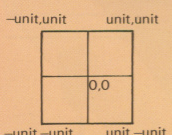


Figure I: Square coordinates

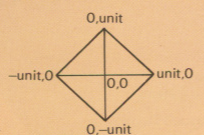


Figure II: Diamond coordinates

10 This holds the title of the program. Everything on the line that follows the REM is ignored by the micro.

20 Clears the screen and puts the Amstrad into Mode 1.

30 The ORIGIN command is used to reposition the graphics origin (the point that the DRAW and MOVE commands treat as 0,0). Normally this is at the bottom left of the screen, but here the ORIGIN command shifts it to the centre.

40-70 These lines form a FOR . . . NEXT loop with loop control variable unit. The STEP 10 means that *unit* varies from 0 to 190 in increments of 10 each time round the loop.

50 Calls the subroutine that draws the squares. Notice that each time round the loop the value of *unit* is different. Since the subroutine uses the value of *unit* to calculate where to draw the lines, this means that the subroutine will produce squares of differing sizes each time it is called.

60 Another subroutine call, this time one that has the Amstrad drawing diamonds. Again as *unit* has a different value each time round the loop, so the diamond drawn differs in size.

80 When the loop has finished repeating the program drops out of it and comes to the final subroutine call. This draws the connecting lines.

90 This WHILE . . . WEND loop is endless. Since 1=1 is always true, the loop just cycles over and over. It is included to stop "Ready" appearing on screen and spoiling the display.

100-160 These lines form the subroutine that draws each square. Figure I shows how the coordinates of the squares are related to each other in terms of the variable *unit* and the origin 0,0. As *unit* increases in value so the sides of the squares will increase in length. If you find that confusing, try working out the coordinates of the corners of the squares, as the value of *unit* goes from 0, to 10, to 20 and so on to 190.

170-230 This is the second subroutine. Its job is to draw diamonds. How big each diamond will be depends on the value of *unit* when the subroutine is called. Figure II shows the basic diamond coordinates.

240-330 This last subroutine is called when the FOR . . . NEXT loop has finished drawing all the squares and diamonds. It just draws in the final axes and diagonals.

Give your fingers a rest!

ALL Computing with the AMSTRAD **programs on one cassette for only £3.75**

Why tire your fingers typing when you can get all the programs from Computing with the Amstrad in one value-packed cassette?

Take a look at what's on offer and you'll see what we mean. We've games galore, useful utilities – plus lots of exciting extras . . .

These are the programs you'll get with this month's cassette:

- ***DIGGER:*** *Addictive arcade action. Can you guide our intrepid hero in his search for crystals and avoid the aliens?*
- ***SIMON:*** *Hours of fun with our Amstrad version of this simple children's game. You won't want to stop!*
- ***KINGDOM OF CRAAL:*** *A devious fantasy full of surprises, conjured up by the A team. Put your patience to the test and don't take it too lightly! This intriguing adventure will satisfy even the most seasoned adventurer.*
- ***TEXT EDITOR:*** *You'll never use a typewriter again once you've used this powerful, yet simple word processor.*
- ***REACTION TIMER:*** *Your reactions may not be what they used to be. Test them out on this simple little program.*
- ***ORIGINS:*** *Simple but effective, this demonstates the dramatic effects obtained by changing the graphics origin.*
- **PLUS all the example programs from our Sound and Graphics articles.**

You can still get last month's cassette. It contains:

- ***SMILEY:*** *Can you avoid the Grumpies as you guide Smiley round the labyrinth? Guarantees hours of fun.*
- ***CODE:*** *You don't have to be a mastermind to play this intriguing logic game – but it helps!*
- ***BINARY:*** *Baffled by binary bits? Let our utility help you out.*
- ***DANCER:*** *Simple but fun. You'll find our dancer is a lovely little mover.*
- ***TRAPPER:*** *You'll need quick-thinking, fast reactions and downright cunning to pen the monster of the maze.*
- ***SCROLLER:*** *Add that professional touch to your text with this slick sideways scrolling routine.*
- ***LETTER LITTER:*** *Keep your Amstrad tidy and learn the keyboard layout at the same time with this entertaining educational game.*
- **PLUS all the example programs from our Sound and Graphics articles.**

You've read the mag – now load the tapes!

HOW TO ORDER

Please send me the cassette of programs from *Computing with the Amstrad.*

Please send: ☐ This month's cassette ☐ Last month's cassette

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

☐ I wish to pay by: ☐ Access ☐ Visa

No. _____

Expiry date _____

Signed _____

Name _____

Address _____

_____

POST TO: Amstrad Tape Offer, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

Turn your Amstrad into an electronic typewriter with ROLAND WADDILOVE'S

TEXT EDITOR

HAVE you ever wished you could turn your Amstrad into a typewriter? With its superb editing commands – the electronic equivalent of Tippex – you'd be able to correct your typing errors with ease.

You could of course buy an all singing, all dancing word processor but they cost a fair bit. So we've come up with a text editor. It may not have a lot of frills, but it gets the job done and you don't need a PhD to understand it.

TextEd allows you to enter, print, load, save and manipulate text with a minimum of effort. It is designed to be easy to use, being menu driven with prompts where necessary.

It's also fairly robust and contains a reasonable amount of error checking and mug traps.

Most of the basic functions found in professional word processors have been implemented to increase the power of TextEd.

The only facility excluded is left and right justification – only left hand justification is available. This means that the right hand margin will be uneven. Justification is possible, but you will have to do this yourself.

When the program has been typed in and run a menu will be displayed on a Mode 2 screen.

There are 12 options, each listed with an associated reference character. Option selection is by pressing the key corresponding to this reference character – you don't need to press Enter after the selection.

The 12 options are:

E – Enters the edit mode. This allows the document in memory to be edited. It is also the option you want when starting to write a document. More of this later.

P – Will allow you to print your text. When selected, you will be asked to press the space bar when ready. This allows you to prepare the printer for printing. When space is pressed, the text is printed out exactly as it appears in the edit mode.

L – Load a previously saved text file. If you already have a document in memory you will be prompted with a message asking if you are sure you want to load a new text file.

This is done because the document in memory will be lost as soon as a new file is loaded. Press Y if you wish to load a new file, or N if you want to retain the file in memory – in which case the menu will be displayed again.

If you select Y you will be asked for the new file's name. Enter it, and press Play – making sure you have the correct tape in the tape recorder. Once the file has been loaded the menu will be displayed.

S – Save a text file. This saves the file in memory onto tape. If the file is too short – less than two lines – it cannot be saved, and the message, "not enough text" will appear. Otherwise you will be asked to enter the name of the file.

Now put a blank cassette in the tape deck and wind it to a suitable position. When you are ready to save the file, press a key. The text file will now be saved. The menu will reappear when the saving is complete.

N – New file. This wipes the text file from memory. Since the command is so destructive a safety feature has been added. As with the L command, you must confirm your option. Pressing Y will delete the text file, while N returns you to the menu.

