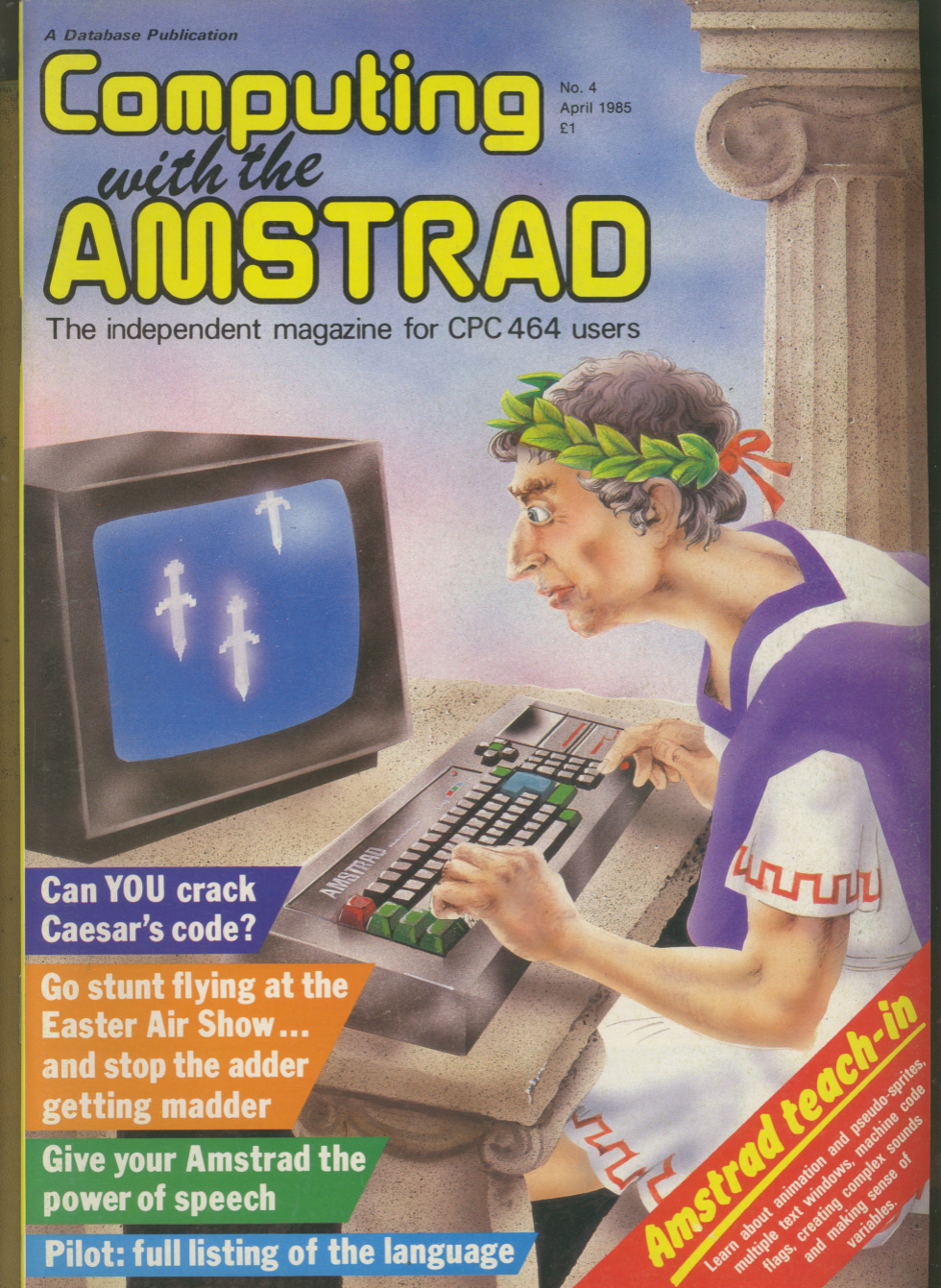
A Database Publication

Computing with the AMSTRAD

No. 4
April 1985
£1

The independent magazine for CPC 464 users



Can YOU crack Caesar's code?

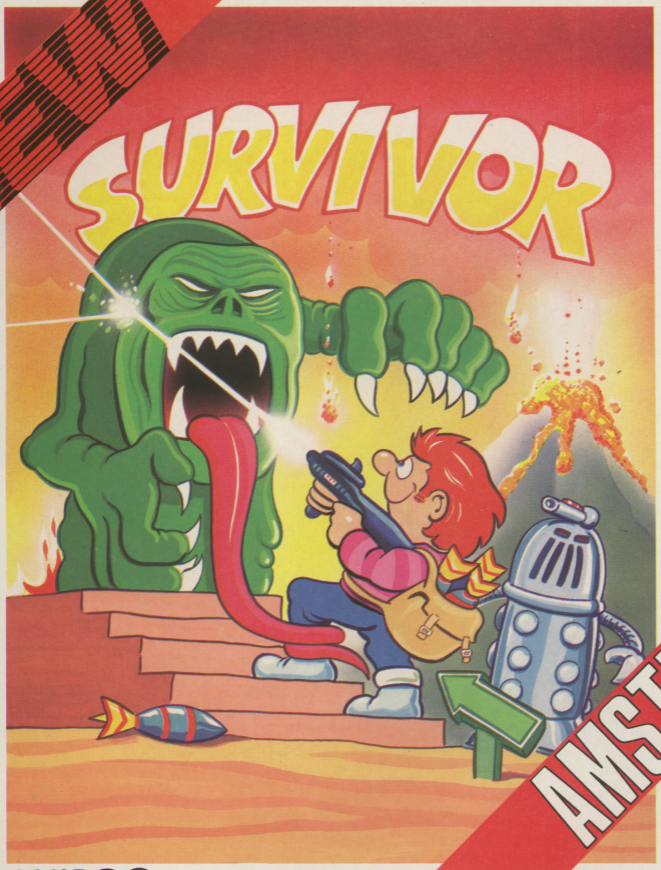
Go stunt flying at the Easter Air Show... and stop the adder getting madder

Give your Amstrad the power of speech

Pilot: full listing of the language

Amstrad teach-in

Learn about animation and pseudo-sprites, multiple text windows, machine code flags, creating complex sounds and making sense of variables.



AMSTRAD

ANIROG

SURVIVOR Search the haunted rooms of Deadstone Abbey for the untold treasures left from years gone by. However, as you help Angus around the ancient building beware of the evil spirits who will chase Angus wherever he goes. All he has to defend himself is his trusty gun and porcupine bombs. Luckily for Angus there are various objects lying around the Abbey such as ammunition, money bags, and bottles of life giving elixir. There are 1008 various rooms all presented in remarkably clear and colourful graphics with beautifully smooth scrolling screens. Ahead lies a terrifying challenge for Angus and its up to you to help him. Are you the sole survivor!

AMSTRAD £7.95

HOUSE OF USHER Enter the House of Usher at your own risk, as you may never leave again. However, once inside there is a choice of nine rooms to select. Behind each door is a totally different action packed arcade game, each of which are certain to strain your nerves to the limit. If you manage to get through these nine rooms another two secret rooms (x and y) will appear, but beware the evil powers of the House of Usher.

AMSTRAD £7.95

FLIGHT PATH Flight Path is without doubt the best flight simulator on the C/16 and Amstrad. The many elaborate features include; Altometer, flaps, directional headings, crosswinds, fires, ground warning lights and reverse thrust to name but a few. Also included are smooth graphics as you take off, cruise over mountains, and land once again.

AMSTRAD £6.95

3D TIME TREK As sole survivor of the planet "Corillian" your quest is one of anger and revenge. The starship you are flying is full of the latest inboard computers and extra powerful sensors. Also included are full 3D graphics, to add unbelievable realism to this fantastic journey through time itself, and beyond.

AMSTRAD £7.95

MOON BUGGY You must skillfully manoeuvre your jumping patrol vehicle over dangerous moon craters as well as large boulders and cunningly placed mines. Not only this but avoid the hovering alien spaceship as it bombards you from above.

AMSTRAD £7.95

Mail Order: 8 HIGH STREET HORLEY SURREY 24 HOUR CREDIT SALES HORLEY 02934 6083 Payment by: P.O. - ACCESS - VISA

AVAILABLE FROM YOUR COMPUTER STORE

AMSTRAD BBC CBM64 SPECTRUM MEMOTECH ENTERPRISE ATARI MSX

EMERALD ISLE

Emerald Isle lies in the Bermuda Triangle, hidden by fogs and peopled by the remnants of a strange civilisation. This great new adventure has 200+ locations and 200+ pictures on the Amstrad, BBC, Commodore 64, MSX and Spectrum.

As an adventure fan, I viewed the prospect of reviewing the latest Level 9 game with excitement. This particular game marks something of a landmark since it's the first Level 9 adventure with graphics and, much to my surprise, the game doesn't suffer adversely. Granted the location descrip... shorter than before... assured there i... text.

The game... Snowball an... ego Kim Kir... the Planet Ec... life. The sc... with about 2... of them illustra... any details of the ... keep you busy for man... hour.

The game is full of tricky puzzles, some horrendously devious. You gain points for solving the adventure, not for collecting treasure.

The illustrations are drawn in multicolour mode and while relatively simple, are well designed and do enhance play. Best of all, they're drawn quickly. The game is accompanied by a very good instruction manual and the usual envelope for a free hint.

Overall, excellent value which, in my view, is unsurpassed by any other software house in this country. Buy it and enjoy. A.W.

Level 9 produces a range of text games that are inventive, involved and usually large. If you're keen to ... version of the original mainfr... venture its **Colossal Adventu...** lot of similarities but has more ... and a slightly altered set of p... Other games by the same ... carry on from where...

Pick of the week

Hard core adventure fans and all persons of taste and discernment will be delighted to hear that the sequel to *Snowball*, *Return to Eden* is now available for almost... than almost anywhere else. You'll need it. A mandatory purchase for adventure fans

CRASH Readers Awards 1984 Winner

BEST TEXT-ONLY ADVENTURE

SNOWBALL

by LEVEL 9 COMPUTING

Text-only adventures still represent a large chunk of the market, with Level 9 one of the most respected (although they too have turned to graphics with their latest release). *Snowball* won this category hands down, with *Dennis through the Drinking Glass* by Applications coming in second and Incentive's *Mountains of Ket* following close behind. There were many votes in this category for games which were not copyrighted in 1984, and were thus ineligible

In adventures it is usual to boast of the number of locations, but *Snowball* probably takes the biscuit with over 7,000. Of the game, Derek Brewster said, 'This is very much my idea of an adventure and is set to become a classsic.'

Return to Eden

A new title from Level 9 is always reckoned to be an Event, and Pete Austin groupies won't find anything to disappoint them in Return to Eden – which also marks Level 9's first graphic ...ering. I'm still not sure whether ...ures actually improve my ...ment of an adventure – but ...'s visual appeal makes a ...ng case for them. Return ...atures nearly 250 ...'m told!) and those I ...o explore were ...nd quickly drawn. ...features ...m Kimberley ...ring 110% in ...erself trying ...r rap in ... and ...d to me ...it sure ...awberry

...ridge ...to Eden ...el 9 Computing ...9 Hughenden ...oad High Wycombe Bucks HP13 5PG

...'s other games. ...act, Pete Austin is ...olific in adapting his ...s to other machines, ...I fully expect them ...be available on washing ...achines soon!

Briefly, it's fully up to Level 9 standard. And that's praise indeed.

...utter stacks of ...sorbing puzzles ...for logical and/or ...pired solutions, but the truly confounded (or just simple cheats like me) can write to Level 9 for a free sanity-saving clue sheet.

...'s *Dungeon Adventure* ... descriptions of the vario... ...s and characters in the ...e superb — and once ... find it very difficult to drag ...e of the world hidden ... the computer. Be warned, *Dungeon Adventure* certainly gets ten out of ten for addictive quality!

Popular

It also includes graphics, and is full of the frustration problems which are Level 9's hallmark.

Available from the HMV Shop and good computer stores everywhere. If your local dealer doesn't stock Level 9 adventures yet, use the coupon to buy them from us, or ask him to contact: Centresoft, Gay Byrne, Greyhound, Lazer, Leisuresoft, Lightning, Lime Tree, LVL, MCD, Microdealer, R&R, TBD, Thorn EMI, Tiger, Websters, Wonderbridge etc.

	Cassette £9.95	Disk £11.95 BBC/CBM64
COLOSSAL ADVENTURE: The classic mainframe game, with 70 bonus rooms.	☐	☐
ADVENTURE QUEST: An epic puzzle journey through Middle Earth.	☐	☐
DUNGEON ADVENTURE: 40 treasures to find and 100+ puzzles to solve.	☐	☐
SNOWBALL: Immense Science fiction game with 7000+ locations.	☐	☐
RETURN TO EDEN: Vegan SF adventure (Amstrad, CBM 64, MSX and Spectrum versions have 240 pictures).	☐	☐
LORDS OF TIME: Imaginative romp through World History.	☐	☐
ERIK THE VIKING: Game of Terry Jones' book for Amstrad, BBC, Commodore 64 and Spectrum only. All with 180+ pictures.	☐	
	Cassette £6.95	
EMERALD ISLE: Fantasy adventure for all micros. Amstrad, BBC, Commodore 64, MSX and spectrum versions have 230 pictures.	☐	

I ENCLOSE A CHEQUE/PO FOR £9.95 PER CASSETTE (£6.95 FOR EMERALD ISLE) OR £11.95 PER DISK

My name: ____________

My address: ____________

My micro is ____________ (one of those listed below with at least 32K, or 64K for MSX graphics games). Send coupon to:

LEVEL 9 COMPUTING
Dept , 229 Hughenden Road
High Wycombe, Bucks. HP13 5PG

CWA/4

AMSTRAD BBC CBM64 SPECTRUM MEMOTECH ENTERPRISE ATARI MSX

Vol. 1 No. 4 April 1985

Managing Editor: **Derek Meakin**
Features Editor: **Peter Bibby**
The A Team: **Mike Bibby**
Alan McLachlan
Kevin Edwards
Roland Waddilove
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.

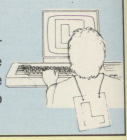
Subscription rates for
12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

ABC Member of Audit Bureau of Circulations

7 NEWS
Keep up to date with the latest happenings and new arrivals in the busy, expanding world of the Amstrad computer.

13 BITS AND BYTES
We continue our exploration of binary numbers with a look at the logical operators NOT, AND and OR.

14 BEGINNERS
In the latest instalment of our popular series for the novice programmer we learn the benefits of longer variable names, and how to convey data to them via INPUT.



18 CAESAR CIPHER
Use your Amstrad's sophisticated string handling techniques to investigate one of the earliest codes in history.

21 SOFTWARE SURVEY
More and more software houses are producing programs for the CPC464. Here the latest releases come under the scrutiny of our team of frank and thorough reviewers.

26 MACHINE CODE
Our easy-to-follow machine code series continues with a look at loading to and from memory, easy sums, flags and simple loops.

30 SIMPLE SPRITES
By making use of Amstrad Basic's logical operators and in-built interrupts you can create sprite-like effects to enhance your programs.

Contents

32 SPELLING

You have to match the correct spelling of a word with its definition in this educational program designed for the early learner.

36 SOUND

This month Nigel Peters addresses the envelopes in the fourth part of his definitive series on sound.

42 ANIMATION

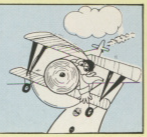
In the first part of a major series on Amstrad animation we investigate moving objects around the screen using the PRINT and LOCATE commands.

45 ANALYSIS

Trevor Roberts takes a look at simple string manipulation, using it to create a palindrome tester.

49 EGG BLITZ

There are Easter Eggs, a stunt pilot and a Human Pyramid of boy scouts in our special game for Eastertide.



52 GRAPHICS

We continue looking into windows with a discussion of multiple text windows. Double gazing, indeed!

61 ALEATOIRE

From submarine action to playing your cards right, our regular puzzle series continues its look at random numbers.

62 MAD ADDER

Can you appease our aggravated adder's appetite by steering him in his relentless search for food? There's fun for all the family in this classic game.

64 PILOT

Another language for your Amstrad. Pilot is easy to understand and ideal for computer aided learning.

69 POSTBAG

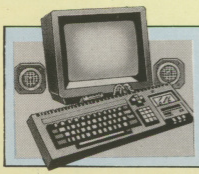
Just a small selection from the many interesting and informative letters you've been sending us.

74 AL'S BEAT

Our tame Mr Plod takes a look at who's who in the world of programming and gives some hints on debugging programs.

76 READY REFERENCE

We put the facts on string handling, number and case conversion at your fingertips.



THIS MONTH'S SPECIAL READER OFFER

Teach your Amstrad to talk with the dk'tronics speech synthesiser, stereo amplifier and twin speakers – all specially designed to compliment the Amstrad CPC464. And with our special reader offer you can save £5 off the normal retail price.

Details on Page 10

GET SORCERY OR YOU'LL TURN INTO A FROG

"Get Sorcery or you'll turn into a frog." Well, that was the verdict of Which Micro on the Commodore 64 version. Now available on the 14th February is a greatly enhanced, stunning new version of Sorcery for the Amstrad and MSX. 40 screens of amazing and breathtaking graphics (50 in the case of the MSX version), with endless hours of tough and challenging game play.

What the press said about the '64' version:

"Get Sorcery or You'll turn into a frog" – Which Micro (CBM 64).
"Virgin's best game since the screen-scorching Falcon Patrol I & II" – Commodore User (CBM 64).
"I have played this game every night since I received it" – TV Gamer (CBM 64).
"Fast, wacky and very polished" – Commodore Horizons (CBM 64).

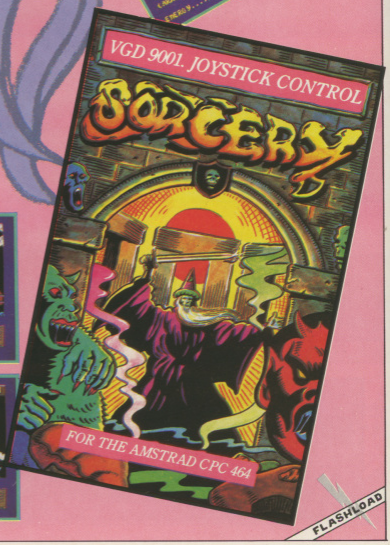
The World has fallen into the Dark Ages. The Evil Forces of the Necromancer are creeping to every last corner of the ancient land that was once the powerful domain of the Great Sorcerers. You are the last free Sorcerer, all the others have, one by one, been entrapped by the hideous evil that abounds.

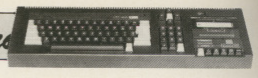
So don't delay, get your copy now. It's available now from leading computer stores.

Available for

AMSTRAD CPC 464 £8.95	CBM 64 £8.95
MSX £8.95	SPECTRUM 48K £6.95

ALL SCREEN SHOTS FROM AMSTRAD CPC 464 VERSION.





Business future is bright

THE Amstrad has a brilliant future as a vehicle for specialist software applications in the commercial field according to MASS Business Systems.

The company has chosen the disc-based CPC464 as the cornerstone of its tailor-made software packages for small businesses.

Says MASS director Stephen Lake: "We believe a whole new market is opening up for the Amstrad.

"The CPC464 with specially written software puts the cost of specialist application programs well within the range of thousands of people who would not normally have considered a computer for their business".

Specialist

The MASS approach is simple. First the company finds a small business application and, in conjunction with the experts from that particular field, produces specialist software.

Nothing unusual in that. However, as the cost of the complete service – including all hardware, software, delivery, tuition, help lines and maintenance – is well under £1,000, others within the selected industry are attracted, thus creating a new market.

Utility

Applications so far include software for butchers, scrap merchants, cardboard box manufacturers, vehicle recovery – and even a program to assist with sausage making.

Says Lake: "Take for example the butchers' utility which assists with carcase costings. This was previously only available on a system costing more than £3,000.

"Our system with added features sells for £749.95 and we expect to sell more than 200 systems this year".

Education authorities opt for the Amstrad

AMSTRAD has made the big breakthrough into the education market and is set to challenge the strongholds of RML and BBC in the coming months.

Three local education authorities in the North West have already officially accepted the CPC464 as an alternative to the BBC Micro for teaching purposes.

And between 20 and 30 other LEAs across the country have made enquiries about the Amstrad or are currently evaluating the machine.

Interest is especially keen in London and Kent, areas where the RML has been the educational market leader.

Amstrad's main selling point is its price to schools – £189 for a green screen CPC464 or £279 for the colour version.

This makes it roughly half the price of the equivalent BBC Micro and one third the price of the ageing RML 480Z.

Gareth Littler, of Amstrad's educational distributor Northern Computers, told *Computing with the Amstrad:* "Our aim this year is 100 machines a day to schools and colleges.

"This would put us ahead of RML and Sinclair and alongside the BBC".

He added that should Acorn's current problems result in the BBC contract becoming available, Amstrad would compete for it against ACT's Apricot and the Sinclair Spectrum.

THE CPC464 has been voted Most Exciting New Computer in a poll of readers of a weekly computing magazine. Amstrad finished well ahead of second place Sinclair QL, with the ZX Spectrum third. Fourth was the Commodore Plus 4 and fifth the Enterprise 64k.

Improve

Littler said strenuous efforts are being made to increase the amount of educational software available for the machine, but agreed there was still a great need for programs at GCE level.

"However we expect this situation to improve quickly as software houses react to our penetration of the educational market", he added.

BIG BROTHER IS DUE

THERE will be four new major Amstrad products this year, but only one of them is likely to be a computer.

The other three will be add-ons, *Computing with the Amstrad* understands, one of them a networking interface.

The new computer will be a larger, more expensive version of the CPC464 with 128k memory and built-in disc drive.

It will run all the software from the CPC464, including that written for cassette.

The new machine will be exhibited to the trade in May and released to the public in late June, according to Amstrad.

"There will be no price change for the CPC464. It is already recognised to be the most cost effective computers on the market", say Amstrad.

5¼ in DRIVE ARRIVES

A PLUG-IN 5¼in second drive for the CPC464 has found a ready market with Amstrad users.

Timatic Systems, who developed the drive in cooperation with Amstrad to augment the company's own three inch drive, claim to be "delighted with the response".

Nor does the company expect the boom in interest to diminish. "It will grow correspondingly as users' appreciation of the machine grows", a Timatic spokesman told *Electron User.*

"They will realise it is a gateway to a whole world of other software".

Composer released

SOFTWARE publisher Kuma Computers has released Music Composer for the Amstrad owner, whether experienced musician or musically inclined novice.

The complete ranges of notes, rests, key signatures and tempo settings allow those with a knowledge of music a large degree of flexibility when composing.

As music is composed it is displayed in musical notation form on the screen. Pieces can be saved onto cassette for future use and there are also facilities for transposing, copying and combining pieces.

Music Composer comes on cassette complete with manual and costs £9.95.

HEADING FOR THE HIT PARADE

THE popular Amstrad games character Wally Week from Micro-Gen could be on his way to becoming a recording star.

His latest program "Everyone's A Wally" is the subject of a song written by John Watkins (left) pictured here with Micro-Gen managing director Mike Meek and actor/singer Mike Berry (right).

Meek says several major recording companies are interested in releasing the Wally song and before long it could be competing for a place in the Hit Parade.

Adventures and games arrive thick and fast

FOLLOWING up its Amstrad adventure titles, Interceptor Micros has released its first arcade game for the CPC464.

Chopper Squad puts users behind the controls of a rocket helicopter with the task of saving the world from alien attack. Price is £6.

★ ★ ★

THE best selling flight simulation program Fighter Pilot from Digital Integration has been converted for the Amstrad.

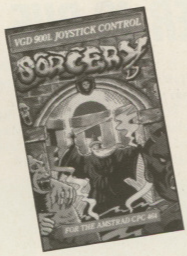
The program puts the user in the pilot's seat of one of the world's most exciting fighter aircraft – the F15 Eagle USAF jet.

In the Amstrad version sound effects have been added to heighten the realism of the 3D cockpit view and air-to-air combat. Price is £8.95.

★ ★ ★

FIRST titles for the Amstrad from PSS are translations of the software firm's successful "Frank 'N' Stein" and "Battle for Midway" programs.

Amsoft will market "Frank 'N' Stein", a 50 level platform game in which the player has to collect the parts of Frank's body. Price is £8.95.



The World War II "Battle for Midway" will be handled directly by PSS. Price is £9.95.

★ ★ ★

FOUR of the most popular releases from Mirrorsoft have been adapted for the CPC464.

Award-winning The Games Creator, £12.95, allows users to design, orchestrate, play and save their own unique games.

And Mirrorsoft says it will publish the best programs designed by Amstrad owners using Games Creator.

For early learners there are First Steps With the Mr Men, £8.95, and Here and There With the Mr Men, £7.95.

Caesar's Travels is a complete interactive program storybook package with 18 different endings and costs £7.95.

★ ★ ★

A VERSION of "The House of Horrors" from Solid Software has been produced for the CPC464. The maze adventure with arcade elements, in which problems are presented to the player in text and graphic format, costs £6.95.

★ ★ ★

AS many as 16 colours appear at the same time in Sorcery, the new release from Virgin Games for the Amstrad.

The game spans 40 different locations peopled by an army of mobile monsters ranging from floating phantoms to disembodied boar's heads.

Cascading waterfalls, simmering cauldrons and swirling rivers make up the water hazards, while skulls and sadistic sentries provide some of the human obstacles.

Aim of the game is to manoeuvre by joystick the world's last remaining sorcerer through the perils and rescue his eight imprisoned colleagues.