Ø – This option exits from the program. Be careful you don't press Ø if you want the text file in memory. If you do, the text file is lost and the program ends.

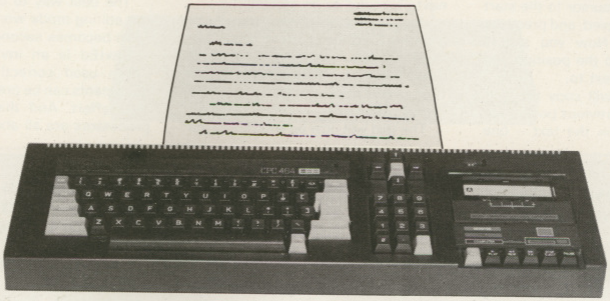
T – Set tab positions. This option allows you to set the four tab positions. You will be asked for the first, second, third and fourth column positions one after another. The tab position must be within the 71 column display – the column width will be less if a left margin has been set.

The tabbing function is invoked by pressing Tab while in the editing mode. This then moves the cursor to the next tab setting you have defined.

M – Set the left margin. This allows

VARIABLES

Option$	Option chosen from menu.
char$	Key pressed.
column%(4)	Tab positions.
start	Start of memory reserved for text.
here	Start of text.
address	Address of cursor.
finish	Address of last character.
last	Last available address for text.
length	Length of line.
insert%	Flag to show mode.
copy	Position of copy cursor.
a$	Start label.
b$	General variable.
i%,j%,k%,i	General variables.
x%	x coordinate of cursor.



you to define the number of characters for the left margin. When the document is viewed each line of text is moved over to the left by the number of characters selected.

This is useful for preparing documents to be punched and filed. Normally the punch holes would destroy part of the printout. This option offers a simple remedy.

R – Re-defines the colours. This allows you to change the colour of the paper, pen and border. If you want to change the pen colour for example, press P. On pressing the key, the colour of the pen is automatically changed.

This colour is the next one available in the palette. The O and B keys change the colour of the paper and border, respectively. Press Enter when you are happy with the colours selected.

C – Send codes to the printer. Some printers allow you to select different character sets, or different printing styles. This option can be used to send the necessary codes to the printer to select them. You will be asked for the character number to be sent to the printer – the Ascii number.

Since most codes consist of several numbers, you will be repeatedly asked for numbers until you enter –1. This is used to terminate the list.

One point to note is that the Amstrad only allows you to send the numbers 0 to 127 to the printer – since only seven bits are used to transfer data to the printer.

H – Help page. This gives information on the keys used in the editing mode – very useful for those with only short term memories.

D – Define a function key. This allows you to program the number keys on the numeric pad – 1 to 9. You will be asked for the key number to be programmed and the string you wish to assign to the key.

This allows the keys to be programmed with words frequently used throughout the text – saving time and wear and tear on the keyboard.

The most important command of the 12 is the Edit command. When this has been selected the editing mode is invoked. Now you can type in your document or edit an existing one. The start of the text is denoted by the message ** start ** – this allows you to keep track of your position in the document.

On the screen you will see a flashing cursor. Moving this, by using the cursor arrows, allows you to position the cursor at the point in the text where you wish to begin editing.

When you have moved to a suitable position in the text you can start typing. You will notice that it is much the same as typing in a program except this time you type in words instead of Basic commands.

The screen acts like a small window. This window can be moved up and down the text by using the cursor up and down keys – allowing you to view and edit any section of the text no matter how large it is.

If the text size were limited to one screen of text, only small documents could be prepared – making the utility useless to people who need large text files.

There are two editing modes, write and insert – toggle selection by pressing CLR. The bottom left corner of the screen shows the current editing mode.

In write mode everything you type is printed at the cursor position. If text is already there it will be overwritten.

Insert mode will create space for the new character entered. This is done by shuffling the existing text down memory, so making space for the new character.

Any word that splits over two lines will look much neater if it is put onto the next line. This can be done by moving the cursor to the start of the split word and pressing the large Enter key – this must be done in the insert mode. The split word will now be dragged onto the next line.

Other significant keys in the editing mode are listed below. This list can also be obtained by selecting option H from the menu:

- The *cursor* can be moved to the next tab position by pressing the TAB key.
- The *large Enter key* moves the cursor to the start of the next line.
- Pressing *Delete* causes the character to the left of the cursor to be deleted – the cursor is moved to the previous position of the deleted character as well.
- The *small Enter key* is used to set a ▶

marker position when copying blocks of text.

Copying text is quite easy. First move the flashing cursor to the start of the text to be copied, and press the small Enter key. Now you should move the cursor to the position you wish to copy the text to.

Pressing Copy will copy the text from the cursor's previous position to the new position in the text – one character at a time. This is very similar to editing a Basic program, except there is no second cursor showing the next character to be copied.

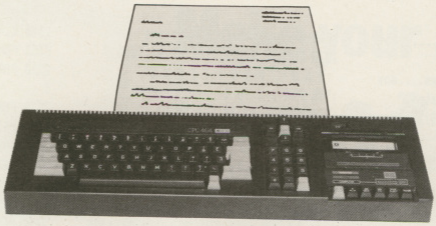
Justification requires extra spaces between words to pad out each line, resulting in a tidy right hand edge. Move the cursor to the position where the pad is required and press the space bar – while in insert mode.

The text will now be shuffled to the left by one character. The pads should be inserted until the required result has been achieved.

The best way to understand how the editing mode works is to try it. It soon becomes second nature.