"We've aimed to make this game a visual spectacle as well as a mental challenge", says Virgin's commercial director, Jeremy Cook.

"Everything that could be animated we've animated. We believe it's the best game yet for the Amstrad". Price is £8.95.

Special Basic

A SPECIFICALLY commissioned version of BBC Basic for the Amstrad is being produced by Northern Computers.

This CP/M program takes up approximately 16k RAM memory and will only be available on 3in disc. It provides BBC Basic standard commands, procedures, disc files, long variable names and graphics facilities.

Man who bugs micros without really trying

LIKE most first time home computer owners, Peter Strickland has been devoting much of his spare time of late to sitting in front of his gleaming new Amstrad CPC464.

However in his case it isn't just for enjoyment. He's waiting for it to go wrong.

Not that the 42-year-old textile technician hasn't any faith in the machine. After all, he wouldn't have bought one if that was the case.

The reason is that whenever Peter Strickland goes near anything microprocessor-based, it invariably goes on the blink.

"If it involves a computer, you can almost guarantee it will malfunction if I'm around", he told *Computing with the Amstrad*.

A recent visit to a computer controlled textile machinery line ended in chaos simply because Peter was a member of the party. The machine went beserk every time he approached. Yet when he moved a few feet away it returned to running smoothly.

Even his microchip based calculator won't work for Peter. It refuses to add up, subtract or multiply correctly when he presses the buttons. Yet handed over to someone else, it coughs up the correct answers every time.

Now scientists have suggested that the bizarre behavioural patterns of computers when Peter Strickland is around may be due to him being a multi allergenic.

That's someone who is highly susceptible to allergic reactions – so much so that they can be triggered off from a distance.

The mere whiff of North Sea gas from "half a mile away" is sufficient to render Peter Strickland almost comatose – and that's just one of a thousand things to which he is allergic.

Researchers are now working on the theory that people like Peter are particularly receptive to electronic fields. And some experts are convinced that multi allergenics transmit their own electronic fields, so causing interference with computers.

"They believe it is a question of interacting with electronic circuitry", says Peter, who now regularly attends a clinic where attempts are being made to desensitise him.

The usual method of dealing with allergies is to come up with an antigen, an antidote for the symptoms.

But in the case of Peter and his fellow sufferers, scientists believe the answer may lie in an "electrical antigen" made up of a cocktail of different frequencies.

Using oscillators, they have been running a range of electric currents through glass vials containing saline solution which will hold a charge.

Then the patients are called upon to hold the vials for around 10 minutes every day to see if the treatment will in turn ward off "harmful" electronic fields.

"It may all sound a bit far fetched", admits Peter Strickland, "but they have had some very good results so far.

"However until someone comes up with the perfect answer, I'm still expecting my CPC464 to malfunction any time now".

LET YOUR AMSTRAD GO TO YOUR HEAD

GAMES software sound effects like you've never heard them before are promised by Alligata's latest product – a device that adapts the Amstrad's sound output to work with headphones.

The "electronic box" amplifier – so new it hasn't been given an official name as yet – will work with both high and low impedance headphones, including those made for hi-fi enthusiasts as well as the small "Walkman" type.

Fidelity

Alligata says the device will cost about £20 and will probably be released in May.

It will use the stereo capability of the Amstrad – the only computer capable of producing stereo sound – and will control fidelity as well as sound level and prevent distortion due to overloading.

Light pen on the way

A HIGH resolution light pen for the Amstrad from DK'tronics should be launched in the next few weeks.

A spokesman for the company told *Computing with the Amstrad* that work on the product was "very advanced – to the point where we're almost ready to put a price on it.

"We are hoping to get a much higher resolution for this light pen than for others on the market, while still pricing it competitively".

WITH SERVICE IN MIND...

THE Amstrad has proved itself just about the most reliable micro on the market. But when something does go wrong you may find your dealer doesn't offer a repair service, or if he does you may have a long wait ahead of you.

Fortunately a number of firms have set up facilities to bring help fast to micro owners with problems.

One of them is Rumbelows which has 40 service clinics throughout the country where you can take your CPC464 for on the spot repairs.

Rumbelows' computer engineers undergo an intensive training period to develop their skills, and are also qualified to install peripherals.

Once the work has been carried out the Rumbelows' engineer will then give the machine a comprehensive test using the Diagnostic Rompack, which has been designed to test all functions of the Amstrad.

Now you can teach your Amstrad to talk!

How it works

AT the heart of the dk'tronics speech synthesiser lies an incredibly powerful chip that has split the English language into its component parts – or allophones as they are known.

Altogether there are 59 allophones and five pauses stored in the speech chip's internal ROM. These can be combined to create a virtually unlimited vocabulary.

The potential of this chip is realised by dk'tronic's sophisticated, yet simple to use software. The brilliant program design enables the Amstrad to actually speak the words you type, in straightforward English, without having to resort to complicated phonetic spelling or difficult programming techniques.

Written to be as user friendly as possible, the synthesiser adds eight powerful commands to Amstrad Basic.

If you prefer complete control over your programs, though, full details are given for Basic and machine code programmers to exploit the tremendous scope of the synthesiser without using the software supplied.

In fact this system supports four different modes of use.

The first mode allows you to sound words using only the Amstrad's normal Basic commands. However, as you get more ambitious with your speech, a second mode is provided. This gives eight extra commands to use from Basic, making using the synthesiser even easier.

The third mode is the text to speech converter. When this is in operation speech can be typed in using normal English and the Amstrad does the rest. There's no need to work out the allophones as in the other two modes – the Amstrad does it for you.

As if all this wasn't enough there's the fourth mode. This has the synthesiser converting whatever appears on the screen into speech. Using this, you can literally listen to your listings!

YOU can add an exciting new dimension to computing with your Amstrad – with the help of this remarkable new product from dk'tronics.

It comes complete with the latest and very versatile speech chip, a powerful stereo amplifier and two high-quality 4in speakers, specially designed to match the Amstrad CPC464.

And because this is a special reader offer it comes to you at £5 off the normal retail price of £39.95!

Fitting it is simplicity itself. All you have to do is to plug the synthesiser's interface into the floppy disc port at the back of the Amstrad and the jack plug into the stereo socket – and away you go!

With its volume and balance controls you will find you can put dramatic realism into the sound output of your Amstrad. All sounds that previously came from the Amstrad's 1½in mono speakers are now sent out via the interface in stereo.

So even when you're not using it as a speech synthesiser, it can bring startling depth and drama to the music and sound effects of all your favourite games!

These are the sounds – and pauses – you can create on your Amstrad

A	AE	26	fat	F	FF	40	fire	NG	NG	44	bans	TH	TH	29	thin
A	AY	20	great	G	GG2	61	go	O	O	23	cot	TH	DH1	18	they
AI	AIR	47	hair	IG	GG3	34	wig	O	OW	53	snow	TH	DH2	54	bathe
AR	AR	59	farm	GU	GG1	36	guest	O	UW1	22	do	U	AX	15	succeed
AU	AO	24	aught	GE	ZH	38	beige	OO	UW2	31	food	UH	OO	30	cook
B	BB1	28	rib	H	HH1	27	he	OR	OR	58	store	U	YY1	49	compute
B	BB2	63	big	H	HH2	57	hoe	OU	OW	32	ouch	V	VV	35	even
C	KK1	42	common	I	IH	12	fitting	OY	OY	5	toy	W	WW	46	wool
'C	C	8	uncle	I	Y	6	sky	P	PP	9	pub	WH	WH	48	whig
K	KK2	41	sky	IR	ER2	52	bird	R	RR1	14	real	Y	YY2	25	yes
CH	CH	50	church	J	JH	10	jury	R	RR2	39	brain	Z	ZZ	43	zoo
D	DD1	21	could	L	LL	45	luck	R	YR	60	fear	PA1	PA1	0	10 mS
D	DD2	33	do	L	EL	62	angle	S	SS	55	sat	PA2	PA2	1	30 mS
E	EH	7	bend	M	MM	16	milk	SH	SH	37	shirt	PA3	PA3	2	50 mS
E	EE	19	see	N	NN1	11	earn	T	TT1	17	its	PA4	PA4	3	100 mS
ER	ER	51	cater	N	NN2	56	no	T	TT2	13	top	PA5	PA5	5	200 mS

Column 1: Sound. Column 2: Allophone name. Column 3: Allophone number. Column 4: Example word.

Look at what this package offers you:

- ★ Speech synthesiser with almost unlimited vocabulary
- ★ Easy-to-use commands – it accepts normal English words
- ★ Built-in stereo amplifier with twin speakers
- ★ Programs can run while the speed chip talks

Eight additional Basic commands

|SPON	Speech on.
|SPOF	Speech off.
|FEED,n	Feed speech buffer direct.
|FLUS	Clear speech and text buffers.
|SPED,n	Speech speed.
|OUTM,1	PRINT text to speech.
|OUTM,2	Screen output to speech.
|OUTM,3	Output to screen and speech.

Please send me the dk'tronics speech synthesiser for my Amstrad CPC464

☐ I enclose cheque for £34.95 *(incl. VAT, p&p)* made payable to Database Publications Ltd.

I wish to pay by

☐ Access Card No. _____

☐ Visa Card No. _____

Signed _____

Name _____

Address _____

POST TO: Speech Synthesiser Offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

Allow 28 days for delivery

CA4

The Chart Topping Flight Simulation Now on CPC 464

FREE COLOUR POSTER AVAILABLE WITH DISK VERSION

FIGHTER PILOT

ALSO AVAILABLE ON 48K SPECTRUM & CBM 64

BY D.K. MARSHALL
ADAPTED BY DARRELL D.

AMSTRAD CPC 464

AMSTRAD CPC 464
Fighter Pilot ☐ £8.95

COMMODORE 64
Fighter Pilot Disk ☐ £14.95
Fighter Pilot Cassette ☐ £9.95

LOADING TIMES
DISK – 147 SECS
CASSETTE – 300 SECS

SPECTRUM
Night Gunner 48K ☐ £6.95
Fighter Pilot 48K ☐ £7.95
Tomahawk 48K **Coming Soon**
TT Racer 48K

Cheques payable to Digital Integration Ltd
I enclose a cheque/P.O. for_____Total
Name_____
Address_____
_____
Or debit my Access Account No.
☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Access
Tele Sales (0276) 684959
Please send to Digital Integration, Dept Watchmoor Trade Centre, Watchmoor Road, Camberley, Surrey GU15 3AJ.
VAT and p. & p. inclusive within UK (Overseas inc. 55p. per cassette)

DIGITAL INTEGRATION
Watchmoor Trade Centre, Watchmoor Road, Camberley, Surrey GU15 3AJ

Trade and Export enquiries welcome. Lifetime Guarantee.

Readers Awards 1984 Winner

Inside story of binary operations...

MIKE BIBBY continues his series on the fundamental workings of the CPC464

IN previous articles we've seen that binary numbers can be added and subtracted just as our more familiar decimal numbers are. And, of course, we can multiply and divide them.

There are, however, other ways of combining two binary numbers that are extremely useful in dealing with computers. They're also easy to use, so let's have a look at them!

Firstly, we'll see how we can NOT a binary number – simple, one-bit numbers first. By the way, we're going to be dealing exclusively with binary numbers this month, so we can drop the % sign.

The rules for doing a NOT are simple:

If the bit is 1 then it becomes 0
If the bit is 0 then it becomes 1

If you like, the NOT converts a bit into its opposite.

So **NOT 1 = 0**
And **NOT 0 = 1**

Why do we use the word NOT? Well, mathematicians often use the number 1 to mean TRUE and 0 to mean FALSE.

So NOT 1 means NOT TRUE, which means FALSE, which is 0. That is, NOT 1 is 0. And, as NOT FALSE is most certainly TRUE, NOT 0 is 1.

If we are to NOT a binary number consisting of several bits, we simply apply the rule for NOT to each bit individually.

So **NOT 10110010**
becomes **01001101**

Some people think of this process as "turning the number on its head" – so it's sometimes called inverting. Others call it taking the complement of the number.

NOT just works on a single binary number. However, there are other sums or operations that have a set of rules for combining two binary numbers.

For instance we can AND two binary numbers. Let's look at the rules for ANDing a single bit with another bit.

When you think about it, there are four possible combinations of bits that we could AND: 0 with 0, 0 with 1, 1 with 0 and 1 with 1.

We write that we are ANDing, say, 0 with 1 as 0 AND 1.

The rules for ANDing are:

0 AND 0 = 0 (case a)
0 AND 1 = 0 (case b)
1 AND 0 = 0 (case c)
1 AND 1 = 1 (case d)

Notice that the only time the result is 1 (TRUE) is when the two bits ANDed are both 1 (TRUE). This helps us to see why we use the word AND to describe the operation.

If you think of the first bit as "this" and the second bit as "that", what we're doing when we're ANDing is asking whether "this and that" is true.

"This and that" can only be true when both "this" is true AND "that", is true – hence the use of AND to describe the process.

For example, consider the statement that it is

dry and sunny

This is true only if dry is true and sunny is true (case d).

If either of the two (or both) are false (cases a, b, c) the whole statement is false, since it isn't both dry and sunny.

We can AND pairs of binary numbers of more than one bit – just apply the rules of ANDing to each bit individually.

For example

10010110
AND 10110011
gives **10010010**

We can also OR two binary numbers. The rules for ORing a single bit with another bit are as follows (again there are four possible combinations):

0 OR 0 = 0 (case e)
0 OR 1 = 1 (case f)
1 OR 0 = 1 (case g)
1 OR 1 = 1 (case h)

In this case, you only get a FALSE result (0) when both bits are FALSE. If either or both bits are TRUE (1) the result is TRUE. It's easy to see why we use OR to describe this. If one, OR the other, OR both is true the whole thing is true!

Let's use the meteorological analogy again. Let's consider the statement that it is

dry or sunny

This is only FALSE when it is NOT dry and NOT sunny (case e), otherwise it is TRUE (cases f, g, h).

To sum up, with OR, the whole thing is true if either or both the things being ORed is true.

As we did with AND, we can OR pairs of numbers with more than one bit – we just apply the rules of ORing to each bit individually.

For example:

10010110
OR 10110011
gives **10110111**

● *In the next article we'll look at EOR, and the use of masks.*

Making capital out of longer variables...

Fourth in MIKE BIBBY's helpful guide through the micro programming jungle

WE saw last month how to label strings with variables. This meant that if we were using a string several times in a program we could use a variable instead of it.

For example:

```
A$="AUSTRALIA"
```

means that, from now on, instead of using "AUSTRALIA" in full in our programs, we can use *A$*.

```
PRINT A$
```

will print out AUSTRALIA for you.

The labels we used last month were all single letters of the alphabet followed by '$'. The dollar sign tells the computer that it is a string we are labelling – such a variable is called a string variable.

It is called a variable because the "contents" of a variable (in technical terms, its value) can vary throughout a program.

Program I should illustrate the point:

```
10 REM PROGRAM I
20 MODE 1
30 A$ = "AUSTRALIA"
40 PRINT A$
50 A$ = "AMERICA"
60 PRINT A$
70 A$ = "AFRICA"
80 PRINT A$
```

Program I

As you will see when you RUN it, the value of *A$* varies as we reassign it during the program. *A$* always takes the last value assigned to it. You may wonder why on earth you would want to use the same variable for different things, rather than label everything separately.

As we shall see, it can be extremely useful.

```
10 REM PROGRAM II
20 MODE 1
30 name$ = "Mr.Smith,"
40 fact$ = "You owe me money."
50 threat$= "Pay up or else."
60 PRINT
70 PRINT "Dear "name$
80 PRINT TAB(5) fact$;threat$
90 PRINT TAB(15) "cordially yours,"
100 PRINT TAB(20) "Mike"
```

Program II

So far we have restricted our string variables to single letters of the alphabet followed by the $ sign, such as *A$, B$* and *C$*.

However there is no need for such a limit – provided we follow them with $. String variables can be made up of several letters, even words.

Program II illustrates the point. It is our most sophisticated program to date, and is well worth having a close look at.

Perhaps the first thing to remark upon is that we are now working in lower case letters as well as capitals. Infuriating as this is at first for the non-typist (myself included), it really is worthwhile.

Notice that in the programs all the Basic keywords are in capitals.

Now the Amstrad forces this on you. Keywords are always LISTed in capitals. If you entered a line such as:

```
10 print "Hello"
```

it would be LISTed as:

```
10 PRINT "Hello"
```

That is, the keyword would be translated into upper case. Notice that there's no such translation of the "Hello". Since this is a string, in quotes, it remains inviolate.

In fact, all the following:

```
10 pRinT "Hello"
10 oRINt "Hello"
10 PRInt "Hello"
```

would be listed as:

```
10 PRINT "Hello"
```

All the variables of Program II *(names$, fact$, threat$)* are in lower case.

This is because I have entered them this way. There's an important point to be made here:

On the Amstrad:

```
10 PRINT threat$
```

and

```
10 PRINT THREAT$
```

would appear different when listed, since the variable would remain in the same case – upper or lower – as when it was typed in. Only keywords are altered.

However, *both lines are equivalent.* The Amstrad doesn't recognise any difference between the variables *threat$* and *THREAT$*. In variable names capitals and lower case letters are equivalent, so:

```
threat$
THREAT$
thrEAT$
```

are considered by the micro to be the same variable.

```
10 REM PROGRAM III
20 MODE 1
30 threat$ = "First"
40 THREAT$ = "Second"
50 PRINT threat$
60 PRINT THREAT$
```

Program III

Program III shows the idea. Line 30 and 40 assign strings to the same variable, despite appearances. The value assigned in line 60 replaces that assigned in line 50 as we saw in Program I.

My recommendation is that you enter all variables in lower case. In

this way, when you LIST the program, the lower case variables will stand out from the keywords, which are LISTed in capitals.

This may not make for easy typing, but it is good programming practice, since you can tell at a glance what's what in a program.

Take a close look at those variable names – we are using actual words for the labels in this program. Again, it is good programming practice to do so, since we can make the label describe what it is labelling. Programs make more sense this way.

Thus we use *name$* to label "Mr. Smith", *fact$* to label "You owe me money", and *threat$* for "Pay up or else".

This may seem long-winded, but it really does help to make your programs more readable, and hence easier to decipher. For example:

```
70 PRINT "Dear " name$
```

really tells you what the line is doing, far more than:

```
70 PRINT "Dear " A$
```

Similarly:

```
PRINT threat$
```

is more meaningful than

```
PRINT B$
```

The moral is, use words for variables (labels) as much as possible – and preferably lower case words.

Actually you can use capitals for variable names and intermix them with lower case letters and also numbers. The rules for doing so are as follows:

- All variable names must begin with a letter, though you can follow this with any mixture of letters, numbers and dots. Letters may be upper or lower case. They will however be considered equivalent.
- You cannot put spaces in the middle of variable names.
- Variables should be kept separate from Basic keywords.

The commonest error is to run a variable into a keyword.

One advantage of using variables instead of directly using strings is that we can easily alter the output of the program.

In the case of Program II, if we want another victim to be the recipient of our letter, just change line 30. For example:

```
30 name$ = "Mr. Jones"
```

From then on all uses of *name$* in the program will refer to Mr. Jones.

In this short program it doesn't make a great deal of difference, but in larger ones, if you had used the string "Mr. Smith" every time, instead of *name$*, you would be in for a lot of retyping.

You'd have to alter every single reference to "Mr. Smith" in the program. If you'd used *name$* throughout, all you'd have to do is change the line assigning *name$*.

Keeping variables in lower case can help us with a perennial problem for newcomers to the Amstrad – mistakenly tagging keywords onto variables.

Enter Program IV exactly as shown – making sure there's a space after print in line 40.

```
10 REM PROGRAM IV
20 MODE 1
30 word$ = "Hello"
40 print word$
```

When you LIST it, you'll see:

```
10 REM PROGRAM IV
20 MODE 1
30 word$ = "Hello"
40 PRINT word$
```

Program IV

RUN the program and all will be fine.

Suppose, though, that you'd missed out that space between print and word$. Enter this version of line 40:

```
40 printword$
```

Now LIST and RUN it. You'll get the error message "Syntax error in 40". As you can see, line 40 is now:

```
40 printword$
```

and the micro doesn't know what to do with this "variable".

More generally, variables tend to "absorb" keywords that run into them, to create larger variables. The keyword in effect "disappears" from the line, leaving the micro in some doubt as to what to do.

So to stop this confusion, leave spaces after – and before – keywords. Actually you can use characters other than space as separators, or delimiters as they are known. Quotes act as delimiters, as in:

```
PRINT"Hello"
```

If you're in the habit of entering variables in lower case, when you list a line, all the words in capitals must be keywords. If there aren't any words in capitals on a line you can see immediately you've slipped up – by forgetting the keyword, or missing out a separator.

Right, armed with that advice, let's return to Program II.

This introduces another new idea, the use of the TAB( ) function. This allows you to specify how far along a line you want the output of a PRINT statement to start.

In Mode 1 there are 40 characters to a line, so the screen can be considered to be 40 columns wide. TAB( ) decides in which column the printout starts. The 40 columns are numbered 1 to 40.

When you change mode the number of characters across the screen – that is the number of columns – changes. For example, Mode 0 only supports 20 columns.

Try running the program in this mode by changing line 20 to:

```
20 MODE 0
```

Can you see what is happening?

After a while TAB( ) becomes second nature. All too often potentially good programs are spoiled because they are set out badly on the screen. Careful use of TAB( ) can avoid this.

To give you some practice, try Program V. This prints out a triangle of asterisks. Can you devise a similar program, using TAB( ), to create a diamond of asterisks in the centre of the screen?

Before you continue, you might find it easier on the eyes if you return to Mode 1 with:

```
MODE 1 [ENTER]
```

So far we have talked about string variables. However there is another kind of variable called a numeric variable.

These are labels just as much as string variables are, only they label

```
10 REM PROGRAM V
20 MODE 1
30 PRINT
40 PRINT TAB(5)"*"
50 PRINT TAB(4)"**"
60 PRINT TAB(3)"***"
70 PRINT TAB(1)"*****"
```

Program V

numbers in such a fashion that we can do sums with them. Try running Program VI.

Line 30 uses the numeric variable *A* to label the number 10. Notice that for a numeric variable we can simply use a letter of the alphabet without following it with the $ sign necessary for a string.

Also since it isn't a string, the value we are giving the variable doesn't have to be in quotes. Hence line 30 is simply:

```
30 A = 10
```

Line 40 prints out, not *A*, of course, but the value that *A* labels, which is 10.

The most interesting part is line 50. Here we multiply the number that *A* labels by two, so that the line prints out 20.

That's the useful thing about numeric variables – you can do sums with them!

Notice that the micro did the calculation, then printed out the result. It didn't do anything wild such as printing out 2 * 10 or whatever.

A sum such as 2 * A or A + 8 is known as a numeric expression. When it encounters a numeric expression, the micro works it out and prints the answer, rather than printing the expression itself.

Try running Program VI with the following versions of line 50:

```
50 PRINT A + 8
50 PRINT A / 4
50 PRINT A * A
```

If you've been following what I've said so far you could be forgiven for thinking that string variables are for labelling words, and numeric variables for numbers.

```
10 REM PROGRAM VI
20 MODE 1
30 A = 10
40 PRINT A
50 PRINT 2 * A
```

Program VI

Life is never that simple. You can, and often do, use string variables for labelling numbers – the point is that you can't do sums with them. Try Program VII, which is based on Program VI, using the string *A$* instead of the numeric *A*.

The "Type mismatch in 50" that you receive shows that you are attempting to do a sum with the wrong type of variable – string instead of numeric.

```
10 REM PROGRAM VII
20 MODE 1
30 A$ = "10"
40 PRINT A$
50 PRINT 2 * A$
```

Program VII

As with string variables, we do not have to (and should not) restrict ourselves to single-letter labels for numeric variables.

We can use words in a manner strictly analogous to string variables, save that we omit the final $ sign. And, of course, we don't put what we are labelling in quotes, since it isn't a string.

Again, capitals and lower case are considered to be identical so *A* is the same as *a*.

Have a look at Program VIII. This is meant to be a cheery greeting for someone when they RUN the program in the computer – the sort of thing I often use in my classes.

```
10 REM PROGRAM VIII
20 MODE 1
30 name$ = "Mike"
40 PRINT "Good to see you, " name$
```

Program VIII

However as it stands it's a bit restricted – after all, only a small percentage of my students are called MIKE. What's really needed is some way for the Amstrad to find out the name of the person so that it can tailor the message to suit.

Program IX fits the bill. The trick here is the use of INPUT name$ in line 40. In Program VIII, line 30 put the value MIKE into *name$*. In Program IX the variable isn't actually attached to a specific value – if you like, you give the program a label, but neglect to tell it what it's labelling.

Instead you type:

```
INPUT name$
```

When the Amstrad reaches this line it waits until you PUT IN, or INPUT, the value you want *name$* to have by typing the value in.

To put it another way, when the computer meets an INPUT statement

```
10 REM PROGRAM IX
20 MODE 1
30 PRINT "What is your name";
40 INPUT name$
50 PRINT "Good to see you, " name$
```

Program IX

followed by a variable, it asks you what you want the variable to be – in fact, it actually puts a question mark on the screen.

You are then supposed to type in the answer followed by Enter, which, as always, sends it to the computer, which then carries on with the rest of the program.

So when you run the above program line 30 asks: "What is your name". Notice that we don't need a question mark – the INPUT statement of line 40 supplies that.

The micro then waits for us to type our reply and send it by pressing Enter. Whatever we have typed in then becomes the value of *name$* – even if we have lied!

Line 50 then prints out the message.

The point of all this is that in Program IX, as opposed to Program VIII, the value of *name$* is not fixed initially, but is decided during the program by the response to INPUT.

This means that every student in the class can now run the program and have the message tailored to themselves.

Incidentally, line 30 is not strictly necessary, but it is only polite to tell people what kind of response you expect them to make. Otherwise they will be met with just a question mark, followed by a cursor – not too "user-friendly" as the jargon has it.

The semi-colon at the end of line 30 "glues" the question mark, or prompt, as it is known, to the preceding "message". Running the program with it omitted should make this clear.

Remember, when you run Program IX and it asks for your name, you must type your reply then press Enter. If you omit Enter the Amstrad won't

receive your answer and will continue waiting. This could be incredibly boring!

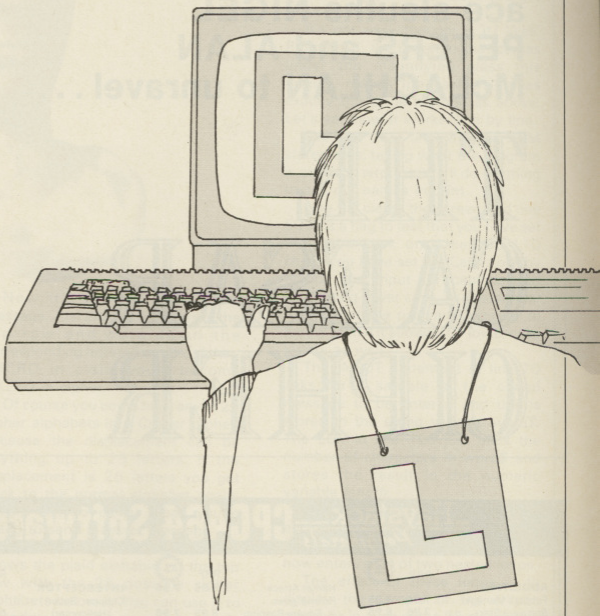
If you make a typing mistake before you press Enter, you can erase it with Delete. Once you've pressed Enter, though, you're stuck with what you've typed.

You can use INPUT with numeric variables as well as strings. Program X demonstrates this. When you get the prompt, try typing in a word rather than a number and see what happens.

A slightly more serious application of INPUT allows you to calculate the product of two numbers, as Program XI demonstrates.

```
10 REM PROGRAM X
20 MODE 1
30 PRINT "How old are you";
40 INPUT age
50 PRINT "I don't believe you are "; a
ge
```

Program X



Look carefully at line 70 and see if you can work out what's happening. *first* isn't in quotes, and so the micro will print the number that *first* labels. "Multiplied by" is printed literally since it is in quotes.

The numeric variable *second* is not in quotes – it may have them on either side, but the quotes on the left are already paired with the quotes on the far left, so they don't count. The micro will, therefore, print out the value of *second*.

"Is" is printed literally, since it is in quotes. *first***second* isn't in quotes, so the sum is done and the answer printed out. Figure I should help to make this clearer.

Finally, try altering Program XI so that it adds or subtracts pairs of numbers.

We've covered an enormous amount of ground here and I suggest that you spend a good while going over the programs. If you are having problems, re-reading the earlier articles will probably help.

Above all, remember it's a "hands-on" course – you can't expect the examples to make sense until you've typed them in!

```
10 REM PROGRAM XI
20 MODE 1
30 PRINT "First number";
40 INPUT first
50 PRINT "Second number";
60 INPUT second
70 PRINT first " multiplied by " secon
d " gives " first* second
```

Program XI

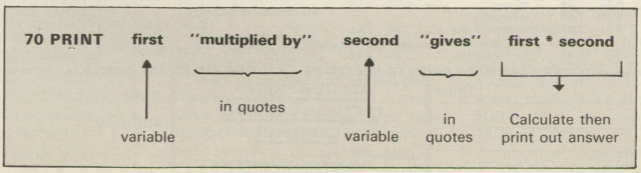


Figure I: Mixing variables and strings in PRINT statements

Tag along with those ace sleuths NIGEL PETERS and ALAN McLACHLAN to unravel . . .

THE CAESAR CIPHER

IN this article we'll be using the Amstrad to help unravel one of the best-known ciphers in the world – the Caesar Cipher. This method of writing secret messages got its name from its inventor, Julius Caesar.

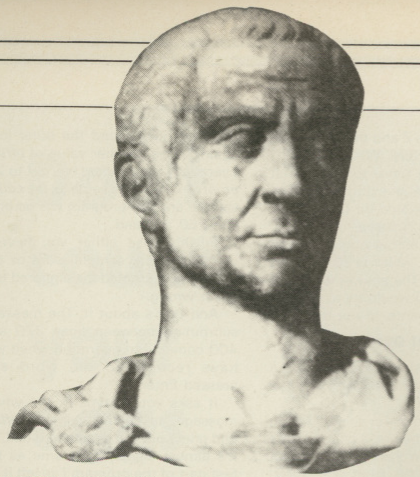
It works by displacing each letter of the message a certain number of places along the alphabet. If that sounds complicated, don't worry too much, it isn't.

All it means is that if I wanted to encode the word AMSTRAD I would first decide on what the shift would be. Suppose I wanted it to be two letters. Then the encoded word would be COUVTCF.

This is because C is two letters along the alphabet from A, O is two letters along the alphabet from M, and so on.

Of course we could have made the shift four letters long – in which case AMSTRAD becomes EQWXVEH – or any other number up to 25.

The simple way to write a message displacement and write out two



in Caesar Cipher is to decide on the alphabets as in Figure I.

```
Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:CDEFGHIJKLMNOPQRSTUVWXYZAB
```

Figure I: Alphabet displacement

Here the displacement is two letters. The top line, or plain, is the normal alphabet. The bottom line, or cipher, is the alphabet displaced by two letters. Notice that when you get past Z the next letters are A and B as the alphabet wraps around.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
FGHIJKLMNOPQRSTUVWXYZABCDE
GHIJKLMNOPQRSTUVWXYZABCDEF
HIJKLMNOPQRSTUVWXYZABCDEFG
IJKLMNOPQRSTUVWXYZABCDEFGH
JKLMNOPQRSTUVWXYZABCDEFGHI
KLMNOPQRSTUVWXYZABCDEFGHIJ
LMNOPQRSTUVWXYZABCDEFGHIJK
MNOPQRSTUVWXYZABCDEFGHIJKL
NOPQRSTUVWXYZABCDEFGHIJKLM
OPQRSTUVWXYZABCDEFGHIJKLMN
PQRSTUVWXYZABCDEFGHIJKLMNO
QRSTUVWXYZABCDEFGHIJKLMNOP
RSTUVWXYZABCDEFGHIJKLMNOPQ
STUVWXYZABCDEFGHIJKLMNOPQR
TUVWXYZABCDEFGHIJKLMNOPQRS
UVWXYZABCDEFGHIJKLMNOPQRST
VWXYZABCDEFGHIJKLMNOPQRSTU
WXYZABCDEFGHIJKLMNOPQRSTUV
XYZABCDEFGHIJKLMNOPQRSTUVW
YZABCDEFGHIJKLMNOPQRSTUVWX
ZABCDEFGHIJKLMNOPQRSTUVWXY
```

Figure II: All the alphabets

Now it's easy to write your message. Just look up its letters in the top row and note down the corresponding letters from the cipher. MICRO in plain would become OKETQ in cipher.

Of course you could have any of 25 cipher alphabets in a Caesar Cipher, because the displacement can be anything up to 25 letters. If the displacement is 26 letters you get back to the normal alphabet again, which isn't so clever when you're trying to send a secret message.

Figure II – a sort of super Figure I – shows the plain alphabet on the top row with the 25 possible cipher alphabets below it. You can use it to encode your cryptic messages in Caesar Cipher with the displacement of your choice.

If you want the micro version of Figure II then Program I will produce it for you.

```
10 abet$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 PRINT abet$
30 FOR loop= 1 TO 25
40 abet$=RIGHT$(abet$,25)+LEFT$(abet$
,1)
50 PRINT abet$
60 NEXT loop
```

Program I

But what, you may ask, has this to do with the Amstrad?

Well, try decoding a Caesar Cipher message such as LTAA SDCT. It's not easy if you don't know what the displacement is. You could spend ages searching through the table in Figure II trying out displacement after displacement until the message makes some kind of sense.

Have a go at figuring out LTAA SDCT using the table. Have you got it? Well done!

It's not easy though, is it? Well, Program II, the program that comes under the magnifying glass this month, makes life easier. It uses exactly the same method as above. Let's see how it works, line by line.

The first three lines are just REM statements, telling what the program is and who wrote it. We'll do anything to see our names in print.

Line 40 clears the screen and line 50 sets a flag to test that you have set the Caps Lock on. Lines 80-110 check you have set the Caps Lock by testing the input for capitals and preventing lower case. The reason for this is that the program is easier to follow than if it allowed for lower case letters as well.

The INPUT statement of line 70 asks for a sample of the coded message to be entered and this is stored in the string variable *word$*. The LEN in line 120 just finds the number of characters in *word$* and stores the result in the numeric variable *length*.

Having taken a sample of the enciphered message and measured the length of the sample, the program now enters a set of two nested loops.

The effect of these loops is to display the sample of the cryptic message in each of the 25 alternative alphabets available. It prints out each of these alternatives along with the number of letters that it is offset.

If a sample actually makes some sort of sense then you press Enter and get the chance to have the Amstrad decode the whole message. Otherwise pressing any other key allows the Amstrad to continue printing out the alternative versions of the sample until eventually one makes sense.

Lines 130 and 250 form a FOR . . . NEXT loop with the control variable *offset*. This ranges in value from one to 26 – once for every letter of the alphabet.

Each time round this outer loop, line 140 sets up a string variable *empty$* and sets it to the null, or empty, string. This is used in the inner FOR . . . NEXT loop formed by lines 150 and 200.

This inner loop takes the sample word apart, letter by letter, and rewrites it in a new alphabet with the ▶

letters offset by the variable *offset*.

Since *offset* is the control variable of the outer loop and varies from one to 26, this means that the inner loop rewrites the sample word with each of the 26 available alphabets. If the person who coded the message has done his job properly, then one of them must make sense.

This inner loop has the control variable *slice* which varies from one to the value of *length*. We met *length* in line 120. It holds the number of characters in the sample word. This means that the loop cycles once for each letter in *word$*.

Line 160 looks complicated but isn't all that hard when you take it bit by bit. The function MID$ takes one letter from the string *word$*. Which letter it takes depends on the value of *slice* which in turn depends on the stage the loop has reached.

Since *slice* varies from one to the length of *word$* this means that every letter of the sample word is selected in turn.

When MID$ selects a letter the ASC function in front of it gives the Ascii value of that letter. The Ascii value is just a number that represents a letter. A is 65, B is 66, and so on until Z is 90. So, as the loop cycles, each letter of the sample is turned into a number which represents it.

Line 160 doesn't stop there, however. It also adds the value of *offset* to the Ascii code for that letter and stores the result in the variable *lettercode*. Depending on how many times the outer loop has cycled, *offset* will vary between one and 26 in value.

What this means is that when the inner loop has finished, the Ascii code for each letter of the sample has been increased by the same offset. All that is needed to see the new word produced from the sample is to take each of these codes in turn and find out what letter they stand for.

This is what line 180 has been doing. CHR$ produces the letter for that particular value of *lettercode* and stores it in the string *letter$*. Each time round the inner loop *letter$* is added to *empty$*. When the loop stops, *empty$* holds all the letters of the original sample, *offset* letters along the alphabet.

But what, you might be wondering, does line 170 do? The answer is that it allows for the wrap around in the alphabet that we saw earlier.

Suppose the letter we were working on was Z and the offset was to be three letters. Well you and I would have the sense to realise that we go back to A and start again. The required letter would be C.

If we didn't have line 170, however, the Amstrad would take the Ascii code for Z, which is 90, and add the offset to it. The answer would be 93 and line 180 would try to find out what 93 represents. If you try:

```
PRINT CHR$(93)
```

on your Amstrad you'll see that it is a square bracket, not the letter C that it should be.

Line 170 allows for this by taking away 26 from the value of *lettercode* if it's over 90. In this case, the result of subtracting 26 from 93 is 67, which is the Ascii code for C.

Once the inner loop is finished the Amstrad goes onto line 210 which displays the new version of the sample word and the offset that produced it. The next two lines cause the program to halt until you press a key. This allows you to look at *empty$* to see if it makes any kind of sense.

When you press a key, that letter is placed in the string *a$*. The next line examines *a$* and if you pressed Enter – which you do when *empty$* makes some kind of sense – it takes you off to the subroutine starting at line 270.

If you didn't press Enter then the Amstrad goes round the outer loop again, trying another value of *offset*. Eventually when *offset* has got to 26, *empty$* will be the same as the coded sample and the program will go onto line 260 and end.

In this case either you haven't recognised the sample word or the person who created the ciphered text got it wrong.

And that's about it. The message subroutine between lines 270 and 400 comes into operation when you have recognised the word and pressed Enter.

It asks you to type in the whole message, takes the value of *offset* and prints out the deciphered text. As it's almost exactly the same as the first part of the program I've left it for you to try and figure out.

There are only two real differences. The first is a new variable *size* in line 300, to hold the length of *cipher$*. The other is line 340 which just allows for the fact that the coded message might have spaces (Ascii code 32) in it. After all, there's no point in adding the offset to a space is there?

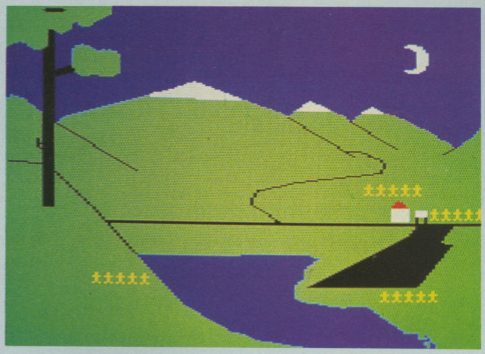
When you've figured it out, you might also be able to see that you could use the subroutine to translate your secret messages into Caesar Cipher.

WPKT UJC.