TextEd is an invaluable utility. When used correctly, high quality documents can be prepared with very little effort. And that's what word processors are all about.

```
10 REM ***** TextEd *****
20 REM *By R.A.Waddilove*
30 MEMORY &3FFF
40 MODE 2
50 ON BREAK GOSUB 400 :REM do nothing
60 GOSUB 240 :REM initialise
70 WHILE option$<>"0"
80 GOSUB 1390 :REM menu
90 IF option$="E" THEN GOSUB 420 :REM
 input
100 IF option$="T" THEN GOSUB 1960 :R
EM set tabs
110 IF option$="M" THEN GOSUB 2060 :R
EM set margin
120 IF option$="N" THEN GOSUB 1900 :R
EM new file
130 IF option$="L" THEN GOSUB 1770 :R
EM load
140 IF option$="S" THEN GOSUB 1660 :R
EM save
150 IF option$="R" THEN GOSUB 2150 :R
EM change colours
160 IF option$="H" THEN GOSUB 1530 :R
EM help
170 IF option$="P" THEN GOSUB 2300 :R
EM print
180 IF option$="C" THEN GOSUB 2440 :R
EM printer codes
190 IF option$="D" THEN GOSUB 2560 :R
EM function keys
200 WEND
210 MODE 2
220 END
230 :
240 REM ***** initialise *****
250 SPEED WRITE 1
260 FOR i%=0 TO 120:READ a$:POKE &AB0
0+i%,VAL("&"+a$)::NEXT
270 DATA DD,5E,00,DD,56,01,D5,DD,6E,0
2,DD,66,03,E5,A7,ED,52,44,4D,E1,54,5D
,03,13,ED,B8,CD,06,B9,E1,DD,7E,04,77,
E5,DD,66,08,2E,10,CD,75,BB,E1,DD,46,0
6,7E,E5,C5,CD,D9,BD,C1,E1,23,10,F5,C9
280 DATA DD,5E,00,DD,56,01,D5,DD,6E,0
2,DD,66,03,A7,ED,52,44,4D,03,62,6B,1B
,ED,B0,CD,06,B9,C3,23,AB
290 DATA CD,06,B9,DD,6E,00,26,01,DD,5
E,02,DD,56,03,D5,C3,28,AB
300 DATA 21,00,40,11,01,40,01,FF,6A,3
6,20,ED,B0,C9
310 MOVE 0,75:DRAW 0,398,1:DRAW 639,3
98:DRAW 639,75:DRAW 0,75:DRAW 0,0:DRA
W 639,0:DRAW 639,75
320 DIM column%(4)
330 INK 0,13:INK 1,0:BORDER 10:CALL &
AB6B
340 start=&4000:a$="** START **"
350 FOR i%=1 TO LEN(a$):POKE start+i%
-1,ASC(MID$(a$,i%,1)):NEXT
360 WINDOW #1,2,79,22,24:WINDOW #2,2,
79,2,20:WINDOW 5,75,2,20
370 length=71:here=start+80:finish=he
re:last=43775-length:copy=here
380 column%(1)=10:column%(2)=20:colum
n%(3)=30:column%(4)=40
390 option$="":KEY 139,""
400 RETURN
410 :
420 REM ***** input text *****
430 address=here:copy=here:row%=1:x%=
1:insert%=0
440 LOCATE #1,3,1:PRINT #1,"Mode: wri
te";TAB(35);"Line:";TAB(60);"Bytes fr
ee:":LOCATE #1,3,3:PRINT #1,"CLR chan
ges mode";TAB(33);"column:";TAB(60);"
CTRL for menu":GOSUB 980
450 LOCATE 1,15:PRINT a$
460 CALL &AB59,length*3,0,here,16:CAL
L &AB59,length-1,0,here+length*(row%+
2),19
470 WHILE INKEY(23)=-1:char$=INKEY$
480 LOCATE x%,16:PRINT CHR$(PEEK(addr
ess));
490 IF char$)=CHR$(&F0) THEN GOSUB 98
0
500 IF char$)CHR$(31) AND char$(CHR$(
127) THEN GOSUB 710
510 IF char$=CHR$(13) THEN GOSUB 1130
520 IF char$=CHR$(&7F) THEN GOSUB 790
530 IF char$=CHR$(9) THEN GOSUB 1270
540 IF char$=CHR$(16) THEN GOSUB 1350
550 IF char$=CHR$(&E0) THEN GOSUB 620
560 IF INKEY(6)>-1 THEN copy=address:
SOUND 1,100,5
570 LOCATE x%,16:PRINT CHR$(95);
580 WEND
590 RETURN
600 :
610 REM ***** copy *****
620 IF finish<address THEN finish=add
ress
630 WHILE char$=CHR$(&E0) AND finish<
last
640 char$=CHR$(PEEK(copy)):GOSUB 1210
:char$=INKEY$
650 copy=copy+1-(copy>address AND ins
ert%)
660 WEND
670 GOSUB 980
680 RETURN
690 :
700 REM ***** print *****
710 IF finish<address THEN finish=add
ress
720 WHILE char$)CHR$(31) AND char$<CH
R$(127) AND finish<last
730 GOSUB 1210:char$=INKEY$
740 WEND
750 GOSUB 980
760 RETURN
770 :
780 REM ***** delete *****
790 IF finish<address THEN RETURN
800 WHILE char$=CHR$(&7F) AND address
>here
810 IF insert% THEN CALL &AB3B,x%,len
gth,0,finish,address:finish=finish-1
ELSE POKE address,32:LOCATE x%,16:PRI
NT * *:IF finish=address THEN finish=
finish-1
820 x%=x%-1:address=address-1
830 IF x%=0 THEN x%=length:GOSUB 1090
840 char$=INKEY$
850 WEND
860 GOSUB 980
```