```
10 REM CAESAR CIPHER
20 REM NIGEL PETERS
30 REM AND ALAN MCLACHLAN
40 CLS
50 flag=0
60 PRINT"Set caps lock on and"
70 INPUT "enter test word.....",wor
d$
80 FOR loop = 1 TO LEN(word$)
90 IF ASC(MID$(word$,loop,1))>90 THEN
 flag=1
100 NEXT
110 IF flag =1 THEN PRINT CHR$(7): GO
TO 40
120 length=LEN(word$)
130 FOR offset = 1 TO 26
140 empty$=""
150 FOR slice= 1 TO length
160 lettercode = ASC(MID$(word$,slice
,1))+offset
170 IF lettercode)90 THEN lettercode=
lettercode-26
180 letter$=CHR$(lettercode)
190 empty$=empty$+letter$
200 NEXT slice
210 PRINT empty$,offset
220 a$=""
230 WHILE a$="":a$=INKEY$:WEND
240 IF a$=CHR$(13) THEN GOSUB 270:END
250 NEXT
260 END
270 REM*********message**********
280 PRINT:PRINT
290 INPUT"Cipher....",cipher$
300 size=LEN(cipher$)
310 message$=""
320 FOR slice =1 TO size
330 lettercode=ASC(MID$(cipher$,slice
,1))
340 IF lettercode()32 THEN lettercode
=lettercode +offset
350 IF lettercode)90 THEN lettercode=
lettercode -26
360 letter$=CHR$(lettercode)
370 message$=message$+letter$
380 NEXT slice
390 PRINT message$
400 RETURN
```

Program II

Don't count your eggs



I WAS lucky enough to be able to spend two weeks in Scotland last summer and the highlight of the trip was my visit to see the nesting ospreys at Loch Garten.

So when **Osprey!** from Bourne Educational Software arrived in the office, I grabbed it with enthusiasm. I wasn't disappointed.

Produced in conjunction with the RSPB, and with an excellent 32 page colour booklet to complement the program, it's a fascinating simulation of the problems faced by the osprey as a Scottish breeding bird.

You are the manager of a nature reserve where ospreys are nesting. The booklet has given you an outline of the bird's history and you have to pick which year you want the simulation to start. The earlier the year, the harder the game.

Your aim is to make sure the birds successfully breed and rear their chicks. To do this you have to decide what your wardens are going to do during the vital spring and summer seasons.

Some are needed to chase away egg stealers while others have to manage the site and keep disturbance from visitors to a minimum. Also wardens have to be spared to make people aware of the ospreys and to encourage public support.

And, just like real life, when you've made your choices and allocated your resources you sit back and watch what happens.

The graphics are beautiful, painting a picture of the reserve and the nest site. You can watch as the ospreys swoop down to fish and take them to the nest. Sadly, if you haven't allocated enough wardens to guard duty, you can also watch the egg thieves at work.

Even the visitors can be a nuisance, their cars disturbing the birds if you haven't picked the right number of site wardens. And to make it worse factors totally out of your control such as the weather affect the final result.

Your success or failure at one reserve is taken as representative of the whole of Scotland and after the spring season you are shown how the ospreys population has fared under your protection.

You continue until you reach the year 1981 or you've run out of ospreys – horrible thought. You can then compare your efforts with the magnificent results the RSPB achieved in reality which are shown in the booklet.

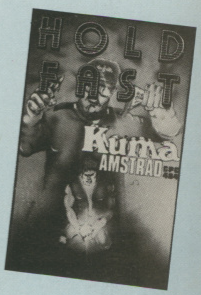
It's a smashing program. The instructions are excellent. The graphics and animation are more than adequate and the whole package has the quality we've come to expect from Bourne.

Even the fact that it's educational – the well-illustrated booklet has a history of the osprey and a "Things to Do" – doesn't spoil the fun.

So if you haven't been to Loch Garten yet you can console yourself playing Osprey! until you get the chance. And even when you've felt the magic of the ospreys in real life, you'll still want to play it. Recommended.

Nigel Peters

Protest... and survive



HOLDFAST BY Kuma Computers is like nothing I have played before. It is fascinating and I have become totally absorbed in it.

The program is a simulation of non-violent resistance, and is a strategy game involving a great deal of thought.

You are cast as an influential member of the village of Holdfast in Dictatoria. For many years the village has paid its taxes but the government has refused to grant any aid for development.

The village is in desperate need of a clinic and school, but there is a crisis on Dictatoria's border and the government says that it cannot afford to provide these because of the ever increasing defence budget.

Your task is to persuade the government – by non-violent means – to grant the aid so that the school and clinic can be built.

Many options are available at each stage. You can organise protest marches to the capital, picket the local government offices, picket the local military base (and be harassed by the police), stage a sit in at the mayor's office, refuse to pay your taxes and hold press conferences. And this is just for starters.

Every now and then there are newsflashes and press reports such as "US holiday makers tell of conditions in

Holdfast" or "Local trade unions declare solidarity".

The game is menu driven. The initial one presented on loading gives you the option of negotiating with the government, various methods of indirect action, no further action at present or to seek advice.

At each stage you are required to make a selection from a menu, which often leads to another menu from which you can decide on the action to be taken.

The menu can change depending on what action you have chosen and with some, an option may only be chosen once so must be used at the correct stage of the game.

This is one game that I shall be playing for a long while. After several hours I still have not managed to wear down the government.

Holdfast is a well written and thought provoking simulation. Adventurers would find it quite interesting and it could also be a valuable educational program used to stimulate discussion of politics and local government.

Roland Waddilove

Colossal Adventure – the BIG one

COLOSSAL Adventure is Level 9's version of the original Crowther and Woods Colossal Caves and what a version it is. It is virtually identical to the original, but has an extra 70 rooms in the end game.

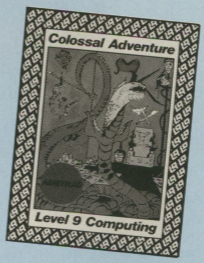
This is no mean feat when you consider that Colossal Caves was written on a mainframe.

Your task is to explore the countryside and enter the massive underground cave system, solving puzzles and collecting treasures along the way.

Simply put, but not so simple to accomplish. You can expect to take weeks rather than days to solve this adventure.

You begin on a road outside a cabin which contains several useful objects. A quick survey of the surrounding area is essential, but don't succumb to hunger too soon – you'll need to grin and BEAR it.

Now you should be ready to tackle the underground cave



system. Your first problem will be in getting past the snake.

Remember the old adage about a bird in the hand. You should start finding the first of your treasures about now and also have found a magic word that will speed you back to the cabin – a suitable place for storing things.

Try to be careful with your torch. You won't last very long if your batteries go flat. Talking of which, you'll have to be good at mapping mazes to find a replacement battery, but don't forget to take some money with you.

You should find most of the locations accessible but you will have to find all of the treasures to enter the end game and solve all the puzzles.

The biggest problem you are likely to encounter is Witt's End. Once you enter every move will bring you back to where you started.

Try making the same move a few times if you want to get out. Incidentally, it's a good idea to take on the role of paperboy about now.

You will find quite a few mazes, all of which can be mapped – if, that is, you can find the right way of mapping them.

One final tip. Don't waste any time on the shadowy figure – it's your reflection in a mirror.

I have one slight niggle. Level 9 has used its own character set which isn't very readable and would probably have benefited from a different background colour.

It's difficult finding new superlatives to describe Level 9's adventures. I have no hesitation in recommending this as one of the best I have ever seen.

Paul Gardener

Really friendly assembler

THE majority of programmers, myself included, start off by assembling their first few machine code programs by hand.

This is fine for short, simple routines, but there comes a stage when the time spent assembling the code is simply unacceptable and some sort of tool is required to speed up program development.

If you have reached that stage then you will be taking a close look at the various assemblers available for the Amstrad. Look no further, Arnor's **Z80 Assembler, Disassembler and Editor** is the best I have seen.

All assemblers have pretty much the same range of functions, and they come complete with an editor, disassembler and memory lister.

The differences between them lie more in their ease of use and user friendliness.

Arnor's is by far the easiest to use and most friendly that I have come across.

There are two different methods of entering assembly language mnemonics. The first is using the editor.

Typing in assembler using it is very similar to using a word processor and just as easy. Simply type in the mnemonics, labels and comments as you would a letter or document.

The cursor keys enable you to move to any position in the text and Delete/Clr work as normal.

As with a word processor there are search and replace, and block move or delete facilities.

Whole lines can be deleted or space made for a new line, and the cursor can be sent to the start or end of each line.

The biggest advantage of this assembler is the ability to mix assembly language and Basic. ASSEMBLE has been added as an additional command to Basic:

Each line of source code must be preceeded by a ' which is interpreted by Basic as a REM statement. However if ASSEMBLE is placed at the start then the assembler takes over, reading each line and compiling it.

This means that the code can be quickly and easily altered using Basic's editor, and loading and saving involves fewer steps.

The disassembler and memory lister is useful occasionally, either if you have lost the source code or if you want to have a look at someone else's program or the ROMs.

Roland Waddilove

On the ball

ROLLABALL is an unusual, original and quite clever game from Timeslip. But having said that, it is not the sort of game that I will play very often.

It's difficult to say why, but

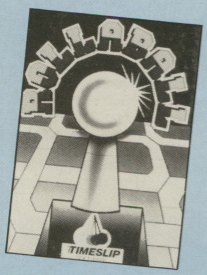
perhaps it is because it just does not have the addictive quality of an arcade game or the challenge of an adventure.

When loading has finished a pretty neat tune is played while you decide whether to read the instructions or not.

The tune is the best part of the program and once or twice I loaded Rollaball just to listen to it and didn't bother to play the game.

The instructions are poorly displayed and should be altered. They are printed in a small box in the middle of the screen, all in upper case and without any punctuation whatsoever. This is surprising as the address for Timeslip is The Old Primary School in Stoneyburn, West Lothian.

The game is rather difficult to describe. Try to imagine a plate of spaghetti with the strands all tangled up. Well



instead of spaghetti it is a track along which a ball rolls.

The object is to keep the ball moving along the track and away from the border and any holes for as long as possible.

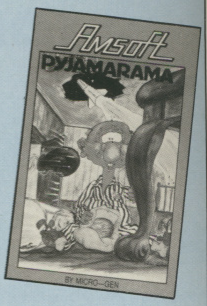
Sections can be slid around like in one of those plastic letter puzzles that children play with, so that the direction of the ball can be altered if it is heading for disaster.

It is quite difficult to keep the ball going, as you have to look as far ahead as possible along the track to see where it is heading and work out how to alter it to take the ball away from danger.

Rollaball is an unusual game with good sound, but I think you may tire of it rather quickly.

Roland Waddilove

This nightmare is a dream



NOW here's a game for the insomniacs among you.

Wally Week (sounds like a national holiday for burkes doesn't it?) is trapped in one of his own nightmares and the only way out is to try and wake himself up.

To do this he must head out into the nightmare-ridden house in search of his alarm clock.

Pyjamarama by Microgen is an arcade adventure game. The house has many rooms, most of which are riddled with nasty creatures or are apparently dead ends.

This is where the adventure aspect comes into play. Around the house are scattered literally dozens of objects. Using logical thought, which put me at a disadvantage from the start, you must deduce which of the objects should be carried into each of the rooms.

In this way new routes open up into the deeper recesses of the house.

In true adventure fashion, as you can only carry two items at a time you must remember where you last saw a certain item and then hope to find your way back to it.

Nightmares can be pretty tiring at times, so a constant display of Wally's energy is shown at the top of the screen – it's the level of liquid in a glass of Horlicks.

It decreases slowly during normal movement but fair gulps down when Wally is caught by the bouncing choppers or grasping hands, just two of the eye watering hazards that lie in wait.

The graphics are very Sinclair Spectrum, each character being large, crude and of only one colour. This didn't detract too much from the game's appeal as the characters' response to the keys or joystick was quick and accurate.

Another point I liked was that death didn't always come within the first two minutes of the game. It normally took me ten or fifteen minutes to fail successfully.

If you wait long enough when presented with the title page the program will enter its demo mode. Ah ha, I thought, now I can find out how to get past the Venus fly traps.

On your bike, replied the program as Wally died in the demo as he had under my control.

Scores are displayed as a percentage once you have finished the game. My score was on a par with Wally's IQ. It looks like it's going to be a long night, Wally.

Carol Barrow

Monty to the rescue

WITHOUT extra terrestial beings I don't know what programmers would find to write about. In **Mutant Monty** by Artic Computing, the evil aliens have captured a fair maiden and are holding her hostage.

All you have to do to rescue her is battle your way through to screen number 40. Just the game for that spare five minutes you might have.

Hero of the piece is called Monty and is a bit of a Barry Manilow lookalike. I believe that the mutant prefix in the title refers to the aliens.

The screens are a mixture of caves, mazes and buildings, each divided into a series of corridors. Along these march a multitude of vicious aliens, some drawn in great detail.

Monty's loyalties are divided between rescuing the fair maiden and collecting the gold nuggets that can be found on each screen.

However as the exit door to the next screen will not open until all of the gold has been collected you might as well get rich for your efforts.

This is one of those games in which you have to be in the right place at the right time or you will never complete the screen.

Each move has to be coordinated with the movement of the aliens. You may have to attempt a screen five or six times before you finish it, but once you have figured out the safe route you will have no more problems.

As you will soon become aware Monty can move at two different speeds. Press a key and release it and he will continue to travel in that direction.

Continue to hold the key down and his speed suddenly doubles – great for outrunning aliens, but lousy when you are quietly following one along a corridor.

Another feature for which I will be eternally grateful is the fact that Monty has five lives – the games tend to last that little bit longer.

I thoroughly enjoyed dodging my way through to screen five, yet I am certain that the princess will be old and grey before this particular prince comes to her rescue.

Jon Revis

A touch of the Cluedos

AS soon as I saw the picture on the cassette cover I thought "Al, this is right up your street". There's a picture of a lean, lanky police sergeant, about 6ft 14in carrying an enormous truncheon and a magnifying glass, stealthly plodding his size 10s in search of a villain – a murder suspect? I unravelled the package with bated breath.

My excitement was short lived. The theme of **Detective** by Argus Press is who dunnit – simply a micro version of the table top favourite Cluedo, itself an extremely sophisticated Mastermind.

However a very good job has been made of it, catering for up to six players.

Your task, of course, is to use tact and deduction to find the correct combination of four unknowns – not letters, numbers, or coloured pegs, but Who dunnit? Where? Why? and What with?

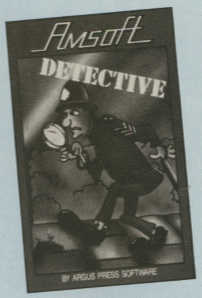
The scene is an English country house where the butler's body has been found dumped on the stairs.

Each player adopts the role of one of the game's nine characters and must discover who has committed the foul deed.

The names are entertaining, with Sid Weasel an obvious suspect before you start, Boris Goddenuv a shade on the dodgy side and Dolores Stardust a right little raver.

Once you've chosen your character it is displayed as a small graphic face so that you can follow it through the game.

Each player is given a set of secret number codes for the culprits, weapons and motives. This has been arranged so that when you make your guess (sorry, deduction) no one else can follow your



logic.

The other players are asked to look away while each set of codes are displayed.

The playing board then appears as a plan of the house and gardens – seven internal locations and two outside.

The computer dice is thrown by pressing a key and each player can move any number of places up to the maximum on the dice. Once your little graphic character is at a location you make an accusation using your codes.

Success or otherwise is indicated by a series of Ys and Ns representing the four unknowns in order. On a second level of difficulty the Ys only are shown in no specific order – much more difficult as there's no indication which you've guessed right.

You move around the locations making your deductions until either someone arrives at the correct solution or the maximum number of turns – 10 on level 1, 15 on level 2 – are completed.

The graphics are well handled and colourful. The sound is a bit tedious – each turn is heralded by a wailing police siren – but the volume can be turned down.

The game isn't going to be a chart-buster but it is fun and should be popular with younger children.

Alan McLachlan

Scrolling along to success

AS an arcade game Scramble never made my ratings, as I always came a cropper when tackling the second screen – the one with no missiles but all those bouncing balls in confined spaces.

Star Avenger from Kuma is a Scramble game, but it is a vast improvement upon the original.

You must pilot your ship over 10 different regions of the alien planet's surface. Some will be familiar to seasoned Scramblers, yet some form new and testing scenarios.

Screen nine, the Fortress, is one such. Apart from the standard barrage of missiles there are numerous anti-aircraft gun emplacements.

These swivel around taking pot shots in your direction, and the shells explode star burst fashion, destroying anything within range.

Technically Star Avenger is the best micro Scramble that I have played, as the sideways scrolling action is faultless. There is however one omission – there are no bombs to be dropped.

As the control keys are not mentioned in the sparse instructions I spent the first three games figuring out which keys controlled the ship.

Desperately I looked for the bomb toggle, but without success. It was at this point that I took a breather. The next thing I knew – a demonstration mode was up and running.

The very first part of the demo mode was a description of the control keys, and sure enough there wasn't one for dropping bombs.

This therefore determined

future tactics, as every run had to be a low level ground hugger in order to take out the fuel dumps vital to your mission.

You are given the option of starting on any screen at any of the five skill levels.

Should any screen prove too difficult just refer to the demo mode.

Star Avenger is the first Scramble clone to be written for the Amstrad and I have a funny feeling that it will be the last.

Jon Revis

Step out with Roland

ROLAND Goes Square Bashing is the latest from Durell continuing the saga of our little friend.

In a Qbert-type variation the screen consists of a 3D series of steps up or down at random in four different directions. User definable keys allow you to move in any of these directions in order to access each of the steps in turn.

Looking like a mutant sugar cube wearing Doc Martins, our intrepid hero drops in on the first screen of 20.

Immediately the square on which he is standing begins to

shrink and you have to move him immediately to a neighbouring one.

Each square you move to begins to shrink and you have to be nimble to move onto the next, keeping your eyes open for a route to cover all the steps.

Your aim is not to be left stranded on an isolated step, as it slowly shrinks to nothing leaving you to fall to your death.

The first sheet is relatively easy – it even says so at the bottom of the screen – but succeeding ones get progressively more difficult.

Once you find your way around a particular screen you should never have another problem with it, as it doesn't change. This is offset by the fact that you only have one life. Therefore one moment of indecision and your game is over.

However the game does have a useful facility for moving to any screen of your choice to practice and work out the best route.

To add to your frustrations sarcastic comments appear at the bottom of each screen before you start. Examples: "Chicken feed", "Up and down we go" and "I don't get it mum".

I lost interest in the earlier screens once I'd mastered them and was soon using the facility to skip to harder more testing levels.

The keys seemed a little unresponsive at times and this resulted in my getting stuck on shrinking squares quite often. The "one life" situation really was a pain sometimes.

The game was infuriatingly addictive. The 3D graphics were excellent and the sound acceptable, but I don't think it will take the market by storm, as there was too little screen variety.

Dave Mac

Wong's on the run

CHAOS reigns in **Mr Wong's Loopy Laundry** (Artic Computing). As a result of some paranormal influence, the once inanimate objects scattered around have developed minds of their own and are out to get their owner.

Will your quick wits keep Mr Wong alive, or is he all washed up?

The dirty laundry can be found on each of the four floors. To the strains of oriental music Mr Wong must run from floor to floor via the stairs, collecting each item of clothing in turn and placing it in the chute on the top floor.

Our intrepid Chinaman's progress is hindered by a steam iron, a laundry bag, and most fearsome of all, some soap suds.

For his defence, Mr Wong has only his agility and the occasional box of starch to call upon.

When you are in a desperate situation, which tends to be quite often, the starch can be thrown at your attacker causing temporary paralysis and allowing you to make your escape.

There are however two warnings that must accompany the use of the starch. Firstly you are only allowed five puffs per box, and secondly you cannot fire another puff while the first one is still affecting a baddie.

Fortunately new boxes appear at random intervals, but these must be collected quickly or they will vanish.

Graphically the screen layout is very reminiscent of the Burger Time/Mr Wimpey programs produced for the BBC Micro and Spectrum. The characters are large, colourful but not very detailed.

The game can be played with either joystick or keyboard, and herein lies my only complaint. Why oh why, when there are all of those keys do some programmers insist on using the cursor key cluster to move a character?

I struggled unnecessarily while attempting to complete a simple first screen merely through straining to cram five fingers into a two inch square.

When using a joystick Mr Wong is a pretty average program. Without the stick it is practically unplayable.

Jon Revis

REVIEWED SO FAR

American Football	APS
Amstradraw	B.S. McAlley
Astro Attack	Amsoft
Amstrad Tutorial	Amsoft
Atomsmasher	Romik
Blagger	Alligata
Colossal Adventure	Level 9
Country Cottages	Sterling
Cubit	Mr Micro
Detective	Argus Press
Ts*hert	Microbyte
Flight Path 737	Anirog
Football Manager	Addictive
Fruity Frank	Kuma
Forest at World's End	Interceptor
Galaxia	Kuma
Gems of Stradus	Kuma
Ghouls	Micro Power
Happy Letters	B.E.S.
Harrier Attack	Durrell
Happy Writing	B.E.S.
Holdfast	Kuma
Hunchback	Ocean
Map Rally	B.E.S.
Manic Miner	Software Projects
Message from Andromeda	Interceptor
Mr Wong's Loopy Laundry	Artic Computing
Mutant Monty	Artic Computing
Osprey	B.E.S.
Punchy	Mr Micro
Pyjamarama	Microgen
Roland in the Caves	Amsoft
Roland Goes Square Bashing	Durell
Rollaball	Timeslip
Royal Quest	Timeslip
Snooker	CDS
Snooker	Gem
Snowball	Level 9
Survivor	Anirog
Star Avenger	Kuma
Test Match	CRL
The Moors Challenge	Timeslip
Timeman One	B.E.S.
Timeman Two	B.E.S.
Wordhand	B.E.S.
Wordwise	B.E.S.
Xanagrams	Postern
Z80 Assembler, Disassembler and Editor	Arnor

Shoot out at the Abbey

DEADSTONE Abbey, the setting for **Survivor** from Anirog, has lain deserted for many years, yet surprisingly the rooms are strewn with bags of gold.

The reason's not hard to find – the gold has remained untouched – because nobody dares enter the place for two good reasons.

Firstly the monastic inhabitants were a little homicidal, and planted bombs in several of the rooms on each floor.

Secondly there are the many and varied ghoulish creatures who inhabit the abbey.

So why should Angus, hero of the piece, subject himself to such dangers? The answer is simple – he's skint.

Being one of the original high rise abbeys it has seven floors, access to each floor being dependant upon Angus disposing of every bomb upon his present level.

The little Scot enters the abbey possessing a pistol, 40 bullets and a limited supply of energy.

Fortunately both can be replenished. Bullets are provided by walking over the pistol symbols and energy can be found in the form of bottles of elixir.

Use of the pistol requires some quite nimble footwork as it only shoots in a horizontal plane. You often end up running rings around the ghouls in an attempt to shoot them.

They fall into several classes of nastiness. The most harmless are the amorphous blobs in cowboy hats who wander around aimlessly.

We also have the flying haggises. Both these and the lizards sit on the cruise missiles and are the next nastiest group of gremlins.

Finally we have the cyborgs, dead ringers for Daleks, but twice as nasty.

Using a combination of good shooting and plenty of the old elixir you can make a game last for 10 to 15 minutes. This combined with the excellent graphics ensure that Anirog has got a winner on its hands.

James Riddell

MIKE BIBBY takes the A train again in Part Four of his introduction to machine code and introduces loading bytes to and from memory locations

Mnemonics that make your code come to life

LAST month we saw that the Z80 has a special sort of memory location inside it called the A register, which can hold exactly one byte of data.

We saw how we could load it with any number in the range 0 to 255 with the instruction LD A which stands for LoaD the A register. For instance, to load the A register with &FF we would use the instruction:

```
LD A,&FF
```

The opcode for loading the A register with a number is &3E. We follow it immediately with the number we want loading into the A register. The above instruction would translate as:

```
3E FF
```

Let's have a look at a program, stored at &3000, that uses this opcode:

address	hex code	mnemonics
3000	3E 07	LD A,7
3002	CD 5A BB	CALL &BB5A
3005	C9	RET

If you have Hexer, the hexadecimal loader we listed last month, you'll be able to load this into memory with the enter code option. Otherwise, you can poke the code into memory with:

```
POKE &3000,&3E
POKE &3001,&7
```

and so on.

Once you run the program – from Hexer or with a CALL &3000 – you'll be rewarded with a beep. You've loaded the A register with 7, the Ascii code for a beep, and then called the firmware routine at &BB5A which prints out the Ascii character in the A register.

Try altering the program so that it prints out various letters of the alphabet.

Notice that the address of the routine goes in lo byte followed by hi byte. That is, 5A followed by BB. This is the standard practice on the Z80.

The routine at &BB5A is, in a way, a primitive machine code equivalent of PRINT – handling only one character at a time. Let's have a look at a routine that corresponds to INPUT in a similar way.

address	hex code	mnemonics
3000	CD 18 BB	CALL &BB18
3003	CD 5A BB	CALL &BB5A
3006	C9	RET

Here the routine at &BB18 waits for a key to be pressed, then stores its Ascii code in the A register. Note carefully, it's &BB18 *not* &BB1B. Don't mistake that last 8 for a B.

Having got the code in the A register, the program then calls the routine at &BB5A and prints the character out. If you run the program a few times you'll see that it indeed echoes onto the screen the key you've pressed.

So we've got two routines now, one to get a character into the A register and one to print it out.

We've already seen that, since the hexadecimal opcodes are a bit difficult for we humans to remember, we use mnemonics to simplify things. For example:

```
CALL &BB18
```

is far more readable than:

```
CD 18 BB
```

We can make things better still by giving our routines names, or labels as they're known. We could call the routine at &BB18 CharIn, for example, standing for Character In.

Similarly we could call the routine

The A register is rather more versatile than the others...

at &BB5A CharOut – no prizes for guessing why! The program above then becomes:

```
address  hex code   mnemonics
3000     CD 18 BB   CALL CharIn
3003     CD 5A BB   CALL CharOut
3006     C9         RET
```

Note that the actual code itself hasn't changed – it's still the same hex numbers. The mnemonic part, though, is much more readable.

When you've got a byte in the A register there are quite a lot of things you can do with it. Have a look at the following program:

```
address  hex code   mnemonics
3000     CD 18 BB   CALL CharIn
3003     3C         INC A
3004     CD 5A BB   CALL CharOut
3007     C9         RET
```

The instruction INC A is the interesting one. This takes the number in the A register and INCreases its value by one. Its opcode is &3C. What happens in the above program is that the micro waits for you to press a key. It then stores the Ascii code of that character in the A register.

Next the INC A increases the value of that Ascii number by one. Finally the character corresponding to the new, increased number is printed out on the screen.

So if you run the program and type A (Ascii code 65), a B will appear on your screen (Ascii code 66).

Why not try altering the above program so that, having printed out the character with the INCreased Ascii code, you then INCrease the A register yet again and print that out?

Having learned that there's an INC A instruction, you won't be surprised to find that there's a DEC A instruction that DECreases the A register by one. It's opcode is &3D.

It shouldn't be too hard to follow what the next program does:

```
address  hex code   mnemonics
3000     CD 18 BB   CALL CharIn
3003     3D         DEC A
3004     CD 5A BB   CALL CharOut
3007     C9         RET
```

Why don't you try using INC and DEC to give you ABC on the screen if you simply press the B key? You'll need to call CharIn, then decrease the A register, call CharOut, increase the A register and so on.

The Z80 has quite a few registers other than A, and we'll be meeting them next month. For the moment though, let's concentrate on the A register as it's rather more versatile than the others.

One thing we can use it for is to PEEK and POKE memory. You know that you can alter the contents of a memory location with POKE – that's how we loaded our first programs.

When you POKE you must specify two things:

- The address of the memory location you want altering.
- the value you wish to change it to.

Now a machine code equivalent of POKE would be:

```
LD (&2FF9),A
```

What this does is to take the number already in A and then load that number into the memory location specified.

As with POKE, we've needed two things:

- The address we wish to alter – in this case specified in brackets.
- The value we want it changing to – in this case it's supplied by the A register.

So:

```
LD (&2FF9),A
```

would read *load the memory location &2FF9 with the contents of the A register.* Its opcode is &32 and the hexadecimal translation of the above instruction would be:

```
32 F9 2F
```

Notice that we specify the location we want to load the A register into *after* the opcode in our usual lo byte, hi byte fashion.

The following program shows how we can use this idea:

```
address  hex code   mnemonics
3000     3E 20      LD A,&20
3002     32 F9 2F   LD (&2FF9),A
3005     C9         RET
```

If you run the program, then use Hexer to examine memory location &2FF9, you'll find that it has had &20 stored in it.

Just as we can POKE with the A register, so we can PEEK. The instruction:

```
LD A,(&2FF8)
```

will load the A register with the contents of memory location &2FF8, effectively PEEKing that location. It's opcode is &3A and the above instruction would translate into:

```
3A F8 2F
```

Notice once again the brackets around the memory location in the mnemonic. These are important – they mean that the instruction refers to the contents of the memory location specified within those brackets.

Here's a program that uses both peeking and poking:

```
address  hex code   mnemonics
3000     3A F8 2F   LD A,(&2FF8)
3003     32 F9 2F   LD (&2FF9),A
3006     C9         RET
```

This simply loads the contents of memory location &2FF8 into the A register, then loads memory location &2FF9 with the contents of the A register.

In effect what's happened is that we've copied the contents of memory location &2FF8 into &2FF9, via the A register.

Use Hexer to enter the code into

your micro, then enter two *different* numbers into locations &2FF8 and &2FF9.

Run your code, then use the Examine option – you'll find that the contents of &2FF9 have become the same as those of &2FF8.

Incidentally, this is why our Examine option defaults to &2FF8 and not &3000 where all our code has started. I'm going to use the locations &2FF8 to &2FFF to POKE the results of our machine code into, for examination after we've run our programs.

To use the jargon, I've reserved these bytes as program workspace.

So far we've used the INC instruction to increase the number in the A register by one. There is another way to do this:

```
address  hex code   mnemonics
 3000    CD 18 BB   CALL CharIn
 3003    C6 01      ADD A,1
 3005    CD 5A BB   CALL CharOut
 3008    C9         RET
```

The mnemonic ADD A, as its names implies, *adds* the single byte following the comma to the A register. The opcode for this is &C6. In fact the above program simply waits for a key press then prints out the character with the next higher Ascii code.

Notice that it takes one more byte than the version which just used INC A.

The advantage with ADD A, is that we aren't restricted to adding one to the A register. We can add any number we want in the range 0 to 255. Try altering the program so you add, say, 2 to the Ascii value of the key pressed and so on.

We can use our poking techniques to store the results of our addition sums:

```
address  hex code   mnemonics
 3000    3E 02      LD A,2
 3002    C6 02      ADD A,2
 3004    32 F8 2F   LD (&2FF8),A
 3007    C9         RET
```

Use Hexer's examine option on location &2FF8 to see if the micro knows its sums. If you stick to small numbers you'll find it does. However, bigger numbers can cause problems, as the following shows:

```
address  hex code   mnemonics
 3000    3E FF      LD A,&FF
 3002    C6 01      ADD A,1
 3004    32 F8 2F   LD (&2FF8),A
 3007    C9         RET
```

What we've done is to load the A register with 255 (&FF) and then add 1 to it. However if you run the code and then examine &2FF8, you'll find the value 0 stored there. What's going on?

The problem lies in the fact that the A register can only hold a single byte, which restricts it to numbers in the range 0 to 255. The answer to our sum is 256, which is too big to be stored in a single byte. So what the Z80 does is to start again from zero once it goes past 255.

So, as far as the A register's concerned:

255 + 1 = 0
255 + 2 = 1
255 + 3 = 2

and so on.

It's a bit like a car's odometer – once you go past 99,999 miles you start again at 0 miles.

I always visualise the numbers laid out in a circle, as in Figure I. Adding numbers takes you clockwise round the circle, subtracting numbers takes you anti-clockwise.

When you're subtracting, things go awry when you get an answer less than 0. This time, as you'll see from Figure I, "going past" 0 gives us 255, not minus 1.

Thus, as far as the A register is concerned:

0 – 1 = 255
0 – 2 = 254

and so on.

The following program shows what happens:

```
address  hex code   mnemonics
 3000    3E 00      LD A,0
 3002    D6 01      SUB A,01
 3004    32 F8 2F   LD (&2FF8),A
 3007    C9         RET
```

We've used a new instruction, SUB A, which allows us to subtract the byte immediately following its opcode from the A register. Its opcode is &D6, so to subtract &7F from the A register, the hex code would be:

D6 7F

The previous program attempts to take 1 away from zero. If you run the code and then examine &2FF8, where the result of the subtraction is stored, you'll see that it has the value 255 (&FF).

You're going to get some pretty alarming answers to your sums unless you take this behaviour into account. Fortunately the Z80 has a warning signal to tell you something odd has occurred. It's called the carry flag. (You'll see exactly why in a later article.)

Flags are very important to machine code programmers. We use them to check the state of our program – in much the same way as a doctor uses your pulse, blood pressure and so on as aids to his diagnosis.

However the numbers associated with flags don't vary as much as pulses and temperatures. A flag can have only two values: 0 and 1. (a nice two-state, or binary, system!)

When a flag has the value 1 we say that it is *set*. When it has the value 0 we say that it is *clear*.

The Z80 has several of these flags to help us keep track of what is going on, but for the moment we'll concentrate on the carry flag. Now the Z80 uses the carry flag to warn us when our sums are going awry in the manner we've just seen. That is, when we're:

- Using ADD A and getting a result bigger than 255.
- Or using SUB A and getting a result less than 0.

In both of these cases the Z80 automatically sets the carry flag – that is, it gives the carry flag the value 1.

In fact every time you do an ADD A or a SUB A the Z80 adjusts the carry flag. If the sum has stayed "within bounds" – that is, hasn't gone over 255 or under 0, the carry flag is given the value 0. And, as we've said, if it does infringe the limits, the carry flag is set.

Then, depending on the condition of the flag, we can take appropriate action.

We use them a bit like Basic's IF

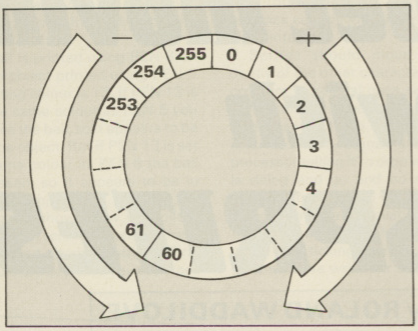


Figure I: How the numbers cycle in a byte

statement. For example, *IF the flag is not set THEN do this* and so on.

However we don't tend to look directly at the flags and then decide what to do – we use instructions that automatically take the states of the flags into account.

To illustrate the use of flags we'll have to jump ahead a bit (no pun intended). Have a look at the following piece of code:

```
address  hex code    mnemonics
3000     C3 00 30    JP &3000
```

Now this is an exceptionally silly piece of code. In fact it's the machine code equivalent of:

```
10 GOTO 10
```

except that if you run the code you'll have to reset your machine to get out of the endless loop.

You see, the JP opcode (&C3) tells the micro to continue the program from the address given in the two bytes immediately after that opcode (in lo byte, hi byte order). If you like, it tells the Z80 to JumP to that address and carry on from there. So:

```
C3 00 30
```

would tell the micro to go and do the program starting at memory location &3000. However if you look at the program you'll see that it itself is stored at &3000. So it goes and does itself again. Then of course it goes and does itself again – and so on ad infinitum.

This type of jump is called an *unconditional* jump. That is, as soon as the micro gets to the jump instruction it does it.

There are ways, though, you can give the micro a "choice". The Z80 has a set of *conditional* jumps. Whether or not the micro does the jump depends on the state of the flags.

The one we're interested in at the moment is the JP NC instruction (opcode &D2).

JP NC stands for *JumP if Not Carry*. That is, jump to the address specified if the carry flag is not set, that is, it's clear. If the carry flag is set, however, the micro doesn't perform the jump – it simply continues, performing the next instruction.

We can use this instruction to create a loop, as the next program illustrates. It's our most complex program to date, so we'll examine it very carefully.

```
address  hex code    mnemonics
3000     3E 20       LD A,&20
3002     CD 5A BB    CALL CharOut
3005     C6 01       ADD A,1
3007     D2 02 30    JP NC,&3002
300A     C9          RET
```

The first thing we do is to load the A register with 32 (&20) – the Ascii code for a space. Then we call the CharOut routine to print the character in the A register on the screen – in this case simply a space.

The CharOut routine doesn't affect the value in the A register so this will still be 32 after the space has been printed. We then add one to the A register.

Now we come to the clever bit. The next instruction is:

```
JP NC,&3002
```

That is, if the carry flag isn't set, jump back to address &3002 and continue the program from there.

The carry flag won't be set, it will be clear, since we haven't gone over 255 in adding one to 32. So the program will jump to &3002 and do the instruction there.

As the instruction at this address calls the CharOut routine, the effect of taking the jump will be to print out the character whose Ascii code is 33 (32 + 1), the contents of the A register.

The number in the A register is then increased to become 34 and, as this still doesn't set the flag, we again take the jump. The character with Ascii code 34 is then printed out, the number in the A register increased to 35, the jump again taken, and so on.

In fact this will continue looping round like this, printing out characters with increasing Ascii values until we have finally reached the character with Ascii value 255. Having printed it out, we add one to it, which should give us 256. As we've seen, the A register can't hold this. In fact it becomes 0. However the Z80 warns us by setting the carry flag.

This time, therefore, when we come to the jump we don't take it, because the carry flag *is* set. We "drop through" to the next instruction which, in this case, is RET.

The effect of this program is to print out all the characters with Ascii codes from 32 to 255. Incidentally, I started at 32 because the numbers below this are control codes, and can cause some rather funny effects.

Now you might like to try tidying the program up by making the first instruction a call to a routine to clear the screen (&BB6C). That's fine, but remember, if you do so, to alter the address you're jumping to – the call to CharOut won't be at &3002 any more.

Just in case you're wondering if you can use INC A instead of ADD A,1 – you can't. Irritatingly INC A doesn't alter the carry flag as does ADD A.

Well that's all for this month. Next month we'll look at the other registers.

Get moving with SPRITES!

By ROLAND WADDILOVE

A SPRITE is a character which can move freely about the screen, independently of any other characters, and no matter what the program happens to be doing at the time.

It can move smoothly on and off the screen and pass in front of or behind other characters without disturbing them. They may be controlled either by hardware, software or a mixture of both.

As you can imagine, a sprite is a handy thing to have around when writing games programs, but unfortunately the Amstrad does not support them. However don't despair, it is possible to simulate many of their properties.

The best way would be to use machine code, which is extremely fast and would allow several sprites to be on the screen at once. Main problem there is that machine code is very difficult and time-consuming to write.

Can it be done in Basic? Well if you only want one or two sprites the answer is yes. We can use Basic interrupts to jump to a subroutine at a specified interval and print the character at the graphics cursor, exclusively ORing it with the background so that anything else on the screen is not altered in any way when the sprite is subsequently removed.

The XOR graphic option is set by printing one of the control characters, (see chapter 9 in the manual). After executing:

PRINT CHR$ (23) ;CHR$ (1)

the actual colour displayed of any point plotted or drawn is calculated by exclusively ORing the PEN number of the point with the PEN number of the background. Then:

PRINT CHR$ (23) ;CHR$ (0)

restores normal operation.

To understand how this works and to be able to predict the resulting colour, we need to know a bit about binary. Our Amstrad can help us here, being an expert on the matter.

The 16 colour mode 0 is really the only suitable one for sprites so we will concentrate on it. The pens are numbered 0 to 15 and in binary we need four digits or bits. To find out what the binary equivalent of a number is simply ask your Amstrad!

PRINT BIN$ (number,4)

where 'number' is between 0 and 15.

XOR is a logical operator treating its arguments as bit patterns. To find out what XOR does ask the expert:

PRINT 0 XOR 0 – gives 0
PRINT 0 XOR 1 – gives 1
PRINT 1 XOR 0 – gives 1
PRINT 1 XOR 1 – gives 0

So to exclusively OR two numbers compare each bit and write down the result as above.

What is 8 XOR 5? Old know-all will tell you it is 13. Is it right?

8 1000
5 0101
8 XOR 5 1101

To find out what 1101 is:

PRINT &X1101 – gives 13

So if you plot a point with graphics PEN 8 on top of a point with PEN 5 using the XOR option the result is PEN 13.

What happens if you plot the point again with PEN 8?

13 1101
8 1000
13 XOR 8 0101

The original colour is restored, 0101 is 5.

We can now predict the colour obtained when using the XOR option. After that struggle it's all downhill now – the rest is fairly straightforward.

TAG will enable characters to be printed at the graphics cursor using graphics commands to position and

```
10 REM Program...1
20 MODE 0:BORDER 0
30 FOR i%=0 TO 15
40 READ j%:INK i%,j%
50 NEXT
60 FOR i%=1 TO 7
70 PAPER i%:LOCATE 2*i%+2,5
80 FOR j%=1 TO 15
90 PRINT " ";CHR$(8);CHR$(10);
100 NEXT
110 NEXT
120 PRINT CHR$(23);CHR$(1)
130 TAG:PLOT -10,-10,8
140 WHILE 1
150 FOR x%=-30 TO 640 STEP 4
160 MOVE x%,200:PRINT CHR$(231);
170 FOR delay=0 TO 50:NEXT
180 MOVE x%,200:PRINT CHR$(231);
190 NEXT
200 WEND
210 DATA 0,6,9,24,1,8,10,26
220 DATA 7,6,7,24,7,8,7,26
```

Program 1

colour them. By selecting our pens and inks carefully a character can appear to pass in front of or behind another character.

If a ball is printed using PEN 8 on top of an object printed using PEN 5 the resulting colour is 13. If PEN 13 is set to the same colour as PEN 5 you cannot see the ball, so it appears to be behind the object. But if PEN 13 is set to the same colour as PEN 8 the ball can be seen, so it appears to be in front of the object.

Program I shows how to move a ball across the screen, passing in front of and behind alternate stripes. First the colours are set, then seven stripes are printed using pens 1 to 7. The XOR option is set, the text and graphics cursors joined and the main loop entered.

Sprites should move independently of the other characters and the program in general once set moving. To do this we need to use interrupts (see chapter 10 in the manual). An interrupt is a signal sent to the processor telling it to drop whatever it is doing and go and do something else, returning when it has finished.

Four timers are available for use by the programmer. The timers are pretty fast, ticking away at 50 times a second. If we write our subroutine to move our sprite at line 1000, then we can say:

EVERY 10 GOSUB 1000

This causes the Amstrad to carry out the subroutine five times a second or every fifth of a second, no matter what it is doing. The delay can be any value allowing the sprite to be moved at any speed.

Watch out if you are using very short delays as so much time may be spent moving the sprite that there isn't enough time to carry on with the rest of the program.

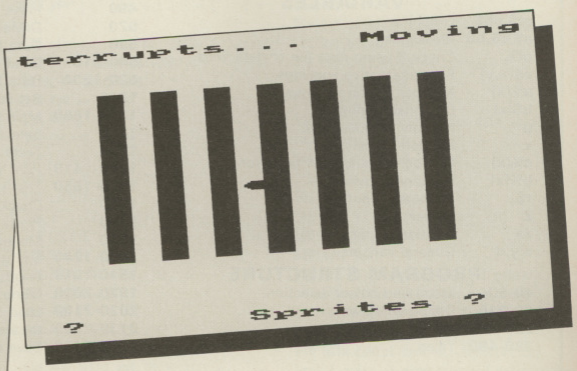
Program II shows how we can combine interrupts with our sprite moving routine.

```
10 REM Program...2
20 MODE 0:BORDER 0
30 FOR i%=0 TO 15
40 READ j%:INK i%,j%
50 NEXT
60 FOR i%=1 TO 7
70 PAPER i%:LOCATE 2*i%+2,5
80 FOR j%=1 TO 15
90 PRINT " ";CHR$(8);CHR$(10);
100 NEXT
110 NEXT
120 x%=-32:xstep%=8:y%=200:ystep%=0
130 PLOT -10,-10,8
140 PRINT CHR$(23);CHR$(1)
150 EVERY 10,3 GOSUB 290
160 a$="Moving a coloured ball using
interrupts...  ":b$=a$
170 c$="   Sprites ?   ":d$=c$
180 WHILE INKEY$<>" "
190 IF LEN(b$)<20 THEN b$=b$+a$
200 LOCATE 1,1:PRINT LEFT$(b$,20)
210 b$=MID$(b$,2)
220 IF LEN(d$)<20 THEN d$=c$+d$
230 LOCATE 2,25:PRINT RIGHT$(d$,18);
240 d$=LEFT$(d$,LEN(d$)-1)
250 WEND
260 END
270 :
280 REM Interrupt subroutine
290 TAG
300 xx%=(x%+xstep%) MOD 640
310 yy%=(y%+ystep%) MOD 400
320 MOVE x%,y%:PRINT CHR$(231);
330 MOVE xx%,yy%:PRINT CHR$(231);
340 TAGOFF:x%=xx%:y%=yy%:RETURN
350 :
360 DATA 0,6,9,24,1,8,10,26
370 DATA 7,6,7,24,7,8,7,26
```

Program II

20	Sets the mode and border.
30-50	Sets the ink colours.
60-110	Prints the coloured stripes.
120	Sets the start position and direction.
130	Sets the colour.
140	Sets the XOR graphic option.
150	Enables the interrupt routine.
160-170	Defines the messages.
180-250	Scrolls the messages in opposite directions.
290	Joins the graphics and text cursor.
300-310	Works out the new coordinates.
320	Erases the old ball.
330	Prints the new ball.
340	Separates the cursors, updates position.

Program II: How it works



SPELING SPELLIN SPILLING SPELLING Quiz

Adapt STEVE LUCAS's entertaining quiz to suit your own requirements

IN this educational spelling game for youngsters four alternative spellings of a word are displayed at the side of the screen and a dictionary definition is displayed at the bottom.

The child has to move a cat until it is next to the correct spelling and then press the space bar. After 20 questions the computer gives your score, together with a list of words which you need to learn for next time!

Three options are offered – using the questions incorporated within the data lines, loading a data file from tape or saving a data file to tape.

The first option uses the game's built-in vocabulary giving 20 questions chosen at random from the data. In order to make the game more useful to teachers and parents the third option allows you to create your own specialised data file on tape.

The computer will ask you to enter

VARIABLES

ct$,cb$	Graphics.
a$,c$,b$	Messages.
pl%	Check if using data file or data lines.
a$(x,y)	Four alternative spellings.
a%(x)	Number of correct answer.
b$(x)	Dictionary definition.
p	Number of questions.
x	Random question.
c%(x)	Number of questions got wrong.
b%(x)	Number of questions.
t$	Keyboard input.
Z	Movement.
za	Position of cat.
s,y,d	General variables.

PROGRAM STRUCTURE

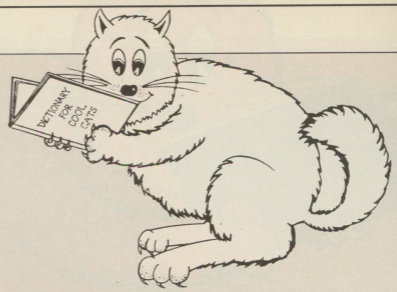
40-50	Mode and colour selection.
60-210	Define windows.
220-270	Define graphics for cat.
280-480	Titles.
490	Plays tune.
520	DIMensions arrays.
530	Data file or data lines.
540-610	Read data if not using a data file.
620-1230	Data for questions.
1240	Set score.
1260-1680	Main loop.
1290	Selects random word.
1300	Tests if already set.
1320-1410	Print words.
1420-1550	Test keys and move cat.
1560	Checks answer.
1570	Prints correct or incorrect.
1660	Moves to end of game.
1690-1740	End game.
1840-1910	Another game?
1920-2010	Menu.
2020-2120	Load data file.
2130-2290	Create data file.
2300-2430	Save data file.

60 "questions". For each question you must enter the four alternative spellings, the dictionary definition and the number of the correct alternative.

The micro will prompt you to place a tape into the recorder and save a copy of your file. It will then go straight on to the game using your words in order that you can check that your data is correct.

To reload your new data file choose option two at the start of the program.

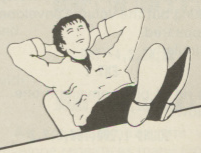
```
10 REM ** Spelling Quiz **
20 REM ** By  Steve W. Lucas **
30 REM (C) Computing with the Amstrad
40 MODE 1:INK 0,6: INK 1,24: INK 2,20
: INK 3,1
50 BORDER 22
60 REM ** define windows **
70 WINDOW #1,1,15,1,5 :PAPER #1,1:PEN
 #1,3
80 WINDOW #2,1,15,6,10 :PAPER #2,2:PE
N #2,3
90 WINDOW #3,1,15,11,15:PAPER #3,1:PE
N #3,3
100 WINDOW #4,1,15,16,20:PAPER #4,2:P
EN #4,3
110 WINDOW #5,16,22,1,21:PAPER #5,0:P
EN #5,3
120 WINDOW #6,1,40,21,25:PAPER #6,3:P
EN #6,1
130 WINDOW #7,23,40,1,20:PAPER #7,1:P
EN #7,0
140 REM ** define characters **
150 SYMBOL AFTER 240
160 SYMBOL 242,48,32,32,32,32,32,31,1
5
170 SYMBOL 243,1,3,7,7,3,1,255,255
180 SYMBOL 244,0,128,192,240,176,240,
248,136
190 SYMBOL 245,7,3,1,1,0,1,1,0
200 SYMBOL 246,255,255,253,129,128,12
9,193,0
210 SYMBOL 247,134,128,128,128,128,12
8,192,0
220 REM ** define graphics for cat **
230 RESTORE 2450
240 FOR x=1 TO 12:READ d
250 ct$=ct$+CHR$(d)
260 IF d<>32 THEN cb$=cb$+CHR$(d)
270 NEXT x
280 REM ** titles **
290 a$="      S p e l l i n g    Q u
i z"
300 c$="           by  Steve W. Lucas"
310 B$=" "+CHR$(10)+CHR$(8)+" "+CHR$(
10)+CHR$(8)+CHR$(8)+" "
320 FOR x=1 TO 37
330 PEN 3
340 LOCATE X,5:PRINT b$
350 LOCATE X+1,5:PRINT b$
360 LOCATE X+2,5:PRINT b$
370 LOCATE X+3,5
380 IF x<28 THEN PRINT cb$
390 LOCATE x,10
400 PEN 2
410 PRINT MID$(a$,x,1)
420 LOCATE x,17
430 PEN 1
440 PRINT MID$(c$,x,1)
450 NEXT x
460 PEN 3:FOR x=9 TO 29 STEP 4
470 LOCATE x,2 :PRINT cb$
480 NEXT
490 PEN 1:GOSUB 1690:REM ** play tune
 **
500 DATA 239,239,213,239,190,239,239,
213,190,179,159,239,239,213,190,179,1
59,239,239,213,239,159,239,142,239,12
7,239,127,142,159,179,190,213,239
510 REM ** to add extra questions , i
ncrease the size of the array in the
next line **
520 DIM a$(61,4),a%(61),b$(61),n$(50)
,c%(20),b%(60)
530 GOSUB 1930:REM ** use data file o
r data in program **
540 IF pl%)0 THEN 1240
550 RESTORE 630:REM ** use data suppl
ied in program **
560 FOR x=1 TO 60
570 FOR y=1 TO 4
580 READ a$(x,y)
590 NEXT y
600 READ a%(x),b$(x)
610 NEXT x
620 REM ** The data lines all contain
 4 alternative spellings, the number
of the correct one and a dictionary d
efinition **
630 DATA contol,kontroll,controll,con
trol,4,power of directing and restrai
```




```
ning
640 DATA casette,cassette,casete,cass
ete,2,type of tape recorder
650 DATA complicated,complhicated,com
plecated,complacated,1,involved
660 DATA probablie,probabley,probably
,probebly,3,most likely
670 DATA altaring,altering,alturing,h
altering,2,changing
680 DATA vibrasion,vibration,vibrasia
n,vitratian,2,continuous movement
690 DATA concider,concidur,considure,
consider,4,contemplate
700 DATA figure,fighure,phigure,phigr
e,1,shape
710 DATA whasteful,wastfull,wastefuli
,wasteful,4,not economical
720 DATA amplifies,amplifies,amplyf
ies,amplyfies,2,makes louder
730 DATA flasching,flashzing,flachsin
g,flashing,4,a lamp turning on and of
f
740 DATA manual,manuel,mannual,mannue
l,1,done by hand
750 DATA encountar,encountur,encountt
er,encounter,4,close contact
760 DATA expreshun,expresion,expressi
on,hexpression,3,wording or phrase
770 DATA content,contant,contente,kon
tent,1,satisfied
780 DATA acumalate,accumulate,acumala
te,accumerlate,2,heap up
790 DATA assembel,assemble,asemble,as
embel,2,bring together
800 DATA situation,sithuation,situasi
on,situachion,1,position you are in
810 DATA computer,computar,commputer,
computur,1,an electronic machine
820 DATA endles,hendless,endless,endl
less,3,without end
830 DATA envelope,henvelope,envalope,
enverlope,1,used for sending letters
840 DATA extraction,extracshion,extra
ktion,extracsion,1,a removal
850 DATA forcabley,forcibley,forsable
y,forsibley,2,done using force
860 DATA halucination,hallucination,h
allucinatian,halucinatian,2,illusion
870 DATA idolise,idolize,idalise,idal
ize,2,to love
880 DATA imposibal,imposible,impossib
le,impossable,3,cannot be done
890 DATA mechanical,mecanicol,mecanic
al,mechanicle,1,works by machinery
900 DATA necesary,neccessary,necessar
y,neccesary,3,needed
910 DATA navigation,navigashion,navig
atian,navigaton,1,mapping of a course
920 DATA orchid,orkid,orchyd,orcid,1,
flowering plant
930 DATA oxigen,oxegen,oxygen,oxejun,
3,a gas
940 DATA spatious,spachious,spasious,
spacious,4,with plenty of room
950 DATA alocation,aloccation,allocat
ion,alloccation,3,designation
960 DATA compulsory,compulsary,compul
sery,compullsory,1,must be done
970 DATA controler,controllor,control
ler,controlor,3,person in charge
980 DATA calendar,calender,callendar,
callender,1,table of the year's dates
990 DATA appreciate,apreciate,aprecie
te,appresiate,1,set a high value on
1000 DATA style,styal,stial,stile,1,d
esign
1010 DATA sterilise,steralise,sterali
ze,sterilize,4,to get rid of microbes
1020 DATA sindicate,syndicate,syndeca
te,sindecate,2,group of people
1030 DATA tempreture,teaperature,temp
eretune,temparature,2,degree of heat
1040 DATA hidrogen,hydrgen,hydrogen,h
iderogen,3,a chemical
1050 DATA bugerigar,bugarigar,budgeri
gar,budgeregar,3,a type of bird
1060 DATA administer,adminster,admine
ster,adminestar,1,look after affairs
1070 DATA pharmecy,pharmarcy,pharmerc
y,pharmacy,4,chemist's shop
1080 DATA possession,posesion,possesi
on,posession,1,something owned
1090 DATA retaleate,retaliate,retalya
te,retalate,2,get your own back
1100 DATA stomach,stomache,stomake,st
omuch,1,part of the body
1110 DATA rhithea,rithem,rhythem,rhyt
hm,4,part of music
1120 DATA sucessful,succesful,success
ful,sucesful,3,accomplished
1130 DATA substansial,substanshul,sub
stantiel,substantial,4,considerable
1140 DATA disappointing,dissappointin
g,disapointing,dissapointing,1,not up
 to expectations
1150 DATA capital,capitol,kapertal,ka
pitol,1,of chief importance
1160 DATA whipet,whippet,wippet,whipe
t,2,type of dog
1170 DATA transperent,transparent,tra
nsparant,transperant,2,can be seen th
rough
1180 DATA nececity,necesity,necessety
,necessity,4,something which is neede
d
1190 DATA asasination,assassination,a
ssasination,asassination,2,murder
1200 DATA bronkitis,broncitis,bronchi
tis,bronchitus,3,an illness
1210 DATA centenary,centenery,sentena
ry,sentenery,1,hundredth anniversary
1220 DATA sentrafugal,centrafugal,cen
trifugal,centrifugul,3,force of a spi
nning object
1230 DATA conventional,convenshional,
conventionul,convensional,1,normal
1240 s%=0: REM ** set score to zero *
*
1250 CLS:RANDOMIZE TIME
1260 REM ** main control loop **
1270 WHILE cc%<99
1280 FOR p=1 TO 20: REM ** set 20 que
stions **
1290 x=INT(RND(1)*60+1)
1300 IF b%(x)>0 THEN 1290
1310 b%(x)=1
1320 IF t%=x THEN 1290
1330 CLS @1:CLS @2: CLS @3:CLS @4:CLS
 @5: CLS @6:CLS @7
1340 PRINT @7:PRINT @7:PRINT @7:PRINT
 @7,"       UP = A"
1350 PRINT @7:PRINT @7:PRINT @7,"
 DOWN = Z"
1360 PRINT@7:PRINT@7:PRINT@7,"  (Spa
ce   Bar)"
1370 PRINT@7:PRINT@7," to select wor
d."
1380 LOCATE (40-LEN(b$(x)))/2,24:PEN
1:PRINT b$(x);
1390 FOR y=1 TO 4
1400 PRINT @y:PRINT@y," ";a$
(x,y)
1410 NEXT y
```

```
1420 Z=2:za=2
1430 T$=INKEY$
1440 PRINT CHR$(30)
1450 t$=UPPER$(t$)
1460 IF T$="A" THEN za=z:Z=Z-5:PRINT CHR$(7)
1470 IF T$="Z" THEN za=z:Z=Z+5:PRINT CHR$(7)
1480 IF Z<2 THEN Z=2
1490 IF Z>17 THEN Z=17
1500 IF za>0 THEN LOCATE #5,1,za:PRINT#5,"     "
1510 IF za>0 THEN LOCATE #5,1,za+1:PRINT#5,"     "
1520 IF za>0 THEN LOCATE #5,1,za+2:PRINT#5,"     ":za=0
1530 LOCATE #5,2,z:PRINT#5," "+t$
1540 LOCATE #5,1,1
1550 IF t$<>" " THEN 1430
1560 IF z=2 THEN c%=1 ELSE IF z=7 THEN c%=2 ELSE IF z=12 THEN c%=3 ELSE c%=4
1570 CLS #7:PRINT CHR$(7):LOCATE #7,4,3
1580 IF c%=a%(x) THEN PRINT #7,SPC(3)"Correct":s%=s%+1 ELSE PRINT #7,SPC(2)"Incorrect":LOCATE #7,6,6:PRINT #7, "It was :-":LOCATE #7,5,8:PRINT #7, a$(x,a%(x)):c%(p)=x
1590 LOCATE #7,5,12:PRINT #7,"Score :-"s%
1600 LOCATE #7,5,15:PRINT #7,"Press the "
1610 LOCATE #7,4,17:PRINT #7,"<Space Bar>"
1620 LOCATE #7,1,19:PRINT #7,"for next question."
1630 gg$=INKEY$:IF gg$<>" " THEN 1630
1640 t%=x
1650 NEXT p
1660 GOSUB 1770:REM ** end of game **
1670 WEND
1680 END
1690 RESTORE 500
1700 FOR x=1 TO 34: READ d
1710 SOUND 5,d,20,15
1720 NEXT x:RETURN
1730 CLS: LOCATE 4,2:PRINT" Goodbye. Thank you for playing."
1740 PEN 2: LOCATE 10,10:PRINT"You scored :-"s%
1750 GOSUB 1690
1760 END
1770 CLS: LOCATE 2,2:PRINT"At the end of the game you scored "s%
1780 GOSUB 1690:REM ** play tune **
1790 PEN 2:LOCATE 3,5:PRINT"Press the <Space Bar> to view the words I think you need to practice."
1800 sp$=INKEY$:IF sp$<>" " THEN 1800
1810 CLS
1820 FOR p=1 TO 20:IF c%(p)>0 THEN PRINT a$(c%(p),a%(c%(p)))
1830 NEXT p
1840 FOR x=1 TO 20: c%(x)=0: NEXT x:REM ** reset array elements **
1850 LOCATE 5,24:PRINT"Do you want to play again <Y/N> ?"
1860 s%=0
1870 gg$=INKEY$:gg$=LOWER$(gg$):IF gg$="y" THEN CLS:counter=counter+1
1880 IF counter=3 THEN RESTORE 630:counter=0:FOR x=1 TO 60:b%(x)=0:NEXT x
1890 IF gg$="y" THEN RETURN
1900 IF gg$="n" THEN CLS:PRINT"Goodbye. Thank you for playing":END
1910 GOTO 1870
1920 REM ** make choices **
1930 LOCATE 1,20:PRINT"Do you want :-"
1940 LOCATE 1,21:PEN 3:PRINT"1.  To use the data in the program."
1950 PRINT"2.  To load a data file from tape."
1960 PRINT"3.  To create a data file."
1970 gg$=INKEY$:gg$=LOWER$(gg$):IF gg$("1" OR gg$)"3" THEN 1970
1980 IF gg$="1" THEN CLS:PRINT"Please wait a second!":RETURN
1990 IF gg$="2" THEN pl%=1: GOSUB 2030:REM ** load data from tape **
2000 IF gg$="3" THEN pl%=1:GOSUB 2140:REM ** write your own data tape **
2010 RETURN
2020 REM ** load data from tape **
2030 CLS:PEN 1:LOCATE 4,2:PRINT"Please insert the data tape now"
2040 OPENIN"data"
2050 FOR x=1 TO 60
2060 FOR y=1 TO 4
2070 INPUT #9,a$(x,y)
2080 NEXT y
2090 INPUT #9,a%(x),b$(x)
2100 NEXT x
2110 CLOSEIN
2120 RETURN
2130 REM ** write a data file to tape **
2140 FOR x=1 TO 60
2150 CLS:LOCATE 4,2:PRINT"Question number "jx
2160 d%=0
2170 WHILE d%=0
2180 FOR y=1 TO 4
2190 LOCATE 2,y*2+4:PEN 2:PRINT"Spelling number "jyj" .."j:INPUT a$(x,y)