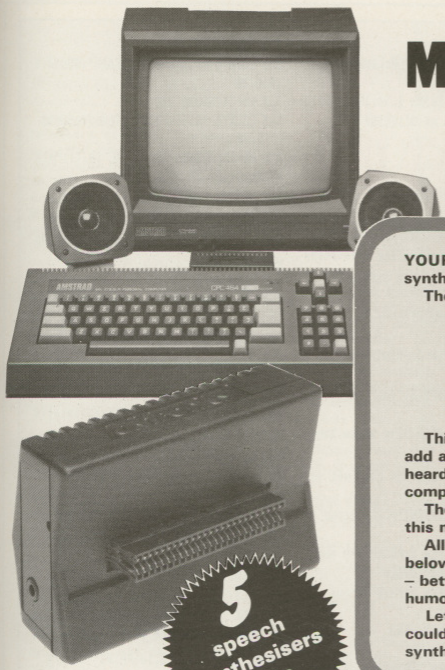


```
870 RETURN
880 :
890 REM ***** move cursor *****
900 IF INKEY(2)>-1 AND address+length
<last THEN address=address+length:GOS
UB 1050
910 IF INKEY(0)>-1 AND address-length
>=here THEN address=address-length:GO
SUB 1090
920 IF INKEY(8)>-1 AND address>here T
HEN x%=x%-1:address=address-1:IF x%=0
 THEN x%=length:GOSUB 1090
930 IF INKEY(1)>-1 AND address+1<last
 THEN x%=x%+1:address=address+1:IF x%
>length THEN x%=1:GOSUB 1050
940 LOCATE #1,40,1:PRINT #1,row%;"  "
:LOCATE #1,40,3:PRINT #1,x%
950 RETURN
960 :
970 REM ***** status *****
980 LOCATE #1,40,1:PRINT #1,row%;"  "
:LOCATE #1,71,1:PRINT #1,last-finish;
" ":LOCATE #1,40,3:PRINT #1,x%
990 IF NOT insert% THEN RETURN
1000 IF here+length+(row%+2)<last THE
N CALL &AB59,3+length,0,here+length+r
ow%,17:RETURN
1010 IF here+length+(row%+1)<last THE
N CALL &AB59,2+length,0,here+length+r
ow%,17:RETURN
1020 IF here+length+row%<last THEN CA
LL &AB59,length,0,here+length+row%,17
:RETURN
1030 :
1040 REM ***** scroll up *****
1050 row%=row%+1:IF here+length+(row%
+2)>last THEN LOCATE 1,20:PRINT ELSE
CALL &AB59,length,0,here+length+(row%
+2),20
1060 RETURN
1070 :
1080 REM ***** scroll down *****
1090 row%=row%-1:PRINT CHR$(30);CHR$(
11):IF row%>15 THEN CALL &AB59,length
,0,here+length+(row%-16),1 ELSE IF ro
w%=15 THEN PRINT CHR$(30);a$
1100 RETURN
1110 :
1120 REM ***** cr *****
1130 char$=" "
1140 IF finish<address THEN finish=ad
dress
1150 GOSUB 1210
1160 IF x%<>1 AND finish<last THEN 11
50
1170 GOSUB 980
1180 RETURN
1190 :
1200 REM ***** insert *****
1210 IF insert% THEN CALL &AB00,x%,le
ngth,ASC(char$),finish,address:finish
=finish+1 ELSE POKE address,ASC(char$
):LOCATE x%,16:PRINT char$:IF finish=
address THEN finish=finish+1
1220 address=address+1:x%=x%+1
1230 IF x%>length THEN x%=1:GOSUB 105
0
1240 RETURN
1250 :
1260 REM ***** tab *****
1270 char$=" "
1280 IF finish<address THEN finish=ad
dress
1290 GOSUB 1210
1300 IF (x%<>column%(1) AND x%<>colum
n%(2) AND x%<>column%(3) AND x%<>colu
mn%(4)) AND finish<last AND x%<>1 THE
N 1290
1310 GOSUB 980
1320 RETURN
1330 :
1340 REM ***** insert/write toggle **
***
1350 insert%=NOT insert%
1360 LOCATE #1,9,1:IF insert% THEN PR
INT #1,"insert" ELSE PRINT #1,"write
"
1370 RETURN
1380 :
1390 REM ***** menu *****
1400 CLS #1:CLS #2:ZONE 39
1410 LOCATE #2,35,2:PRINT #2,"M E N U
"
1420 LOCATE #2,1,5:PRINT #2,SPC(5)"E.
  Enter and edit text.","  T.  Set TAB
 positions.":PRINT #2:PRINT #2,SPC(5)
"P.  Print text.","  M.  Set the margi
n."
1430 PRINT #2:PRINT #2,SPC(5)"L.  Loa
d a text file.","  R.  Redefine colour
s.":PRINT #2:PRINT #2,SPC(5)"S.  Save
 a text file.","  C.  Control codes to
 printer."
1440 PRINT #2:PRINT #2,SPC(5)"N.  New
 file.","  H.  Help page.":PRINT #2:PR
INT #2,SPC(5)"@.  End program.","  D.
 Define function key."
1450 PRINT #1:PRINT #1,TAB(29);"< Pres
s a key >"
1460 WHILE INKEY$<>"":WEND:option$=""
1470 WHILE INSTR(" DETPMLRSCNH@",opti
on$)<2
1480 option$=UPPER$(INKEY$)
1490 WEND
1500 CLS #1:CLS #2
1510 RETURN
1520 :
1530 REM ***** help *****
1540 LOCATE #2,35,1:PRINT #2,"H E L P
"
1550 LOCATE #2,1,4:PRINT #2,"CLR.....
..change mode. Write mode overwrites
whatever is already there, insert
      mode will create space by movin
g everything along."
1560 PRINT #2:PRINT #2,"ENTER.....sma
ll key. Note cursor position, when co
py is pressed text is copied
 from here."
1570 PRINT #2:PRINT #2,"ENTER.....lar
ge key. Move to start of next line."
1580 PRINT #2:PRINT #2,"DEL.......del
ete."
1590 PRINT #2:PRINT #2,"TAB.......mov
e to next tab position."
```

```
1600 PRINT #2:PRINT #2,"CTRL.....ret
urn to menu."
1610 PRINT #2:PRINT #2,"Arrows....mov
e cursor in directions indicated."
1620 PRINT #1:PRINT #1,TAB(29);"( Pre
ss a key )"
1630 WHILE INKEY$="":WEND
1640 RETURN
1650 :
1660 REM ***** save *****
1670 LOCATE 1,2:PRINT "S A V E    T E
 X T    F I L E":PRINT
1680 IF finish-here<length THEN PRINT
:PRINT "Not enough text in the file !
":FOR i%=1 TO 5000:NEXT:RETURN
1690 PRINT:PRINT:LINE INPUT "What is
the name ";name$:PRINT
1700 i=finish-here
1710 SAVE name$,b,here,i
1720 OPENOUT "!"
1730 PRINT #9,length,finish
1740 CLOSEOUT
1750 RETURN
1760 :
1770 REM ***** load *****
1780 LOCATE 1,2:PRINT "L O A D    T E
 X T    F I L E":PRINT
1790 IF finish-here>length THEN LOCAT
E 1,5:PRINT "Loading a new file will
destroy the one at present in the mem
ory.":PRINT:PRINT "Is this ok ? ( Y o
r N ) ":char$="":WHILE char$="":char$
=INKEY$:WEND:IF UPPER$(char$)<>"Y" TH
EN RETURN
1800 CLS #1:CALL &AB6B
1810 PRINT:PRINT:LINE INPUT "What is
the name ";name$:PRINT
1820 OPENOUT name$:CLOSEOUT
1830 LOAD name$,here
1840 OPENIN "!"
1850 INPUT #9,length,finish
1860 CLOSEIN
1870 i%=(80-length)\2:WINDOW i%,i%+le
ngth-1,2,20
1880 RETURN
1890 :
1900 REM ***** new file *****
1910 LOCATE #2,30,3:PRINT #2,"N E W
  F I L E"
1920 LOCATE #2,13,7:PRINT #2,"You wil
l destroy the file at present in the
memory.":LOCATE #2,26,10:PRINT #2,"Is
 this ok ? ( Y or N )":char$="":WHILE
 char$="":char$=INKEY$:WEND:IF UPPER$
(char$)<>"Y" THEN RETURN
1930 finish=here:CALL &AB6B
1940 RETURN
1950 :
1960 REM ***** set tabs *****
1970 LOCATE #2,22,1:PRINT #2,"S E T
  T A B    P O S I T I O N S"
1980 FOR i=1 TO 4
1990 LOCATE #2,30,i+4:PRINT #2,"TAB";
i;".....";SPC(10);STRING$(10,CHR$(8))
;
2000 INPUT #2,b$
2010 IF b$="-" THEN column%(i)=0 ELSE
 column%(i)=VAL(b$)
2020 IF column%(i)<2 OR column%(i)>75
 THEN 1990
2030 NEXT
2040 RETURN
2050 :
2060 REM ***** set margin *****
2070 LOCATE #2,26,2:PRINT #2,"S E T
  M A R G I N"
2080 LOCATE #2,15,10:PRINT #2,"How ma
ny characters wide is the margin ";SP
C(10);STRING$(10,CHR$(8));
2090 INPUT #2,b$
2100 IF b$="-" THEN i%=0 ELSE i%=VAL(
b$)
2110 IF i%<2 OR i%>30 THEN 2080
2120 WINDOW i%,79-i%,2,20:length=80-2
*i%
2130 RETURN
2140 :
2150 REM ***** change colours *****
2160 i%=0:j%=0:k%=0
2170 LOCATE #2,25,2:PRINT #2,"R E D E
  F I N E    C O L O U R S"
2180 WHILE INKEY(18)=-1
2190 LOCATE #2,25,8:PRINT #2,"P.....c
hange pen colour"
2200 LOCATE #2,25,10:PRINT #2,"O.....
change paper colour"
2210 LOCATE #2,25,12:PRINT #2,"B.....
change border colour"
2220 LOCATE #1,25,2:PRINT #1,"Press
ENTER  when finished."
2230 IF INKEY(27))-1 THEN i%=(i%+1)MO
D 27:INK 1,i%
2240 IF INKEY(34))-1 THEN j%=(j%+1)MO
D 27:INK 0,j%
2250 IF INKEY(54))-1 THEN k%=(k%+1)MO
D 27:BORDER k%
2260 WEND
2270 WHILE INKEY$<>"":WEND
2280 RETURN
2290 :
2300 REM ***** print *****
2310 LOCATE #1,25,2:PRINT #1,"Press t
he  SPACE BAR  to print."
2320 WHILE INKEY$<>" ":WEND:CLS
2330 LOCATE #1,25,2:PRINT #1,"     P
R I N T I N G              "
2340 k%=(81-length)\2
2350 FOR i=here TO finish STEP length
2360 FOR j%=1 TO k%:PRINT #8," ";:NEX
T
2370 FOR j%=0 TO length-1
2380 PRINT CHR$(PEEK(i+j%));:PRINT #8
,CHR$(PEEK(i+j%));
2390 NEXT
2400 PRINT #8
2410 NEXT
2420 RETURN
2430 :
2440 REM ***** send codes to printer
*****
2450 LOCATE #2,20,2:PRINT #2,"P R I N
 T E R    C O N T R O L    C O D E S"
2460 LOCATE #2,21,8:PRINT #2,"Type in
 the ASCII codes one at a time....."
2470 LOCATE #2,25,14:PRINT #2,"Type
-1  when you have finished."
2480 i%=0
2490 WHILE i%>-1
2500 PRINT #1,SPC(20):INPUT #1,;"Code
 ";b$
2510 IF b$="-" THEN i%=1000 ELSE i%=V
AL(b$)
2520 IF i%>-1 AND i%<128 THEN PRINT #
8,CHR$(i%);:PRINT #1,"  character sen
t" ELSE PRINT #1,"  <-- ignored"
2530 WEND
2540 RETURN
2550 :
2560 REM ***** define function keys *
****
2570 LOCATE #2,20,2:PRINT #2,"D E F I
 N E  A  F U N C T I O N    K E Y"
2580 LOCATE #2,25,8:PRINT #2,"Which f
unction key ";SPC(10);STRING$(10,CHR$
(8));
2590 INPUT #2,b$
2600 IF b$<"1" OR b$>"9" THEN 2580 EL
SE i%=128+VAL(b$)
2610 LOCATE #2,25,12:PRINT #2,"What i
s the string...";:LINE INPUT #2,b$
2620 KEY i%,b$
2630 RETURN
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 55.



Make your Amstrad TALK!

YOUR Amstrad can talk with the dk'tronics speech synthesiser – the latest add-on for the CPC464.

The synthesiser has:

- **Almost unlimited vocabulary.**
- **Easy to use commands – it accepts normal English.**
- **Built-in stereo amplifier.**
- **Complete program independence – while talking it won't affect the running of your program.**

This unique, easy-to-use speech synthesiser will add a new dimension to your computing. You've heard of computer feedback – now listen to computer talkback.

There are five speech synthesisers to be won in this month's easy-to-enter competition.

All you have to do is send us on the coupon below an imagined dialogue – of at most 150 words – between your Amstrad and yourself. The most humorous and entertaining conversations win.

Let's have your entries by February 28, and you could be the proud owner of a dk'tronics speech synthesiser.

5 speech synthesisers to be WON!

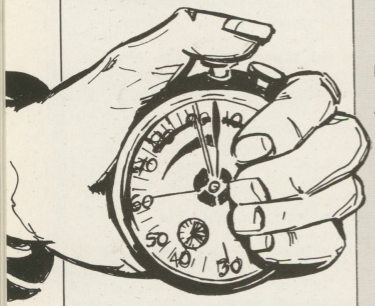
Name _____

Address _____

_____

Send your entry to: Synthesiser contest, *Computing with the Amstrad,* Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY. **Closing date: February 28, 1985.**

On your marks... get set... GO!



Test your reactions with ALAN McLACHLAN

FIND out how fast you are with our Amstrad reaction timer. Are you quicker off the mark than your friends? Does your reaction time vary as the day goes on? Kids, are you faster than your parents? Type in this listing and find out.

When you run the program a white bar will appear on the left of the screen. Watch it carefully because soon it's going to change colour, beeping every time to give you a warning.

At first it becomes red. Then, like a set of traffic lights, it will change to yellow. This tells you to be prepared because green is the next colour and you're going to have to react quickly.

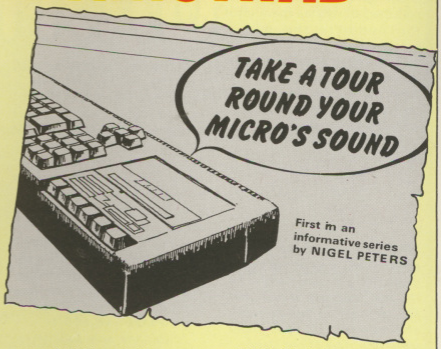
As soon as it turns to green (but not before!) you have to press the space bar. The Amstrad will then tell you how fast your reactions are and what it thinks of them.