2200 NEXT y
2210 LOCATE 1,18:PEN 1:PRINT"Which of these is the correct spelling  (1-4)"j:INPUT a%(x)
2220 IF a%(x)>4 OR a%(x)<1 THEN PRINT CHR$(7):LOCATE 1,18:PRINT STRING$(70,CHR$(32)):GOTO 2210
2230 PEN 3:PRINT"What is the dictionary definition of theword "j:INPUT b$(x)
2240 LOCATE 1,25:PRINT"Is this correct <Y/N) ?"
2250 gg$=INKEY$:gg$=LOWER$(gg$):IF gg$="y" OR gg$="n" THEN 2260 ELSE 2250
2260 IF gg$="y" THEN d%=1
2270 CLS
2280 WEND
2290 NEXT x
2300 REM ** NOW SAVE THE DATA TO TAPE **
2310 CLS:PEN 1:LOCATE 4,2:PRINT"Please insert the tape ready to record"
2320 OPENOUT"data"
2330 FOR x=1 TO 60
2340 FOR y=1 TO 4
2350 PRINT #9,a$(x,y)
2360 NEXT y
2370 PRINT #9,a%(x),b$(x)
2380 NEXT x
2390 CLOSEOUT
2400 CLS:PRINT"Press the <Space Bar> to start your game"
2410 gg$=INKEY$:IF gg$<>" " THEN 2410
2420 CLS:PEN 1: PRINT"Please wait a second!"
2430 RETURN
2440 REM ** data for cat graphics **
2450 DATA 32,32,242,243,244,10,8,8,8,245,246,247
```



Give your fingers a rest . . .

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

See our special offer on Page 77.

Take a deeper look at those volume and pitch envelopes

Part IV of NIGEL PETERS' series on coaxing sounds from the CPC464

THIS month we'll be taking another, deeper look at the volume and pitch envelopes we've explored in the previous two articles.

First of all, however, let's meet some old friends and see how they work. Enter:

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

to define volume envelope one. This will consist of five steps, each step lasting a fifth of a second (20/100). At the start of each step the volume will increase by two.

Next create pitch envelope one with:

```
ENT 1,5,10,20
```

This also consists of five steps, each lasting a fifth of a second. As each step starts the pitch parameter is increased by ten.

Now let's hear them both in action with:

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

This plays the sound on channel A for one second. The SOUND commands an initial pitch is 200 and the initial volume is 5.

Both these are effected by the final two parameters of the SOUND command which allows volume envelope one and pitch envelope one to exert their influence.

The result is a sound that lasts for one second, its pitch decreasing while its volume increases.

Now let's get rid of volume envelope one by typing in:

```
ENV 1
```

and pressing Enter. Defining a volume envelope without giving it parameters effectively deletes that envelope.

Now if you try:

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

you'll still hear the note going down in pitch in five steps, but the volume will stay constant. The pitch envelope is working but the volume envelope has disappeared.

Similarly:

```
ENT 1
```

gets rid of pitch envelope one.

Now:

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

just gives you a one second note of pitch 200 at volume level 5. The envelope parameters have no effect as we've destroyed our envelopes.

Let's take a closer look at the volume envelope. Re-define envelope one with:

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

As this has five steps, each step lasting a tenth of a second, the whole envelope lasts for half a second.

Listen to its effect on:

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

Remember what happens if the duration parameter of the SOUND command exceeds the total duration of the volume envelope? That is, if the SOUND lasts longer than the envelope?

Find out with:

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

which lasts a full second.

As you can hear, the volume envelope works as before for the first half second. Then the envelope stops having any effect and the sound carries on for the remaining half second at the final volume level.

In other words the envelope only works once.

We can make the volume envelope repeat if we want to. It's easy and can produce some interesting effects.

But first I have to admit to another time when I've been less than honest with you.

If you can think back to the January issue, I told you then that the duration parameter of the SOUND command could range from 0 to 32767.

In fact the duration parameter can take values between −32768 and 32767. It can have negative values.

This looks fairly silly at first. Durations of 0 to 32767 fit in with common sense, but what about the negative values? Does the sound go backwards in time?

In reality what happens is that the negative number isn't treated as the length of time the note plays but as the number of times the volume envelope is to be repeated.

The negative sign tells the Amstrad that the number after it isn't a duration but a counter indicating the number of times the volume envelope is to be obeyed.

If the duration parameter is −3 then the volume envelope is repeated three times. If it's −100 then the volume envelope works on the sound 100 times.

Don't just take my word for it, try it out.

Enter:

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

and

```
SOUND 1,200,-2,5,1
```

and hear for yourself. The −2 in the duration parameter causes volume envelope one (called by the last parameter in the SOUND command) to repeat twice.

Since the envelope lasts for half a second the whole sound lasts a full second.

Similarly:

```
SOUND 1,200,-1,5,1
```

lasts for half a second (one repetition of a volume envelope lasting 0.5

seconds) and:

```
SOUND 1,200,-20,5,1
```

lasts for 10 seconds.

As you can see and hear, when the duration parameter is negative the duration of the sound is determined by the duration of the volume envelope and the number of times it is repeated. Hence the volume envelope can control the length of the sound.

Notice, however, that when the envelope repeats, it doesn't go back to the original volume but starts again at the volume level where it left off. When it gets out of range it wraps around.

Take the example of a volume envelope:

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

and

```
SOUND 1,200,-2,5,1
```

Here the envelope will work on the sound twice. Now you might expect the volumes to go 7, 9, 11, 13, 15 for the first repetition and then 7, 9, 11, 13, 15 for the second. This isn't what happens, however.

The first application of the envelope produces the volumes 7, 9, 11, 13, 15 but the second goes on to produce 1, 3, 5, 7, 9. Notice that the volume parameter wraps around into range.

If you find that hard to follow, try:

```
ENV 1,5,2,100
```

and

```
SOUND 1,200,-2,5,1
```

and you should hear what's happening.

The volume envelope affects the SOUND command in exactly the same way the second time around, but the resulting volumes are different as it starts work at a different volume level. This makes the effect of a repeated volume envelope difficult to predict.

However by defining a volume envelope and repeating it just once you don't have to bother calculating the duration parameter of a note. The note will last as long as the volume envelope and no longer.

This can make life easier at times, especially if you alter the duration of any of the steps in the volume envelope. The single repetition automatically adjusts the length of the sound in line with such changes.

Before we leave volume envelopes, try:

```
SOUND 1,200,-3,5
```

and

```
SOUND 1,200,-10,5
```

We seem to be telling the Amstrad to repeat a volume envelope without specifying which one to repeat. Can you understand what's happening?

The answer is that there is a volume envelope in use. It's the default envelope 0, which plays for two seconds at the volume specified in the SOUND command.

Hence the first sound played for six seconds, the next for twenty.

Now that we've dealt with repetition of volume envelopes you'll probably suspect that we'll go on to dealing with repeating pitch envelopes and you're right.

Define a tone envelope with:

```
ENT 1,5,10,20
```

and hear its effect on:

```
SOUND 1,200,100,5,0,1
```

The sound goes down in pitch in five steps. Each step lasts a fifth of a second, so the pitch envelope lasts one second, as does the whole note. What happens if the pitch envelope doesn't last as long as the duration in the SOUND command?

As ever, try it and see. Use:

```
ENT 2,5,10,10
```

and

```
SOUND 1,200,100,5,0,2
```

Here the pitch envelope lasts for half a second while the sound lasts for a whole second.

As you heard, the envelope works its way for the first half second, then the sound continues at the final pitch until its duration is used up.

In other words the envelope is used once, then the sound carries on without the benefit of its assistance.

We can, however, make the pitch envelope repeat. Try defining one with:

```
ENT -2,5,10,10
```

This looks unusual, but the Amstrad will accept it, minus sign and all. The minus sign tells the micro that when that pitch envelope is called by a SOUND it is to repeat itself, if at all possible.

Try it out with:

```
SOUND 1,200,100,5,0,2
```

Now the pitch envelope is repeated during the whole of the time the sound plays. Since the envelope lasts for half a second and the duration of the sound is one second, the envelope is repeated twice.

Notice that we call the envelope in the SOUND command with just the figure 2, we don't use the minus sign. Try:

```
ENT -3,10,5,100
```

with

```
SOUND 1,100,1000,5,0,3
```

and

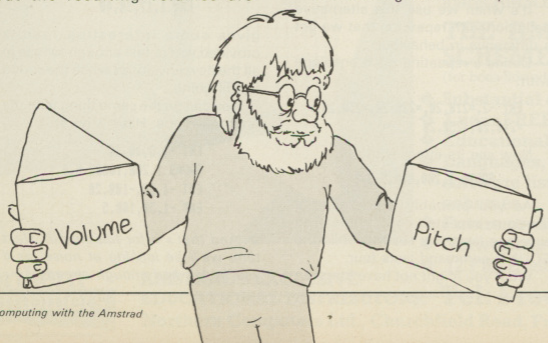
```
ENT -4,5,20,20
```

with

```
SOUND 1,200,150,5,0,4
```

As you can hear, the volume envelope repeats as many times as it can within the duration of the SOUND command.

Notice that repetition with pitch envelopes is very different from



repetition with volume envelopes.

With volume envelopes it's the length of the envelope and the number of times that it's repeated which determines the duration of the sound produced.

In the case of pitch envelopes the duration parameter of the SOUND command holds sway. The pitch envelope just tries to repeat itself as many times as possible within the length of time. When it runs out of time, it stops.

One thing they do have in common is that when the envelope repeats it doesn't go back to the beginning but carries on from where it left off.

Take the case of:

```
ENT -1,5,10,20
```

and

```
SOUND 1,200,200,5,0,1
```

Here the pitch envelope repeats twice.

Again you might think that the pitch of the sound would take the values 210, 220, 230, 240, 250 then 210, 200, 230, 240, 250 for the second repetition. And, again, you'd be wrong!

When the pitch envelope starts over, it doesn't go back to the base pitch in the SOUND command, but carries on from where it left off.

Hence the pitch values produced were 210, 220, 230, 240, 250 the first time through the envelope, then 260, 270, 280, 290 and 300 for the second pass.

You get a series of 10 notes instead of the two identical series of five that you might logically expect. This, especially when combined with wraparound effects at the extremes of the pitch parameter's range, can make predicting what happens when pitch envelopes are repeated very difficult indeed.

So far we've defined pitch envelopes in the form:

```
ENT S,T,V,W
```

There is another form where, instead of using steps to get the notes you want, you just tell the Amstrad which note to play.

It takes the form:

```
ENT =pitch,duration
```

where the *=pitch,duration* can be repeated up to five times.

Here *pitch* is the actual pitch that you want the note to be. It takes the same values as the pitch parameter in the SOUND command and works in exactly the same way.

The *duration* lies between 0 and 255, telling the micro how long that note is to last.

As you know by now, defining a pitch envelope the old way with:

```
ENT 1,5,10,20
```

and then calling it with

```
SOUND 1,200,100,5,0,1
```

gives a sound made up of five notes of pitches 210, 220, 230, 240 and 250.

You can get the same effect using these values in the alternative form of pitch envelope definition:

```
ENT 2,=210,20,=220,20,=230,20,
=240,20,=250,20
```

and

```
SOUND 1,200,100,5,0,2
```

produce exactly the same effect. In fact when you define the tone envelope in this way the pitch parameter in the SOUND command is ignored. Try:

```
SOUND 1,0,100,5,0,2
```

which produces exactly the same notes. Also, despite the alternative form of definition, envelope 2 behaves just like an ordinary pitch envelope.

```
SOUND 1,200,50,5,0,2
```

has it cut off in its prime while:

```
SOUND 1,200,400,5,0,2
```

has it working for one second, then the sound staying at the final pitch for three seconds.

It's when we use this alternative definition with repetition that we get a difference in behaviour.

Define a repeating pitch envelope with:

```
ENT -2,=210,20,=220,20,=230,20,
=240,20,=250,20
```

Now see its effect with:

```
SOUND 1,200,400,5,0,2
```

As you probably expected, the envelope repeated four times. The envelope lasts one second while the SOUND command lasts four.

What you might not have foreseen is that each time the pitch of the notes is exactly the same. They go 210, 220, 230, 240, 250 then 210, 220, 230, 240, 250 and so on.

This new way of defining the pitch envelope doesn't use steps, but refers to the absolute pitch of the notes.

Each repetition goes back to the beginning again and goes through the same sequence made up of notes of the same pitch. This makes it much easier to predict what the envelope will do.

But try to predict what happens when you use repeating pitch and volume envelopes at the same time. Can you foretell what:

```
ENV 1,5,2,20
ENT -1,5,10,20
SOUND 1,200,-4,5,1,1
```

will produce? I doubt it. Using the alternative form of pitch envelope makes things a little easier:

```
ENV 1,5,2,10
ENT -2,=210,20,=220,20,=230,20,
=240,20,=250,20
SOUND 1,200,-4,5,1,2
```

However it's still not all that predictable.

And that's where we leave volume and pitch envelopes except for a little bit of fun.

So far we've defined our envelopes and left them untouched while the sound was playing. What would happen if you re-defined them while it was playing ? Why not try it? Using:

```
ENV 1,5,10,20
SOUND 1,200,-100,5,1
ENV 1,5,10,30
ENV 1,7,3,2
ENV 1,17,30,40
```

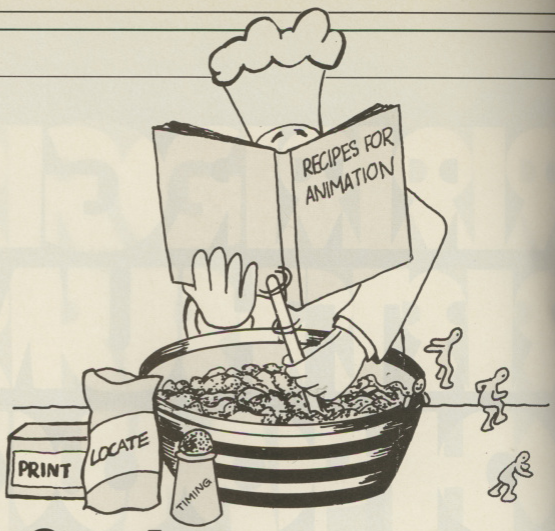
gives some interesting results provided you're fast enough to type in all the new envelopes while the sound is still going.

You can do the same thing with the pitch envelope. Have a go at:

```
ENT -1,5,10,20
SOUND 1,200,10000,5,0,1
ENT -1,20,-100,20
ENT -1,30,100,5
```

● *And that's it for this month. Next time we'll be looking at noise (as if you hadn't had enough already).*


GRRGRRRR
GRRRRRRR
THUMPHO
RRRRGRR
GRRTHUM
HOOHOOG
RGRRREA
Killer Gorilla and Gauntlet. Together on one ca



Cook up some simple animation

First in a short series by KEVIN EDWARDS

HAVE you ever played a game on your Amstrad and wondered how the programmer gets the objects to move around the screen? It certainly looks impressive and would seem to be beyond the scope of tyro programmers.

However if you think animation can only be achieved by complex machine code routines understood only by computer whiz-kids, you'd be wrong.

This, the first of a short series, will show you how to produce simple animation effects using just two Basic commands – LOCATE and PRINT. You are probably familiar with them already as they are among the first things you learn in Basic.

LOCATE X,Y is used to position the text cursor at position X,Y. X is the number of character spaces in from the left of the screen and Y the number of rows down from the top. Once the cursor has been positioned using LOCATE, the PRINT command is used to display text at the cursor's position.

If you don't understand PRINT and LOCATE you'd better learn about them before you tackle the rest of this article. You'll find Chapter 3 Page 4 and Chapter 4 Page 11 of the User Instructions very helpful. These are two of the three ingredients of simple animation.

Timing is the third. You've probably already used LOCATE and PRINT many times without animating anything. It's when you combine the two in the right order that you can make objects move across the screen. Timing is all in animation.

So to sum up the above, positioning, printing and timing must be precise in order to produce true animation. Now let's see them in action.

```
10 REM PROGRAM I
20 MODE 1
30 LOCATE 10,14
40 PRINT"HELLO, I'M MOVING"
50 LOCATE 1,25
60 FOR loop=1 TO 20
70 PRINT CHR$(10);
80 FOR delay=1 TO 50:NEXT delay
90 NEXT loop
```

Program I

Program I prints a message in the middle of the screen and scrolls it upwards.

Lines 30 and 40 position and display the message at position 10,14.

Line 50 moves the text cursor to the bottom left corner of the screen. The cursor is moved to this position so we can PRINT CHR$(10) to move the screen display up one line. This CHR$(10) creates a blank line at the bottom of the screen. In order to make room for it everything else on the screen is moved up one row, the top line disappearing.

Note that screen will only scroll in this way if the cursor is on the bottom row. To be safe you should always position it in the bottom left corner of the screen.

The FOR . . . NEXT loop puts 20 of these blank lines on the screen one after the other. Each new line forces the message upwards with the result it rises smoothly up the screen.

Line 80 is a short delay to slow down the scrolling. See what happens when line 80 is deleted. It really is moving!

Program II also scrolls a message but this time it moves it down the screen. Text can be scrolled down by positioning the cursor at the top left corner of the screen and PRINTing CHR$(11). This is a control code which attempts to move the oursor up.

However as you've already put the cursor at the top of the screen it can't go any higher. Something has to give. The result is that the screen will now

```
10 REM PROGRAM II
20 MODE 1
30 LOCATE 10,14
40 PRINT"HELLO, I'M MOVING"
50 LOCATE 1,1
60 FOR loop=1 TO 20
70 PRINT CHR$(11);
80 FOR delay=1 TO 50:NEXT delay
90 NEXT loop
```

Program II

scroll down one character row and leave a blank line at the top of the screen. Every other row is moved down one.

The FOR . . . NEXT loop formed by lines 60 to 90 performs this operation 20 times, with the result that the message scrolls down.

This method of animation is quite simple but surprisingly effective. To

```
10 REM PROGRAM III
20 MODE 1
30 FOR loop=1 TO 40
40 LOCATE loop,12
50 PRINT CHR$(250);
60 FOR delay=1 TO 40:NEXT delay
70 NEXT loop
```

Program III

test your understanding, why not write a program that puts a message in the middle of the screen, moves it up to the top line, then down to the bottom and back to the middle again?

Unfortunately we can only scroll the screen up or down and not left and right. This sideways scrolling, as it is known, is very useful but we do not have its power within our grasp using PRINT.

Also while scrolling animation is fine as far as it goes, there are many cases where moving the whole screen can create problems. For example, if you need to move a character over a stationary background scrolling will be of no use at all.

When you put a blank line at the bottom of the screen everything moves up to make room for it. Your background, or part of it, disappears.

Happily there are ways of doing this. Most animation involves moving

```
10 REM PROGRAM IV
20 MODE 1
30 FOR loop=40 TO 1 STEP-1
40 LOCATE loop,12
50 PRINT CHR$(251);
60 FOR delay=1 TO 40:NEXT delay
70 NEXT loop
```

Program IV

one or more characters around the screen, so to start with we'll move a single character in different directions and in different ways.

Program III moves a stick man character from left to right and program IV does the same thing, except this time he travels from right to left. In both cases a trail of the character is left behind the moving figure.

Let's have a closer look at Programs III and IV. In both cases line 30 controls the direction of the character's movement in the X axis – across the screen. If variable *loop* increases, the direction of movement will be from left to right. For a decreasing loop variable, as in Program IV, the opposite is the case. The little man enters stage right.

Lines 40 and 50 position and display the character on the screen. The FOR . . . NEXT control variable *loop* determines the X coordinate of the position the character is printed at. The Y coordinate is constant for the two programs, the man being always on the same row.

PRINT CHR$(n) outputs character *n* at the text cursor – line 50. You may have noticed that the number *n* is different in the two programs. This is because two different characters are used.

One character is a stick man facing left, used when the movement is to be from right to left – CHR$(251). The other is a stick man facing right, used when the movement is to be from left to right – CHR$(250).

See Appendix III pages 2-13 of the User Instructions for diagrams of the characters you can use.

Of course it doesn't have to be a little man you print. Try changing the character number to different values but make sure that it's between 32 and 255.

At the moment Programs III and IV move a character across the screen but they leave a trail of little men. We've got something moving, but it's not really true animation. To remove that ugly trail add the following two lines to either Program III or IV:

```
64 LOCATE loop,12
66 PRINT CHR$(32);
```

As you can see this is more like the animation we've come to expect. What happens is that after line 50 puts a man on the screen line 66 PRINTs CHR$(32), a space, at the same position. This effectively erases the man from the screen.

Then the man is printed again one position further on, erased, then displayed in the next position, and so on.

This "now you see him, now you don't" routine gives the impression that it's only one character moving across the screen. Provided the difference in print position isn't too great and it is done quickly, the eye is fooled. It levels out the jerky succession of PRINTs into one smooth piece of movement. And that's true animation.

Notice that the space isn't printed until after the short delay of line 60. If there was no delay between printing the character and erasing it with the space, the character would only appear on the screen for a split second before being erased. Instead of "now you see it, now you don't" it would be a case of "you might catch a glimpse if you're lucky"!

Try deleting line 60 to see what happens – it's not easy to see is it? As was said before, timing is vital in animation.

```
10 REM PROGRAM V
20 MODE 1
30 FOR loop=1 TO 39
40 LOCATE loop,12
50 PRINT CHR$(32);CHR$(250)
60 FOR delay=1 TO 40:NEXT delay
70 NEXT loop
```

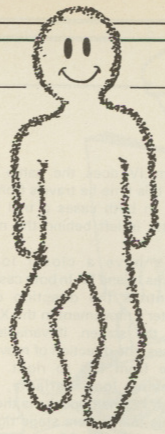
Program V

Programs V and VI use a slightly different method to achieve the same effect. This time two characters are PRINTed in line 50. One is a space, the other is our little man. Have a look at Programs V and VI and see if you can work out why printing a space and the new character at the same time deletes the old character.

What's happening is that each time round the loop line 50 not only prints the little man but also a space next to it. And this space erases the

```
10 REM PROGRAM VI
20 MODE 1
30 FOR loop=39 TO 1 STEP-1
40 LOCATE loop,12
50 PRINT CHR$(251);CHR$(32)
60 FOR delay=1 TO 40:NEXT delay
70 NEXT loop
```

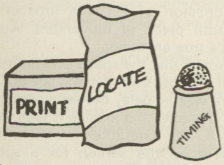
Program VI



little man who was there before.

By doing two things with the PRINT you are saving yourself having to use lines 64 and 66 that you added to the previous two programs. Notice that in Program V the space comes before the little man while in Program VI it's the other way around. Can you see why?

Once you've understood the



above, movement up and down the screen is simple. All you do is alter the Y coordinate of the character's position instead of the X coordinate. Increasing Y moves the character down one line and decreasing moves it up one line.

Try Programs VII and VIII which put this into practice. Notice that when we're going up or down we have to use a separate line to PRINT

```
10 REM PROGRAM VII
20 MODE 1
30 FOR vertical=1 TO 24
40 LOCATE 19,vertical
50 PRINT CHR$(249)
60 FOR delay=1 TO 60:NEXT delay
65 LOCATE 19,vertical
66 PRINT CHR$(32)
70 NEXT vertical
```

Program VII

```
10 REM PROGRAM VIII
20 MODE 1
30 FOR vertical=24 TO 1 STEP-1
40 LOCATE 19,vertical
50 PRINT CHR$(249)
60 FOR delay=1 TO 60:NEXT delay
65 LOCATE 19,vertical
66 PRINT CHR$(32)
70 NEXT vertical
```

Program VIII

the space that erases the little man. The technique we used in Programs V and VI won't work.

Program IX uses vertical and horizontal movement – not at the same time – to move a character round in a square. The subroutine at line 210 displays and deletes the character. The wise among you can have a go at changing the size of the square.

So far our little man has always moved in a straight line. Either the X or the Y coordinate has been constant while the other varied.

Diagonal movement can be achieved by combining vertical and horizontal movement at the same time. That is, both the X and Y

```
10 REM PROGRAM IX
20 MODE 1
30 x=1:y=1
40 FOR walkright=1 TO 39
50 x=x+1
60 GOSUB 210
70 NEXT walkright
80 FOR walkdown=1 TO 23
90 y=y+1
100 GOSUB 210
110 NEXT walkdown
120 FOR walkleft=1 TO 39
130 x=x-1
140 GOSUB 210
150 NEXT walkleft
160 FOR walkup=1 TO 23
170 y=y-1
180 GOSUB 210
190 NEXT walkup
200 END
210 REM display man, wait,
    then remove
220 LOCATE x,y
230 PRINT CHR$(248)
240 FOR delay=1 TO 40:NEXT delay
250 LOCATE x,y
260 PRINT CHR$(32)
270 RETURN
```

Program IX

```
10 REM PROGRAM X
20 MODE 1
30 x=10:y=10
40 FOR upright= 1 TO 5
50 x=x+1:y=y-1
60 GOSUB 210
70 NEXT upright
80 FOR downright= 1 TO 4
90 x=x+1:y=y+1
100 GOSUB 210
110 NEXT downright
120 FOR leftdown=1 TO 4
130 x=x-1:y=y+1
140 GOSUB 210
150 NEXT leftdown
160 FOR leftup=1 TO 4
170 x=x-1:y=y-1
180 GOSUB 210
190 NEXT leftup
200 END
210 REM display man, wait,
    then remove
220 LOCATE x,y
230 PRINT CHR$(248)
240 FOR delay=1 TO 40:NEXT delay
250 LOCATE x,y
260 PRINT CHR$(32)
270 RETURN
```

Program X

coordinates are altered to give the next position of the little man. Program X shows how it's done.

And that's about it for this month. As you've seen, simple animation is just a well timed combination of LOCATE and PRINT.

By printing a character, then erasing it and printing it again in a new position the character can appear to move.

The final program, XI, is just a bit of fun. It's called the The Dancing Man. You'll see why.

● ***Next month:*** *Using colour changes to make your animation more realistic.*

```
10 REM PROGRAM XI
20 MODE 0
30 WHILE 1=1
40 LOCATE 10,12
50 PRINT CHR$(INT(4*RND(1))+248)
60 FOR delay=1 TO 40:NEXT delay
70 WEND
```

Program XI

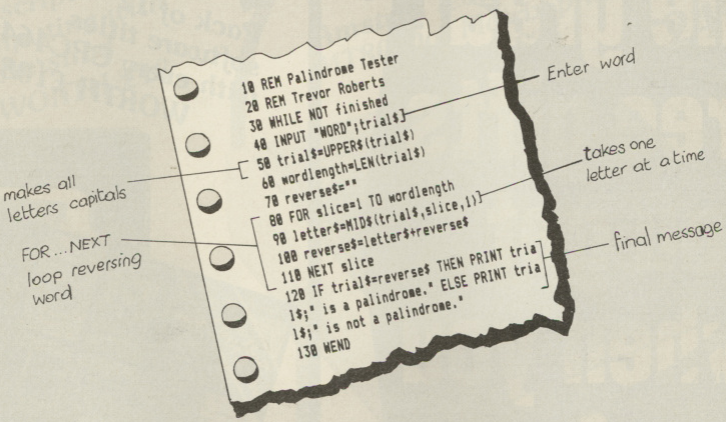
Amstrad Analysis

Palindrome tester

Analysed by Trevor Roberts

DO you know what a palindrome is? No, it's not a camel. It's a word that's spelt the same way forwards and backwards. Rotor is one example while Eve is another.

This month's program uses string handling techniques to see if a word is a palindrome. By the time you've understood it you'll know MID$ and LEN backwards.



10,20 REMs that give information to humans, not the computer.

30–130 These form an endless WHILE . . . WEND loop. Since *finished* is undefined it takes the value 0 (logically false). The NOT inverts this, making it – 1 (true). As *finished* is unaltered by the program the loop continues for ever.

40 Asks for a word which is held in the string variable *trial$*.

50 UPPER$ puts the word to be tested into uppercase letters. This is to avoid occasions where *trial$* is made up of a mixture of upper and lower case letters, for example, Rotor.

60 Finds the length of the test word and stores it in *wordlength*.

70 Initialises a string variable *reverse$*, setting it to the null string, the string with nothing in it.

80–110 These form a FOR . . . NEXT loop which cycles once for each letter in the test word.

90 Each time round the loop MID$ takes one letter from *trial$* and stores it in *letter$*. As *slice* varies from 1 to the length of the word, so eventually each of its letters will have been put into *letter$*.

100 *Letter$* is now added to the front of *reverse$*. By the time the loop has finished each letter will have been copied from *trial$* onto the front of *reverse$*. *reverse$* will be the mirror image of *trial$*.

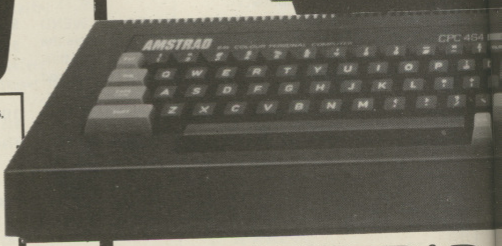
120 Tests *trial$* against *reverse$*. If they match then the word must be a palindrome and the message is printed. If not, the message after the ELSE is printed.

National Micro

Look at the great free gifts you get when you buy from nmc

FREE +

Pack of 12 software titles with every CPC464 WORTH £108



AS SEEN ON TV!

AMSTRAD CPC464

ORDER FORM

Post to:
NATIONAL MICRO CENTRES,
36 St. Petersgate,
Stockport SK1 1HL

Item	Please supply the following:	Qty	£ Total p
		Carriage	
		TOTAL	

Attractive credit terms
Phone for details

Please indicate method of payment:

☐ Cheque payable to National Micro Centres

☐ Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Name .....
Address .....
.....
.....
Tel. No. .....
Signed CA4

Please allow up to 28 days for delivery

Centres

Call in for a demonstration from our friendly sales staff at:

National Micro Centres
36-38 St. Petersgate
Stockport SK1 1HL
or 'phone on:
061-429 8080/8074

FREE

Subscription to Computing with the Amstrad WORTH £12

£359 Colour **£249** Green screen

Mail orders: Carriage £7

FREE

Amstrad lead with every MT80 Dot Matrix printer WORTH £25

£249

Mail orders: Carriage £7

All prices include VAT

We also stock a wide range of other computers:

NEW LOW PRICES

BBC "B"	£359
ELECTRON	£129
COMMODORE 64	£189
SPECTRUM PLUS	£129

Carriage on computers £7

We also stock a wide range of software for ALL micros

Official Amstrad Joystick

***only* £14.95**

Mail orders: Carriage £1

POOLSWINNER

THE ULTIMATE POOLS PREDICTION PROGRAM

Boxed, with detailed instruction booklet

- **MASSIVE DATABASE** Poolswinner is a sophisticated Pools prediction aid. It comes complete with the largest database available – 22000 matches over 10 years. The database updates automatically as results come in.
- **PREDICTS** Not just SCOREDRAWS, but AWAYS, HOMES and NO SCORES.
- **SUCCESSFUL** SELEC guarantee that Poolswinner performs significantly better than chance.
- **ADAPTABLE** Probabilities are given on every fixture – choose as many selections as your need for your bet. The precise prediction formula can be set by the user – you can develop and test your own unique method.
- **SIMPLE DATA ENTRY** All English and Scottish team names are in the program. Simply type in the reference numbers from the screen. Or use FIXGEN to produce fixture list automatically (see below).
- **DISC/MICRODRIVE COMPATIBLE** All versions (except Apple and IBM) are supplied on ta.e, with simple instructions for conversion to disc/microdrive operation.

(This seasons results are supplied with the package so that predictions can start immediately.)

AVAILABLE FOR Spectrum (48k), Commodore 64, VIC 20 (+16K), AMSTRAD CPC464, BBC B, Atari (48K), ZX81 (16K), Dragon, Apple II, IBM pc

PRICE £15.00 (all inclusive)

NEW

FIXGEN 84/5

AT LAST: No more struggling for hours to get the fixture list into the computer. FIXGEN has been programmed with all English and Scottish fixtures for 1984/5. Simply type in the date, and the full fixture list is generated in seconds. Fully compatible with Poolswinner.

POOLSWINNER with FIXGEN £16.00 (all inclusive)
Fixgen alone £5.50

COURSEWINNER v3

THE PUNTERS COMPUTER PROGRAM

Coursewinner is designed to allow you to develop and test your own unique winning system. Using information from daily newspapers or 'Sporting Life', the most important factors can be input and analysed. The program is supplied with a database detailing best trainers and jockeys, and effect of the draw for all British courses. (Flat & National Hunt.)

AVAILABLE FOR Amstrad, Spectrum (48K), Commodore 64, BBC (B), Atari (48K), Apple II

PRICE £12.50 (all inclusive)

AVAILABLE (RETURN OF POST) FROM . . .

VISA phone 24 hrs.

selec SOFTWARE

ACCESS phone 24 hrs.

37 COUNCILLOR LANE, CHEADLE, CHESHIRE. 061-428 7425

PRINTERLAND

UK'S LOWEST PRINTER PRICES
FULL PRINTER SUPPORT FOR THE AMSTRAD
ORDERED TODAY – DELIVERED TOMORROW

DOT MATRIX

	EX VAT	INC VAT
BROTHER HR5	£133.00	£152.95
SHINWA CPA80	£185.00	£212.75
EPSON RX80	£185.00	£212.75
EPSON RX80 F/T +	£212.00	£244.80
EPSON RX100	£347.00	£399.05
EPSON FX80	£314.00	£361.10
EPSON FX100	£425.00	£488.75

NEAR LETTER QUALITY

CANON PW 1080A	£259.00	£297.85
CANON PW 1156	£349.00	£401.35
KAGA KP 810	£259.00	£297.85
KAGA KP 910	£349.00	£401.35

DAISY WHEEL

DAISY STEP 2000	£225.00	£258.75
JUKI 6100	£325.00	£373.75

AMSTRAD ADD ONS

TASWORD TWO	£15.60	£17.95
TASPRINT	£7.74	£8.90
TASCOPY	£7.74	£8.90
PRINTER CABLE	£7.74	£8.90

Educational, G'ment plus O'seas Orders Welcome
Delivery: Printers (Securicor) £10.00
PRINTERLAND, Unit 27, Estate Buildings, Railway Street, Huddersfield HD1 1JP
Showroom open: Mon – Fri 9-6pm Sat morn 9-1pm.
Tel: Huddersfield (0484) 514105 or (0484) 687875

Announcing MAXAM for the AMSTRAD CPC464

SOFTWARE in ROM!

The start of a complete Expansion System . . .

SIDESWAYS ROMS at last!
No more loading . . .
Leaves 40K free!

The perfect system:
* All-powerful Assembler
* Complete Disassembler
* Full screen editor
* Multi-function Adaptor
* Huge expansion potential in one simple unit!

So easy to use and learn . . .

```
10 MEMORY HIMEM-10
20 start=HIMEM+1
30 :ASSEMBLE,start
40 'get start
50 'limit &FFFF
60 'ORG start
70 'CP 10:SCF:RET Z
80 'RST 1,&8752
90 'ORG &BD2B
100'JP start
110'END
```

Software Houses: We have the perfect low-cost system for software in ROM! Talk to us!

Meet MAXAM – a new full-feature no-compromise Assembler/Disassembler/Editor – with a difference. It's in a very full 16K EPROM which plugs directly into the AMSTRAD. No waiting while it loads – it's always there! You can still use the Disc unit. You also get, as a bonus, a new expansion socket for Arnor's new range of Sideways ROM cartridges (containing, for example, our forthcoming Word Processor).

MAXAM uses no BASIC RAM space. It lets you mix BASIC and Machine Code – just like the BEEB! Or, you can assemble direct from the Editor, and you can even use the Editor to edit BASIC programs!

MAXAM is ESSENTIAL software for the AMSTRAD enthusiast.

Cassette (reduced specification): £13.50
Disc: £26.90. All prices include p&p.

MAXAM in ROM £59.90

Arnor LTD
High Quality Software

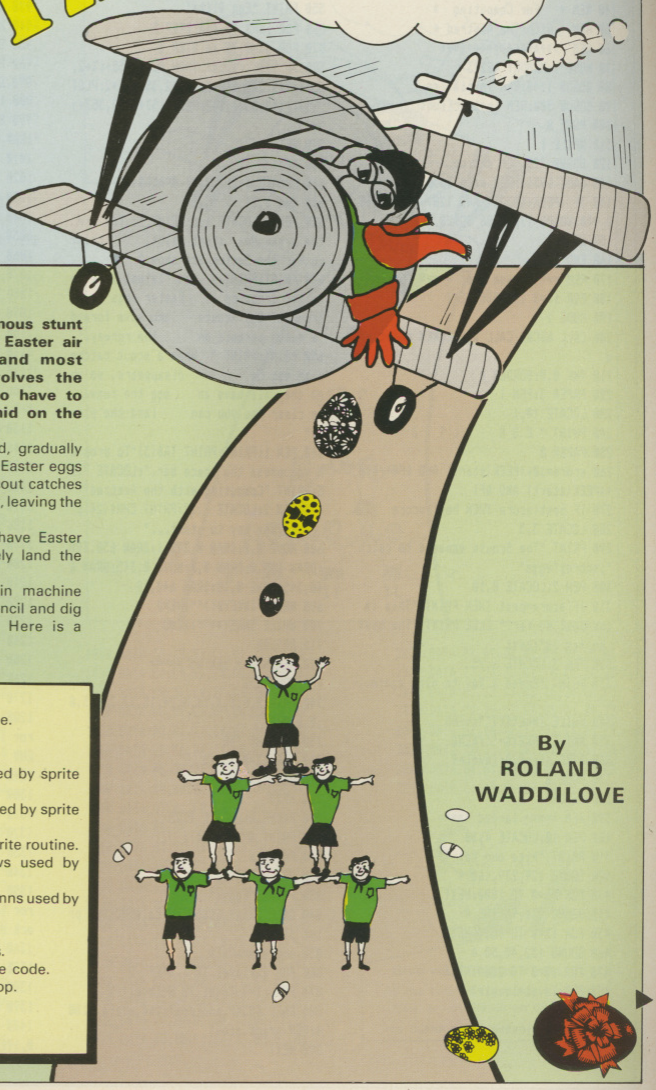
Technical Data

*Super-fast 3000 lines/min assembly *Conditional Assembly *Plain English error messages *Full Expression evaluation *Unrestricted label names *Directives include: ORG, BYTE, WORD, TEXT, RMEM, LET, IF, GET, PUT, LIMIT, CODE, NOCODE, READ. Commands include: LIST, NOLIST, LISTP, TITLE, PAGE, PLEN, WIDTH, DUMP.
*Menu-driven Screen Editor includes move copy and delete block, tabs, search and replace, print all/part of text, Load/Save all/part of text. Disc/ROM version only: Register display, Memory Edit commands, breakpoint, string search in RAM. Link to AMSDOS.

Technical Enqu. 01-852 2174

Cheques/P.O.s to: **Arnor Ltd, PO Box 619, London SE25 6JL. Order Hotline 01-653 1483 (2pm-6pm)**

EGG BLITZ



YOU are Rex the famous stunt pilot training for the Easter air show. Your latest and most dangerous stunt involves the local scout troop who have to form a human pyramid on the runway.

As you pass overhead, gradually losing altitude, you drop Easter eggs to the scouts. When a scout catches an egg he runs off to eat it, leaving the others.

When all the scouts have Easter eggs then you can safely land the plane.

The entire game is in machine code, so sharpen your pencil and dig out your disassembler. Here is a breakdown of the code.

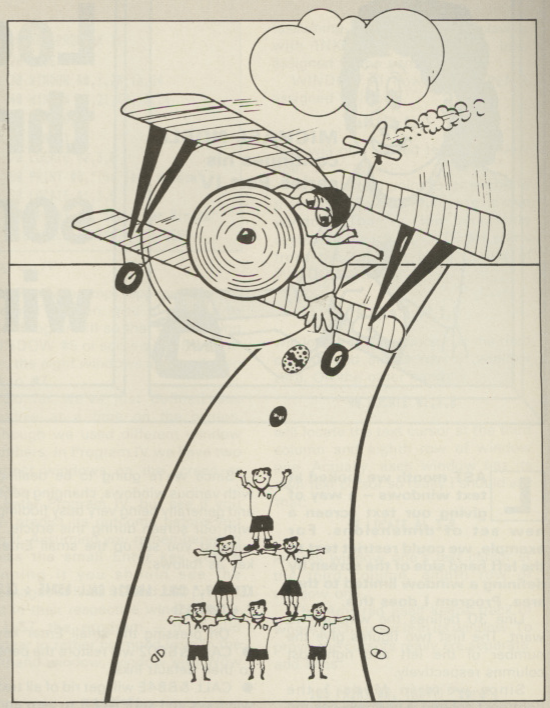
&831D	Score.
&831E	Address of plane.
&8320	Flags.
&8321	Address of egg.
&8323	Old address used by sprite routine.
&8325	New address used by sprite routine.
&8327	Data used by sprite routine.
&8329	Number of rows used by sprite routine.
&832A	Number of columns used by sprite routine.
&832B -&834E	Data for sounds.
&834F	Start of machine code.
&8364	Main control loop.
&84A3	Sprite routine.
&83A6	Move egg.
&8463	Move plane.

By ROLAND WADDILOVE

```
10 REM  ******************
20 REM *    Egg  Blitz    *
30 REM * By R.A.Waddilove *
40 REM *  For Computing   *
50 REM * With The Amstrad *
60 REM  ******************
70 MODE 1:MEMORY &8223
80 GOSUB 1140:REM initialise
90 GOSUB 580:REM instructions
100 MODE 0
110 WHILE 1
120 GOSUB 720:REM set up
130 CALL &8364:REM main loop
140 IF (PEEK(&8320) AND 128) THEN GOSU
B 940:GOSUB 170 ELSE GOSUB 390
150 WEND
160 END
170 REM ===== game over =====
180 FOR i%=0 TO 7000:NEXT
190 MODE 1
200 CALL &BC02:CALL &BB4E:REM reset vd
u
210 INK 0,0:BORDER 0
220 PAPER 3:PEN 1
230 LOCATE 10,1
240 PRINT " E G G    B L I T Z "
250 PAPER 0
260 score=10*(PEEK(start) AND &F0)/&10
+(PEEK(start) AND &F)
270 IF best<score THEN best=score
280 LOCATE 3,5
290 PRINT "The Scouts managed to catch
";score;"eggs"
300 PEN 2:LOCATE 8,10
310 IF score=best THEN PRINT "This is
the best so far!" ELSE PRINT "The best
 so far is";best
320 PEN 3:LOCATE 6,15
330 PRINT "Press a key to play again..
."
340 WHILE INKEY$()"":WEND
350 WHILE INKEY$="":WEND
360 POKE start,0:level=9
370 MODE 0
380 RETURN
390 REM ===== landed =====
400 PEN 1:LOCATE 4,10
410 PRINT "Nice one Rex !"
420 SOUND 129,239,160,4
430 FOR i%=0 TO 1000:NEXT
440 SOUND 130,95,105,4
450 FOR i%=0 TO 1000:NEXT
460 SOUND 132,40,50,4
470 FOR i%=0 TO 2000:NEXT
480 CLS:level=level+2
490 RETURN
500 REM === instructions ===
510 CLS
520 INK 0,6:INK 1,6:INK 3,25
530 BORDER 3
540 PEN 1:LOCATE 1,25
550 PRINT "Egg Blitz"
560 FOR i%=0 TO 144 STEP 2
570 FOR j%=0 TO 16 STEP 2
580 IF TEST(i%,j%) THEN PLOT 155+i%*2,
355+j%*2,2:PLOT 155+i%*2,357+j%*2:PLOT
 157+i%*2,355+j%*2:PLOT 157+i%*2,357+j
%*2
590 NEXT
600 NEXT
610 LOCATE 1,25:PRINT SPACE$(15);
620 INK 1,21
630 PEN 3:LOCATE 1,6:PRINT "  You are
the great Rex, stunt pilot,     traini
ng for the Easter air show."
640 PRINT:PRINT "  Your latest feat in
volves dropping      Easter eggs to a
troop of Boy Scouts     who have formed
 a human pyramid on      the runway."
650 PRINT:PRINT "  When a scout catche
s an egg he          disappears, so wh
en they all have an     egg the runway
is clear and you can     land the plan
e."
660 PEN 1:PRINT:PRINT TAB(3)"To drop a
n egg press the space bar.":LOCATE 7,2
5:PRINT "Computing with the Amstrad"
670 PEN 2:LOCATE 7,22:PRINT CHR$(24);"
 Press any key to start..."
680 MOVE 0,0:DRAW 0,398,1:DRAW 638,398
:DRAW 638,0:DRAW 0,0:MOVE 0,345:DRAW 6
40,345:MOVE 0,18:DRAW 640,18
690 WHILE INKEY$()"":WEND
700 WHILE INKEY$="":WEND
710 RETURN
720 REM ===== set up =====
730 BORDER 1
740 DATA 2,6,1,3,16,3,24,15,12,26,25,0
,0,0,0,0
750 RESTORE 740
760 FOR i%=0 TO 15
770 READ j%:INK i%,j%
780 NEXT
790 PAPER 8:LOCATE 1,25
800 PRINT SPACE$(20);
810 PAPER 0
820 PEN 9:LOCATE 7,1
830 PRINT "Eggs:"
840 CALL &834F,&C050,pdata,&D0C:REM pr
int plane
850 address=&C6F8
860 FOR y%=level TO 1 STEP -2
870 FOR x%=9-level TO 8-level+y%
880 CALL &834F,address+x%*4,mdata,&100
4:REM print scout
890 NEXT
900 address=address-&A0+4
910 NEXT
920 FOR i%=0 TO 1000:NEXT
930 RETURN
940 REM ===== crashed =====
950 SOUND 130,0,0,0
960 FOR i%=1 TO 10
970 LOCATE 1,1:PRINT CHR$(11)
980 INK 1,INT(RND*27)
990 SOUND 4,RND*1000,5,6,0,0,1
1000 FOR j%=1 TO 100:NEXT
1010 LOCATE 1,25:PRINT CHR$(10)
1020 SOUND 4,RND*1000,5,6,0,0,1
1030 FOR j%=1 TO 100:NEXT
1040 NEXT
1050 INK 1,6
1060 PAPER 8:LOCATE 1,25
1070 PRINT SPACE$(20);
1080 PAPER 0
1090 SOUND 1,50,300,5,0,1
1100 FOR i%=0 TO 3000:NEXT
1110 PEN 10:LOCATE 7,3
1120 PRINT "Oh no!":PRINT:PRINT TAB(5)
"You've hit":PRINT:PRINT TAB(6)"a scou
t!"
1130 RETURN
1140 REM === initialise ===
1150 PRINT "Thinking..."
1160 pdata=HIMEM+1:mdata=pdata+156
1170 edata=HIMEM++221:start=HIMEM+250
1180 ENT 1,150,1,2
1190 ENT -2,10,5,1,10,-5,1
1200 ENT 3,200,1,5
1210 ENT 4,50,-4,1
1220 RESTORE 1400
1230 checksum=0
1240 FOR i%=1 TO 236
1250 READ j%:POKE (HIMEM+i%),j%
1260 checksum=checksum+j%
1270 NEXT
1280 IF checksum<>21768 THEN PRINT "Er
ror in character data, please check.":
END
1290 RESTORE 1480
1300 checksum=0
1310 FOR i=&831D TO &8508
1320 READ a$:POKE i,VAL("&"+a$)
1330 checksum=checksum+VAL("&"+a$)
1340 NEXT
1350 IF checksum<>42197 THEN PRINT "Er
ror in machine code,":PRINT "Please ch
eck data statements.":END
1360 POKE start,0
1370 level=9:best=0
1380 RETURN
1390 REM ===== plane =====
1400 DATA 192,0,0,0,0,0,0,240,160,0,0,0,
0,192,128,0,0,0,0,0,240,8,0,0,0,0,0,192,12
```

```
8,0,0,0,0,176,48,0,0,0,0,192,192,0,0,0
,0,48,32,0,0,0,0,192
1410 DATA 192,0,0,0,0,16,0,64,0,0,0,19
2,192,128,0,0,64,204,136,64,128,0,4,13
2,12,72,192,192,192,204,204,192,192,12
8,4,192,192,192,192,12,12,12,12
1420 DATA 12,192,192,4,192,192,192,192
,12,12,12,12,12,72,192,12,192,192,192,
192,192,192,192,192,192,192,192,4,192,
192,192,192,192,192,192,192,192,192,12
8,4,0,0,0,0,0,0,0,4,0,0,0,4,12,0,0,0,0
,0,0,12,8,0,0,0
1430 REM ===== man =====
1440 DATA 32,0,0,32,136,240,160,136,21
6,24,88,136,152,48,48,136,136,36,32,13
6,136,16,0,136,204,196,196,136,204,200
,204,136,68,200,204,0,68,200,204,0,68,
200,204,0,68,136,204,0,68,136,204,0,68
,136,204,0,68,136,204,0,192,128,192,12
8

1450 REM ===== egg =====
1460 DATA 0,188,168,20,188,188,252,252
,252,20,188,188,0,188,168,136
1470 REM === machine code ===
1480 DATA 00,00,00,00,00,00,00,00
1490 DATA 00,00,00,00,00,00,01,00
1500 DATA 02,E8,03,00,04,32,00,02
1510 DATA 00,03,0A,00,00,04,E8,03
1520 DATA 02,00,04,2C,01,00,05,32
1530 DATA 00,02,00,00,D0,07,00,05
1540 DATA 0A,00,DD,4E,00,DD,46,01
1550 DATA DD,5E,02,DD,56,03,DD,6E
1560 DATA 04,DD,66,05,C3,E6,84,CD
1570 DATA 06,B9,F5,21,50,C0,22,1E
1580 DATA 83,21,00,00,22,21,83,3E
1590 DATA 01,32,20,83,CD,63,84,11
1600 DATA 00,00,21,00,00,CD,10,BD
1610 DATA CD,A6,83,CD,0D,BD,01,1B
1620 DATA 00,AF,ED,42,38,F5,3A,20
1630 DATA 83,E6,C0,20,07,3E,42,CD
1640 DATA 1E,BB,28,D8,F1,CD,0C,B9
1650 DATA C9,2A,21,83,7C,B5,20,28
1660 DATA 3A,20,83,EE,01,32,20,83
1670 DATA C8,3E,2F,CD,1E,BB,C8,2A
1680 DATA 1E,83,01,54,38,09,22,21
1690 DATA 83,11,00,83,01,03,05,CD
1700 DATA E6,84,21,34,83,C3,AA,BC
1710 DATA 22,23,83,01,83,38,09,30
1720 DATA 04,01,50,C0,09,3A,C0,82
1730 DATA BE,20,06,2B,2B,2B,BE,28
1740 DATA 2A,3E,83,BE,2B,1D,2A,23
1750 DATA 83,01,00,10,09,30,04,01
1760 DATA 50,C0,09,22,25,83,22,21
1770 DATA 83,11,00,83,01,05,83,CD
1780 DATA A3,84,C9,21,46,83,CD,AA
1790 DATA BC,18,38,E5,21,3D,83,CD
1800 DATA AA,BC,21,01,0C,CD,75,BB
```



```
1810 DATA 3A,1D,83,C6,01,27,32,1D
1820 DATA 83,F5,E6,F0,CB,3F,CB,3F
1830 DATA CB,3F,CB,3F,C6,30,CD,5D
1840 DATA BB,F1,E6,0F,C6,30,CD,5D
1850 DATA BB,E1,11,C8,82,01,04,10
1860 DATA CD,E6,84,11,00,83,01,03
1870 DATA 05,2A,21,83,CD,E6,84,21
1880 DATA 00,00,22,21,83,C9,2A,1E
1890 DATA 83,01,F0,C6,AF,ED,42,38
1900 DATA 08,3A,20,83,F6,40,32,20
1910 DATA 83,21,28,83,CD,AA,BC,2A
1920 DATA 1E,83,22,23,83,23,22,25
1930 DATA 83,22,1E,83,11,24,82,01
1940 DATA 0D,0C,CD,A3,84,2A,1E,83
1950 DATA 01,5B,20,09,3E,00,BE,C8
1960 DATA 21,20,83,CB,FE,C9,ED,43
1970 DATA 29,83,ED,53,27,83,0E,02
1980 DATA CD,19,BD,F3,ED,5B,27,83
1990 DATA 3A,29,83,47,C5,3A,2A,83
2000 DATA 47,2A,23,83,1A,AE,77,23
2010 DATA 13,10,F9,2A,23,83,01,00
2020 DATA 08,09,30,04,01,50,C0,09
2030 DATA 22,23,83,C1,10,DE,2A,25
2040 DATA 83,22,23,83,0D,20,CD,FB
2050 DATA C9,22,23,83,C5,41,2A,23
2060 DATA 83,1A,AE,77,23,13,10,F9
2070 DATA 2A,23,83,01,00,08,09,30
2080 DATA 04,01,50,C0,09,22,23,83
2090 DATA C1,10,E1,C9,00,00,00,00
```

Give your fingers a rest . . .

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

See our special offer on Page 77.

Looking through some new windows

MICHAEL NOELS continues his series Part IV

LAST month we looked at text windows – a way of giving our text screen a new set of dimensions. For example, we could restrict text to the left hand side of the screen by defining a window limited to that area. Program I does this.

Line 30 defines the window we want. The first two figures give the number of the left and rightmost columns respectively.

Since we're in Mode 1 the rightmost column is normally column number 40. In line 30 we've restricted the window to half the screen by setting the rightmost column to 20.

The last two figures of line 30 specify the top and bottom row of the new window respectively. Normally these are rows 1 and 25. As you can see, we've confined the new window between rows 10 and 24. Figure I shows what's happened.