```
10 REM reaction timer
20 REM adapted by Alan McLachlan
25 BORDER 0
30 MODE 1:INK 0,1:INK 1,24:INK 2,20:I
NK 3,6:PEN 1:DIM m$(8):GOSUB 100
40 GOSUB 340
50 GOSUB 390
60 GOSUB 550
70 WHILE INKEY$="":WEND
80 RUN
90 END
100 REM******** initialise **********
110 CLS:LOCATE 14,4:PEN 1:PRINT"REACT
ION TESTER"
120 LOCATE 14,5:PRINT"---------------
"
130 LOCATE 9,8:PEN 2:PRINT"Use this p
rogram to test"
140 LOCATE 14,10:PRINT"your reactions
."
150 LOCATE 5,12:PRINT"The white bar w
hich will appear "
160 LOCATE 8,14:PRINT"on the left of
the screen"
170 LOCATE 12,16:PRINT"will change co
lour."
180 LOCATE 4,20:PEN 3:PRINT"The seque
nce is RED, YELLOW, GREEN"
185 LOCATE 8,23:PEN 2:PRINT"Press any
 key to continue"
186 WHILE INKEY$="":WEND
187 CLS
190 LOCATE 7,6:PEN 2:PRINT"Each time
the colour changes"
200 LOCATE 10,8:PRINT"you will hear a
 beep."
210 LOCATE 11,12:PEN 1:PRINT"Press th
e space bar"
220 LOCATE 2,14:PRINT"as soon as the
colour changes to green"
230 LOCATE 12,16:PEN 3:PRINT"AT THE T
HIRD BEEP"
240 LOCATE 4,18:PRINT"Your time and r
ating will then be given"
250 LOCATE 8,23:PEN 2:PRINT"PRESS ANY
 KEY TO START TEST"
260 WHILE INKEY$="":WEND
270 CLS
275 INK 1,26:PEN 1
280 A$=STRING$(3,143)
290 PRINT:PRINT:FOR I%= 1 TO 18:PRINT
 TAB(8) A$:NEXT
300 B$=STRING$(3,32)
310 FOR I= 0 TO 7:READ M$(I):NEXT
320 I%=0:RANDOMIZE TIME
330 RETURN
340 REM*********WAIT***************
350 N%=6:GOSUB 480
355 LOCATE 18,11:PRINT"READY":SOUND 1
,100,20
360 N%=24:GOSUB 480
365 LOCATE 18,11:PRINT"STEADY":SOUND
1,75,20
370 N%=18:GOSUB 480:LOCATE 18,11:PRIN
T SPACE$(6):SOUND 1,50,50
380 RETURN
390 REM*************TEST*************
400 IF INKEY$ <> "" THEN GOSUB 510:RU
N
420 TIMNOW=TIME
430 FOR I%= 1 TO 20
435 IF I% >19 OR INKEY(47)=0 THEN 460
440 LOCATE 8,I%:PRINT B$:NEXT
460 T%=INT(TIME-TIMNOW)/3
470 RETURN
480 REM**********colour**********
490 FOR I=1 TO 1000+INT(1000*RND)+1:N
EXT:INK 1,N%
500 RETURN
510 REM********too soon*********
520 CLS:INK 1,24
530 LOCATE 1,10:PRINT"WAIT FOR THE GR
EEN LIGHT!!!":FOR I= 1 TO 1000:NEXT
540 RETURN
550 REM***********MESSAGE**********
560 CLS:INK 1,24:LOCATE 17,10:PRINT"Y
OU TOOK ":LOCATE 7,13:PRINT T%; " hun
dredths of a second":FOR DELAY = 1 TO
 500:NEXT:LOCATE 8,16:PRINT"You are "
;:PEN 2:PRINT M$(INT(T%/5))
565 IF INT(T%/5)=7 THEN FOR LOOP=1 TO
 20:SOUND 1,20,3:SOUND 1,70,3:NEXT
570 FOR x= 1 TO 3000:NEXT
580 LOCATE 8,23:INK 3,26,1:PEN 3:SPEE
D INK 50,20:PRINT"PRESS ANY KEY TO TR
Y AGAIN"
585 WHILE INKEY$="":WEND
590 RETURN
600 DATA OUT OF THIS WORLD,TOO GOOD T
O BE TRUE,EXCELLENT,VERY GOOD,GOOD,PO
OR,NEARLY ASLEEP,ASLEEP....WAKE UP!!
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

ALL MOD CONS ON THE AMSTRAD



DO MOD and DIV exist on the Amstrad?

I'm darned if I can find a reference to them in that unsatisfactory tome that comes with the machine. And do they work as normal? – **Robert Chitty, Swindon.**

■ Well MOD exists, and works in the standard way: giving the *remainder* of the division of two numbers. In other words, 7 MOD 3 would leave you with 1, the remainder.

There is the equivalent of DIV but the Amstrad uses the / symbol which you'll find directly above the control key. It works in exactly the same way: it gives the whole number of the division's answer, ignoring any fractions. So 7 / 3 would give you 2.

Bugs are pulling strings

HAS anyone else noticed this bug on their Amstrad:

Suppose you're in Mode 1 and you've got two large strings such as:

```
A$ = "abcdefghijklmnopqrstuvw"
B$ = "ABCDEFGHIJKLMNOPQRSTUVW"
```

then:

```
PRINT A$;:PRINT B$
```

will put each string on a separate line, and not glue the two together as decent Basics should.

The bug occurs when the two strings together take up more than the width of the screen, so watch it when you're manipulating string variables.

The solution is to use:

```
PRINT A$ + B$
```

– John Hayes, Torquay.

Hex on numbers

I'M a bit confused about the way the Amstrad handles hexadecimal numbers.

I was trying to do some quick conversions to denary – and succeeding – using:

```
PRINT &2A
```

and such like.

However when I got to big numbers things went berserk. I kept getting negative numbers. What's going on? – **Eileen Neul, Grimsby.**

■ This is because the Amstrad uses signed integers. This just means that two bytes are used to store each integer in such a way that it can handle negative as well as positive integers. You'll find things work normally up to &7FFF but:

```
PRINT &8000
```

will give you >32768.

```
PRINT &8001
```

will give you >32767 and so on down to:

```
PRINT &FFFF
```

which gives you >1.

If it's still a little unclear, don't worry. Our Bits and Bytes feature will be discussing it in detail in a future issue.

Turn it up!

MAY I compliment Nigel Peters on the beginning of his series on the SOUND command. When I first came across this command in the user instructions, I was appalled, but having read the article SOUND doesn't seem too frightening.

However, now that I've said something nice about him, can I raise one small point? He didn't mention the volume control on the right of the micro.

I spent 20 minutes wondering why nothing worked until it dawned on me that I'd got it turned down low. I wonder how many others wasted their time like this? – **A. Smith, Exeter.**

■ Nigel is holding his head in shame, Mr Smith. However, to be fair to him it's not often that he underestimates the Wally factor.

LD A cuts no ice

I'VE been a bit over ambitious I think. I've been typing in some assembly programs but the Amstrad doesn't seem to like it, throwing out lines like:

```
LD A, 42
```

even though I've typed them in exactly as the book shows.

What's going wrong? – **Harold Richards, Richmond.**

■ You'd probably do well reading Mike Bibby's articles on machine code which started last month. You'll then discover that the machine only understands numbers. The LD A is known as a mnemonic.

We use mnemonics such as LD A and RET in place of the numbers the micro uses. After all, LD A is a lot easier for us humans to remember than &3E – particularly if we know it stands for LoaD A register.

However, even if we understand what LD A means, the

Computing with the AMSTRAD Postbag

WE welcome letters from readers – about your experiences using the CPC464, about tips you would like to pass on to other users . . . and about what you would like to see in future issues.

The address to write to is:

Postbag Editor
Computing with the Amstrad
Europa House
68 Chester Road
Hazel Grove
Stockport SK7 5NY

micro still insists on its &3E.

This being so, people have come up with a clever program that allows us to enter our machine code using mnemonics that humans can understand, like LD A and RET, and then translates them into the numbers the micro expects.

Kuma have brought out the popular ZEN assembler for the Amstrad. You might want to try using this, in conjunction with Mike's articles.

Graphics poser

MAY I be one of the first to congratulate you on your new magazine. I'm sure I won't be the last, because it has that little something that certainly endeared me to your sister magazine, The Micro User.

I enjoyed every page of it and in particular Al's Beat. Was he really a copper? If he was he was missing his way.

I didn't like the way he knocked the ZX80 because I learned on that and found it great fun. Even so I found the article amusing and hope it is a regular feature. – **Cliff Egerton, Wallasey.**

■ Yes, Al was a copper for 18 years making the rank of sergeant, before he left to join us six months ago. He wasn't really knocking the ZX80 as he realises that he owes his present dilemma – working with the A team – to it.

The praise in your letter almost certainly guarantees a regular contribution from him.

Beats 'em all

I AM having problems with graphics on my machine. I know how to define a foreground colour on the text screen using the INK and PEN commands, but when PRINTing at the graphics cursor using TAG, they seem to have no effect.

I have managed to get round the problem by using the PLOT command, which has a parameter for setting the graphics write colour. However it necessitates setting a point off-screen as follows . . .

PLOT –10,100,1

Is this the only solution? – **Alex McKay, Inverness.**

■ We have experimented with this and at the moment cannot find any suitable alternative. We are sure one of our experts out there can come up with an answer.

Micro battle

I'VE got the best micro in the world – an Amstrad – but my elder brother has got an Electron and keeps saying that his is better than mine. What should I tell him? – **Helen Pratt, Newcastle.**

■ There's no point in telling him anything Helen, it's just not worth getting involved in arguments about which is the better micro. Each micro has something that the other hasn't.

For example, the Electron has an Assembler while the Amstrad has 27 colours.

The best micro is the one that suits you best. As it is, we all think that you're lucky having two micros in the house.

Early words

NOT that I'm the suspicious type – otherwise I wouldn't have got married when I did – but did you really get all those letters in the first issue or did you make them up? – **Mark Marks, Trowbridge, Wilts.**

■ Did we make you up Mark?

No real danger

MY mother keeps telling me that I'm getting too close to the monitor's screen. She says that it will make me go blind. Is this true? – **Kevin Black, Orpington.**

■ Only if you keep banging your head on the monitor, Kevin. To be serious, you shouldn't get too near as you could suffer eyestrain. Of course, the knob on the side of your monitor can be used to control brightness. As with everything, moderation is to be recommended.

CALLs can clobber!

WHILE trying out Basic commands I was not familiar with I found out how fatal the CALL command can be.

This happened when I entered CALL 0. The screen went blank and the start up display appeared – the same effect as Shift+Ctrl+Esc.

May I warn other readers of the almost fatal CALL command. – **Derek Fergus, Nottingham.**

■ Quite right! The CALL command can reset or totally "crash" the machine unless it is used correctly. In many cases even Shift+Ctrl+Esc will have no effect.

CALL allows you to execute a machine code subroutine at the location you specify.

If the location is not the start of such a routine then the Z80 processor will try to interpret what it finds – in most cases this will be totally garbage. The old saying goes "Garbage in, garbage out".

Advice to all – use CALL with great care!

Now you've started computing with the Amstrad you'll want to read EVERY issue of . . .

Computing with the AMSTRAD

It's the only way to keep right up to date with what's going on in the ever-expanding world of the Amstrad.

SPECIAL OFFERS!

How to protect your CPC464

Protect your micro with our luxury dust cover made of soft, pliable, clear and water-resistant vinyl, bound with strong cotton and decorated with the magazine's logo.

DUST COVER ONLY £3.95

How to keep your collection complete

Bound in rich burgundy pvc and bearing the Computing with the Amstrad logo, this handsome binder will hold a year's supply of the magazines, firmly secured in place with metal rods.

BINDER ONLY £3.95

Look what you will get each month:

- ★ Reviews of all the very latest games, educational and business programs now being produced for the Amstrad.
- ★ Lots of listings you will be able to key in yourself – games, utilities, graphics – covering the whole field of Amstrad computing.
- ★ In-depth independent evaluations of all the new hardware add-ons now being developed to make your Amstrad much more powerful and much more versatile.
- ★ Lots of easy-to-follow features on everything to do with the Amstrad. Whether you're a beginner or an expert, you'll always find something to delight and intrigue you.

You can also use the form below to order a copy of last month's widely-acclaimed launch issue of *Computing with the Amstrad*.

We're going to make this the most exciting computer magazine ever – so don't miss an issue! If you've got a CPC464, or about to get one, let Computing with the Amstrad show you how to make the most of it!

INTRODUCTORY OFFER!

Take out a 12-month subscription now and pay only £10 instead of the normal £12.

This special offer applies to UK readers only and is valid until February 28, 1985.

SAVE £2!

ORDER FORM

Please send me:

- ☐ The next 12 issues of Computing with the Amstrad at the special introductory offer of £10.
- ☐ Last month's launch issue at £1.
- ☐ Dust cover at £3.95.
- ☐ Magazine binder at £3.95.

I wish to pay by ☐ Cheque ☐ Access ☐ Visa

Card No. .....

Expiry date .....

Signed .....

Name .....

Address .....

.....

.....

.....

Send to: Database Publications, FREEPOST, Europa House, 68 Chester Raod, Hazel Grove, Stockport SK7 5NY.

No stamp needed if posted in the UK.

HISOFT

presents

FONT 464

for the

AMSTRAD CPC 464

FONT 464 *is a font designer and character generator especially developed for the CPC 464 microcomputer.*
Design your own character fonts and graphic symbols with this very friendly and powerful package.

FONT 464 *allows you to create a new design or amend an existing one using set, reset, invert, reflect, rotate, inverse and even animation! Load and save character sets to/from tape, use the new character(s) from BASIC, design your own animated graphics – all this and more with* **FONT 464.**

FONT 464 *is supplied with three interesting and amusing character sets for you to experiment with.*

★ *All this power for: £7.95 inclusive* ★

We also have available for the Amstrad CPC 464:

Hisoft Devpac – our full Z80 assembler and disassembler/debugger with more features than you'll ever need.

Hisoft Pascal – a virtually full implementation of Standard Pascal. Compiles and executes incredibly quickly.

HISOFT
180 High Street North
Dunstable, Beds. LU6 1AT
Tel: (0582) 696421

★ NOW AVAILABLE FOR THE AMSTRAD CPC 464!!

WIN THE POOLS?

SPECTADRAW 3 – THE LATEST VERSION OF THE ORIGINAL AND BEST POOLS PREDICTION PROGRAM FOR THE 48K ZX SPECTRUM!!

AND NOW . . . AMSTRA-DRAW – THE FIRST POOLS PREDICTION PROGRAM FOR THE AMAZING NEW AMSTRAD CPC 464!!

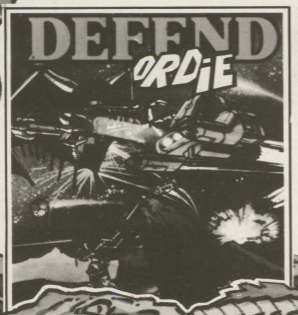
- Supplied with Database containing data on over 10,000 matches since 1980!
- You update the Database each week – but no tedious typing, as team and division names already in program!
- Errors easily corrected – the program even checks your entries!
- Comprehensive instruction manual and menu driven program – easy to use, even for a newcomer to computing!
- Will forecast the least likely draws for those who prefer to bet on fixed odds!
- Built in perm generator – complete your coupon direct from the screen!
- Fully microdrive compatible! (Spectadraw only).
- Compatible with Currah Microspeech – the first pools program to read you its predictions! (Spectadraw only).

Spectadraw 3 for the 48K Spectrum ..... £9.95 inclusive
Amstra-Draw for the Amstrad CPC 464 ..... £9.95 inclusive
(Cheques/P.O.s payable to B.S. McALLEY)

We dispatch every Monday with the database made up to include all matches up to the date of dispatch.

SPECTADRAW (Dept A2),
1 Cowleaze, Chinnor, Oxford OX9 4TD.
(Tel: 0844 52426)

THE ULTIMATE MISSION OF MERCY
DEFEND OR DIE!



DEFEND OR DIE

A classic battle spectacular where skill and reflexes are pushed to the limit, nerves strained to breaking point. Take up the solo mission of saving fellow humanoids from being plucked from your planet surface by the evil landers and then returned to outer space where they will eventually mutate and take up a pursuit on your spacefighter. And as if one deadly struggle is not enough, watch out for the flying pods, swarmers, alien bombers and deathly baiters they're certainly not on your side. A hero is called for, can you fit his spaceboots.

Amstrad

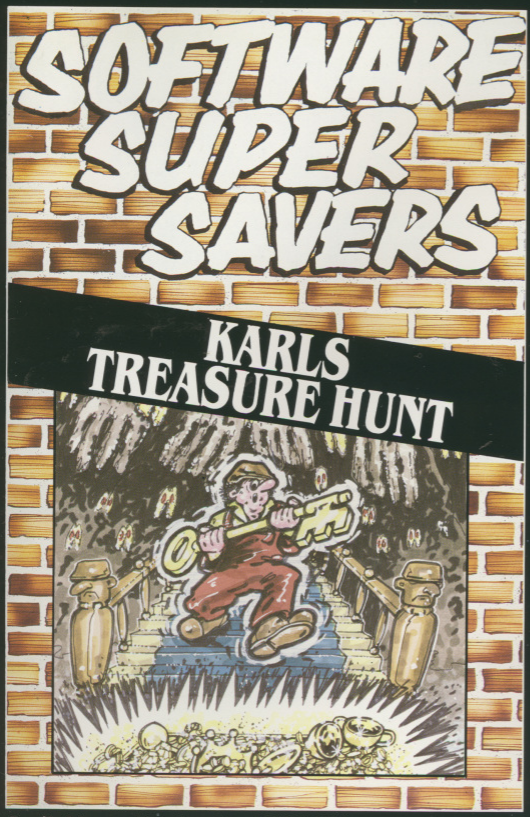
Send for full colour product leaflet (enclose a stamp)
Alligata Software Ltd., 1 Orange Street, Sheffield S1 4DW
Tel (0742) 755796

Alligata Software Limited

TAPE £7.95

Available through good computer stores everywhere

Attention all Amstrad owners!



KARLS TREASURE HUNT

Our hero Karl has fallen on hard times so when he learns that his entry has won 1st prize in a quiz competition he is naturally delighted. His prize is a weekend stay at 'Wonga Mansion' and hidden somewhere in the mansions 40 rooms is a treasure chest. Karl has to collect the 40 keys to unlock the chest. Can Karl's luck be changing for the better.

£2.99

Available on the Amstrad CPC 464

Software Supersavers is a new name to watch out for. We'll be bringing you quality software at a supersaver price. They're not re-hashes of old games but totally original ideas combining to give you an exciting range of new games.

So whatever your software tastes are, Software Supersavers has the game just right for you.

Dealer Enquiries – 051-428 6367 and ask for Lesley

Software Supersavers Ltd., P.O. Box 13, Liverpool L25 7AG

Please send me a copy of
KARLS TREASURE HUNT

I enclose cheque/PO for .....
(Please add £1.00 for orders outside UK)

Access Card No. .....

Name .....

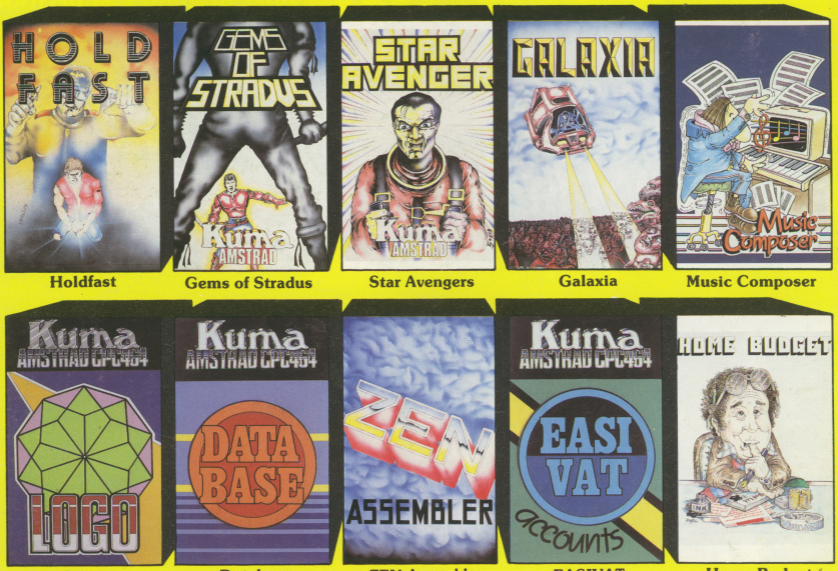
Address .....

.....

For Mail Order only:
Software Supersavers Ltd., P.O. Box 13, Liverpool L25 7AG

the only choice

Kuma AMSTRAD CPC464 software



Holdfast | Gems of Stradus | Star Avengers | Galaxia | Music Composer

Logo | Database | ZEN Assembler | EASIVAT | Home Budget

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC464 Micro-computer.

Book:

● **The Amstrad CPC 464 Explored**

This superb book is designed to let every CPC464 user, at whatever level, get the most from his computer. After an introductory section on the special Basic features, the book looks in depth at the excellent sound and graphic facilities including: ● *Animation* ● *Windows* ● *Character sets* ● *Multitasking* ● *3 Voice Times* ● *M/C routines for Basic* ● *Use of Zen* ● *Use of O/S* ● *Sample programs*

Available from your nearest Amstrad CPC464 Stockist.

Kuma Computers Ltd., 12 Horseshoe Park, Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name .....

Address .....

.....

..... Phone.....

Trade Enquiries Phone 07357-4335