```
10 REM PROGRAM I
20 MODE 1
30 WINDOW 1,20,10,24
40 PAPER 2
50 PEN 3
60 CLS
70 LOCATE 3,8
80 PRINT"The Text Window"
90 WHILE INKEY$ = "": WEND
```

Program I

Since we're going to be dealing with various windows, changing pens and generally being very busy fiddling with our screen during this article, I suggest you set up the small Enter key as follows:

```
KEY 139,"| CALL &BC02: CALL &BB4E : CLS
" + CHR$(13)
```

On pressing the small Enter key

- CALL &BC02 will restore the pens to their default inks.
- CALL &BB4E will get rid of all text windows and set PAPER to zero and PEN to 1.

The effect is to restore our screen to the state it's in at switch-on, before we started meddling. Incidentally, be sure to use capitals where I have – Amstrad Basic won't translate these from lower case as it will other keywords, since they are in quotes.

Set up this key and press it between each program – or your latest program may suffer from the left-over effects of an earlier one.

Now have a look a Program II. In line 30, instead of:

```
WINDOW
```

we've got:

```
WINDOW #0
```

Also PAPER, PEN, CLS and LOCATE have acquired #0 as well. yet if you run Program II you'll see that the output is identical to Program I. So what's going on?

Well, you may remember that last month I hinted that you could have several text windows on the screen at once. In fact, you can have eight distinct windows defined at any one time.

To distinguish between the windows we give them numbers, #0 to #7. We also use these hash (#) numbers to determine the windows that we want our PAPER, PEN, CLS, LOCATE and PRINT commands to effect – as we'll see later.

At the moment, though, you're

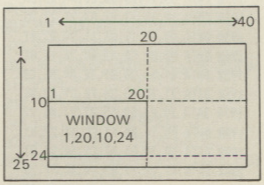


Figure I: WINDOW 1,20,10,24 in Mode 1

```
10 REM PROGRAM II
20 MODE 1
30 WINDOW #0,1,20,10,24
40 PAPER #0,2
50 PEN #0,3
60 CLS #0
70 LOCATE #0,3,8
80 PRINT"The Text Window"
90 WHILE INKEY$ = "": WEND
```

Program II

probably wondering why the #0s in Program II haven't made any difference. The reason is that window zero is the default window. That is, if we just say WINDOW without specifying any hash number, the micro takes it for granted that we mean WINDOW #0.

The same goes for PEN, PAPER. If you don't specify the hash number the Amstrad assumes you mean the command to be effective on WINDOW #0.

Notice that in line 80 of Program II I haven't specified #0 – it doesn't matter of course, since it defaults to window zero anyway. For neatness' sake, though, you might want to replace it with:

```
80 PRINT #0,"The Text Window"
```

The INKEY$ in line 90 of both programs is just to delay the appearance of the Ready prompt.

Program III actually proves that windows other than WINDOW #0 exist. Here all my commands are aimed at window one. Apart from the #1 replacing the #0, the listings of Programs II and III are identical. The output of the program is not, however. Run Program III and see.

```
10 REM PROGRAM III
20 MODE 1
30 WINDOW #1,1,20,10,24
40 PAPER #1,2
50 PEN #1,3
60 CLS #1
70 LOCATE #1,3,8
80 PRINT #1,"The Text Window"
90 WHILE INKEY$ = "": WEND
```

Program III

Notice anything? The window is in the same place, the colour's the same and the message identical – so what's different?

Actually nothing differs until you press a key and drop through the WHILE . . . WEND of line 90. You'll then notice that the Ready prompt appears at the top left hand corner of the screen, not inside the window we've defined, as before.

The reason is that the micro sends all its own messages – such as Ready and Press PLAY – to us via window zero. This is also the window in which our commands to the micro, such as LOAD and SAVE, appear.

As Program III doesn't alter window zero, as do Programs I and II,

```
10 REM PROGRAM IV
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 PEN #0,2
60 PEN #1,3
70 LOCATE #0,3,8
80 PRINT #0,"The Text Window"
90 LOCATE #1,3,8
100 PRINT #1,"The Other Window"
110 WHILE INKEY$ = "": WEND
```

Program IV

the Ready prompt appears on the "normal" screen. You might like to alter Program III so that you're using WINDOW #5 or some such. Remember, the eight windows are numbered #0 to #7.

So far we've just defined one window at a time on the screen, although we used different window numbers. In Program IV we have two distinct windows on the screen at once.

Enter Program IV, but see if you can predict its outcome before you run it. Assuming you remembered to press the small Enter key before running it you should see the messages of lines 80 and 100 printed out in their respective windows.

LIST the program – it should appear in rather cramped form in the lefthand window, window zero. Now try:

```
LIST #1
```

The listing will now appear in window one on the right. So you can LIST to different windows.

Let's have a closer look at Program IV: The two windows (zero and one), defined by lines 30 and 40, are shown in Figure II.

Line 50 chooses PEN 2 for the writing in WINDOW #0 with:

```
50 PEN #0,2
```

This gives us cyan text in window zero

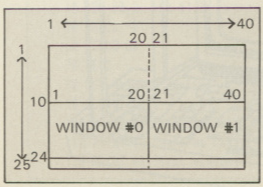


Figure II: Text windows in Program IV

(assuming we haven't been playing with INK and changing the inks assigned to the pens).

WINDOW #1, however, has PEN 3 assigned to it by:

```
60 PEN #1,3
```

This will give us red text in window one.

An important point to remember is that LOCATE works relative to its own window. That is, the coordinate system for each LOCATE #n starts from the top left hand corner of WINDOW #n.

So:

```
70 LOCATE #0,3,8
```

will locate the text cursor at the third column and eighth row of window zero. On the other hand:

```
90 LOCATE #1,3,8
```

will locate the text cursor at the third column and eighth row of window one. Actually, each window has its own text cursor so really I should say that:

```
90 LOCATE #1,3,8
```

locates window one's text cursor at the third column, eighth row of window one.

The Amstrad itself keeps track of where each cursor is in its window. To prove the cursors really are separate, add lines:

```
105 PRINT #0, TAB(9) "zero";
106 PRINT #1, "one"
```

As you'll see, fiddling with the position of the text cursor in one window doesn't affect its position in the other. Mind you, the messages don't really give an idea of the extent of the windows. Adding the following lines should do so:

```
45 PAPER #0,3: CLS #0
46 PAPER #1,1: CLS #1
```

After you've played with Program IV to your heart's content, enter:

```
CLS #2
```

Well, it's not surprising is it? We haven't defined window two!

A point about terminology. So far we've said that PRINT #n prints in WINDOW #n. Actually we should say that PRINT #n prints to STREAM #n (which just happens to flow into WINDOW #n).

Streams are just distinct channels

the micro uses to separate its INPUT and OUTPUT flows. There are nine output streams:

0-7 *Text screen streams – the windows.*
8 *Printer stream.*
9 *Cassette output.*

The input streams are:

0-7 *Keyboard, prompts to relevant windows.*
8 *Keyboard, prompts to printer stream.*
9 *Cassette input.*

Those of you lucky enough to have printers will have used these concepts when listing with:

```
LIST #8
```

Others may have experienced the ideas when using cassette or disc files with:

```
INPUT #9
PRINT #9
```

Anyway, enough of that. In this article we're sticking to windows,

```
10 REM PROGRAM V
20 MODE 0
30 PAPER 10: CLS
40 FOR pane% = 3 TO 17 STEP 2
50 number% =( pane% - 3 ) / 2
60 WINDOW #number%,pane%,pane%+1,1,25
70 PAPER  #number%, number%+1
80 CLS #number%
90 PEN #number%, number%
100 LOCATE #number%,1,12
110 PRINT #number%, number%
120 NEXT pane%
130 WHILE INKEY$= "" :WEND
```

Program V

with output streams 0 to 7. So let's have a look at all eight windows at once. Run Program V.

I leave it to you to work out the gory details – suffice it to say that the windows are labelled 0 to 7.

By the way, the effect of the program is to leave 10 strips on the screen. Don't worry, there are really only eight windows. The left and rightmost strips are left over from when we cleared the whole screen before defining our windows.

If you don't believe me, add the following to the program:

```
140 FOR loop% = 0 TO 7
150 PAPER #loop%, 10
160 CLS #loop%
170 WHILE INKEY$ = "": WEND
180 NEXT loop%
```

The dance of the eight veils, as it's know around the office, will clear a window to the original background colour each time a key is pressed.

What happens if windows overlap? It's quite logical – they share. For example, if windows one and two overlap, clearing either clears the overlapping zone.

Program VI illustrates three overlapping windows. As you'll see, the last one cleared is the only one to survive intact. If you really want to test your skill with windows, try to reverse the way these coloured "cards" are dealt.

Of course the ultimate overlapping is when they nest inside one another. Take a look at Program VII. It should be easy enough to follow.

One of the major uses of streaming

```
10 REM PROGRAM VI
20 MODE 1
30 hor% = 5 : across% =20
40 ver% = 3 : down% = 12
50 FOR pane% = 0 TO 2
60 WINDOW #pane%,hor%,hor%+across%,ver
%,ver%+down%
70 PAPER #pane%, pane% + 1
80 CLS #pane%
90 hor% = hor% + 5
100 ver% = ver% + 4
110 NEXT pane%
120 CALL &BC02
130 CALL &BB4E
```

Program VI

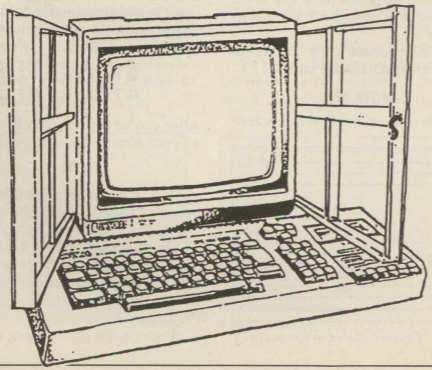
text to windows is to separate input and output – you use one window to display the computer's messages to you, and another for your own replies.

Adventure games often use windows to give a split-screen format, reserving the top of the screen for room descriptions and the bottom for commands and so on.

Program VIII shows an approach to a "guess the number game" using three windows. The first is for your input and the micro's prompts. The other two display your previous guesses. Those greater than the target number are shown in the left hand window and those less in the right.

As it stands, it needs a bit more work on it. However it does show how careful choice of windows can add considerably to a program's appearance and ease of use.

Now suppose we've defined two windows with WINDOW #0 and



```
10 REM PROGRAM VII
20 MODE 1
30 hor% = 4: across% = 32
40 vert% = 4: down% = 16
50 FOR pane% = 2 TO 0 STEP -1
60 WINDOW #pane%, hor%, hor% + across
%, vert%, vert% + down%
70 PAPER #pane%, pane%+1
80 CLS #pane%
90 hor% = hor% + across%/4: across% =
 across% / 2
100 vert% = vert%+ down%/4: down% = d
own%/2
110 NEXT pane%
120 PEN 3
```

Program VII

```
10 REM PROGRAM VIII
20 WHILE -1
30 GOSUB 120:REM Initialise
40 WHILE target% <> number%
50 GOSUB 260:REM input number
60 IF target% = number% THEN GOSUB 350
: REM Correct guess
70 IF target% < number% THEN GOSUB 430
: REM Too high
80 IF target% > number% THEN GOSUB 500
: REM Too low
90 WEND
100 WEND
110 END
120 REM *** Initialise ***
130 REM *** Set Mode  ***
140 REM *** and windows***
150 MODE 1
160 PAPER 2:CLS
170 WINDOW #1,9,16,3,14
180 PAPER #1,1:PEN #1,3: CLS #1
190 WINDOW #2,25,31,3,14
200 PAPER #2,3:PEN #2,1: CLS #2
210 WINDOW #3,4,36,19,24
220 PAPER #3,0:PEN #3,1: CLS #3
230 LOCATE #1,2,12: LOCATE #2,2,12
240 target% = RND(1)*100 + 1
250 RETURN
260 REM *** Input Number ***
270 REM *** Range 1 - 100***
280 CLS #3
290 LOCATE #3,6,3
300 number% = 0
310 WHILE number%<1 OR number%>100
320 INPUT #3,"Number"; number%
330 WEND
340 RETURN
350 REM *** Correct Guess ***
360 CLS #3
370 LOCATE #3,5,2
380 PRINT #3, "Well done." number% "is
 correct."
390 LOCATE #3,3,5
400 PRINT #3, "Press Space for another
 game."
410 WHILE INKEY(47)<>0: WEND
420 RETURN
430 REM *** Too high ***
440 CLS #3
450 LOCATE #3,5,2
460 PRINT #3,"Wrong!" number% "is too
big."
470 PRINT #1, TAB(2) number%
480 GOSUB 570 :REM Get Space
490 RETURN
500 REM *** Too low ***
510 CLS #3
520 LOCATE #3,5,2
530 PRINT #3,"Wrong!" number% "is too
small."
540 PRINT #2, TAB(2) number%
550 GOSUB 570 :REM Get Space
560 RETURN
570 REM *** Wait for Space ***
580 LOCATE #3,3,5
590 PRINT #3, "Press Space for another
 guess."
600 WHILE INKEY(47)<>0:CALL &BB18: WEND
610 RETURN
```

Program VIII

WINDOW #1. All subsequent PRINT #0 and so on refer to window zero while all PRINT #1 refer to window one.

Amstrad Basic has a WINDOW SWAP command that allows you to completely interchange the two windows. For example, if you entered:

```
WINDOW SWAP 0,1
```

the screen area that was labelled as window one magically becomes window zero and vice versa. So, after the swap:

```
PRINT #0,"ABC"
```

will cause ABC to appear in what was the previous window one.

Really, all that's happened is an exchange of labels. Mind you, considering that this will affect all subsequent PEN, PAPER and PRINT statements involving these labels, the effect can be quite impressive!

Program IX shows the results of window swapping. Each time you press space the text in the window interchanges as the labels to the windows are swopped (Line 230).

I can't stress too strongly that it's only the numbers labelling the windows that are swapped. The actual attributes of PEN, PAPER and

```
10 REM PROGRAM IX
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM Message Routine
90 CLS#0:CLS#1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0,"Here we have two"
120 PRINT #0,"text windows."
130 PRINT #0,"This is window 0"
140 PRINT #1,"This is window 1"
150 PRINT #1,"Now let's do a"
160 PRINT #1,"window swap with"
170 PRINT #1
180 PRINT #1,"WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1,"Press Space when"
210 PRINT #1,"you are ready."
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN
```

Program IX

```
10 REM PROGRAM X
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
45 PEN #0,2: PEN #1,3
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM Message Routine
90 CLS#0:CLS#1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0,"Here we have two"
120 PRINT #0,"text windows."
130 PRINT #0,"This is window 0"
140 PRINT #1,"This is window 1"
150 PRINT #1,"Now let's do a"
160 PRINT #1,"window swap with"
170 PRINT #1
180 PRINT #1,"WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1,"Press Space when"
210 PRINT #1,"you are ready."
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN
```

Program X

```
10 REM PROGRAM XI
20 MODE 1
30 wide% = 20
40 WINDOW #0,1,20,10,24
50 WINDOW #1,21,40,10,24
60 PEN #0,2: PEN #1,3
70 text$ = "Centered": GOSUB 160
80 text$ = "in": GOSUB 160
90 text$ = "First": GOSUB 160
100 WINDOW SWAP 0,1
110 text$ = "Centered": GOSUB 160
120 text$ = "in": GOSUB 160
130 text$ = "Second": GOSUB 160
140 WHILE INKEY$="":WEND
150 END
160 REM Centring Routine
170 position% = (wide%-LEN(text$))\2
180 PRINT TAB(position%) text$
190 RETURN
```

Program XI

so on associated with that window still remain with that window under its new label.

Program X illustrates this. Text in the left hand window appears in cyan, whereas that in the right hand window appears in red.

Now you might suppose that when you did the window swap in 230 the colours would also swap. Not so. It's only the labelling number that changes, not the physical characteristics – size, pen/paper colours, cursor position and so on remain unchanged for that window.

One important use of WINDOW SWAP is so that our windows can share subroutines. The subroutine beginning at line 160 of Program XI contains a rather primitive text centring routine which operates on window zero by default.

Having defined two windows (lines 40,50) we can then use this routine to centre text in both. It works straight away for the text in window zero (lines 70 to 90).

To get it to operate on the second window we perform a window swap line 100 so that it's now referred to as WINDOW #0. Then we can centre our text as normal (lines 110 to 130).

So far I've only swapped between one and zero. Of course WINDOW SWAP works between any pair of windows. For example you could add the following line to Program VIII:

```
225 WINDOW SWAP 1,2
```

Now the guesses greater than the target number will appear in the right hand window and those that are less in the left hand one.

If you are feeling adventurous you could try window swapping with an undefined window!

That's all for this month. Next it's the graphics screen.

MYRDDIN FLIGHT SIMULATION

Real time simulation with 3D Graphics

Uses massive 64,000 x 64,000 LONGITUDE & LATITUDE. flying area, making each flight completely different. Developed under pilot instruction to give realistic flight effect. View through cockpit gives moving 3D graphics.

Comprehensive instrument panel with moving needle meters and digital displays.

15 Aircraft types from 100-500 knots airspeed.

3 runways available for refuelling, take-off and landing ... ground and landmark orientation correct with all flying attitudes (rolls etc).

3D landmarks you can fly around.

Hours of absorbing skilful involvement with this superb simulation ... complete with manual and fully detailed chart of landmarks and airfields.

JOYSTICK OR KEYBOARD OPERATION

MYRDDIN P.O. BOX 61, Swindon, Wilts. Tel. (0793) 40661.

Please supply .....

Cheque for total amount enclosed .....

Name .....

Address .....

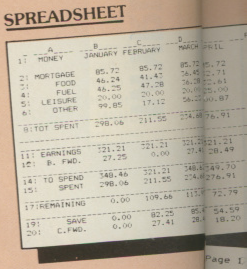
Prices incl. p&p. UK only

MYRDDIN FLIGHT SIMULATION AMSTRAD CPC 64

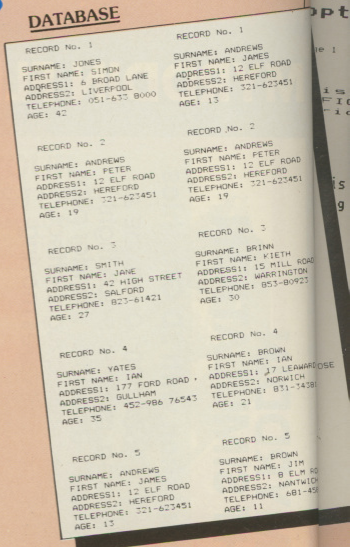
QUICK TO LEAR

THAT'S...

MINI OFFICE



JUST LOOK WHAT THIS PACKAGE CAN DO!



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

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

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

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

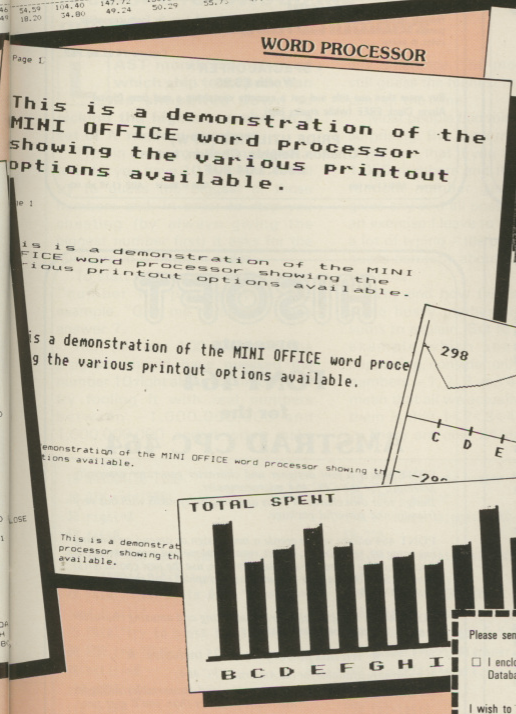
...and it's all at t price of just £

RN, EASY TO USE

Specially written for your AMSTRAD CPC 464

GRAPHICS

WORD PROCESSOR



at the unbelievable

t £5.95

CASSETTE
3" DISC £9.95

DATABASE SOFTWARE

Please send me a copy of Mini Office for my Amstrad CPC464

☐ I enclose my cheque made payable to Database Publications Ltd.

☐ Amstrad cassette £5.95
☐ Amstrad 3" disc £9.95
Please tick box

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

Signed _____

Name _____

Address _____

POST TO: Mini-Office Offer, Database Software, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

A3

GUESS WORK!

Aleatoire follows up the question of random numbers

LAST month's problem of which ship to torpedo can be simplified down to picking the highest number in a set of N random numbers.

Type in Program I. A simple test is to tell it you have 10 numbers. It will try to guess the biggest of those numbers and, in order to stop you cheating (by always giving the biggest number first) it asks for the numbers in a random order.

To start with, whenever it asks for "number N" just reply N. For example, "Give me number 7", then answer 7.

Run this test 100 times and the program should have guessed the number 10 right about 27 times. Now try fooling it with real numbers between −1,000,000,000 and 1,000,000,000 and it won't make any difference. The program should still guess the highest number 27 per cent of the time.

Most people find this "prescience" surprising. But even more surprising is the fact that if you tell it you have 1,000 numbers and then link it to a random number generator which gives any and all possible numbers – an exercise I leave to you but it saves a lot of typing – the program will still guess correctly about 27 per cent of the time.

Why and how this is so requires some heavy mathematics and hard sums to explain. But to give a simple example which shows what is happening, consider only four pseudo numbers – 1, 2, 3 and 4. By pseudo I mean that all we actually know about them is that 1<2<3<4.

There are only 24 different ways that these numbers can be ordered. These are:

1		1	2	3	4
2		1	2	4	3
3		1	3	2	4
4		1	3	4	2
5	*	1	4	2	3
6	*	1	4	3	2
7		2	1	3	4
8	*	2	1	4	3
9		2	3	1	4
10		2	3	4	1
11	*	2	4	1	3
12	*	2	4	3	1
13	*	3	1	2	4
14	*	3	1	4	2
15	*	3	2	1	4
16	*	3	2	4	1
17	*	3	4	1	2
18	*	3	4	2	1
19		4	1	2	3
20		4	1	3	2
21		4	2	1	3
22		4	2	3	1
23		4	3	1	2
24		4	3	2	1

What the program does in the case of N=4 is to ignore the first 4/2.71828 numbers, that is the first number, and then guess at the next number that is bigger than any previous number.

Following this rule it will always succeed in the sequences marked * – 11 times in 24 on average, so it will guess the highest of four random numbers nearly half the time.

As the set of numbers N gets larger this success rate drops, but will never go below 27.1828 per cent no matter how big N becomes. Incidentally, 2.71828 is the transcendental number "e" or EXP(1).

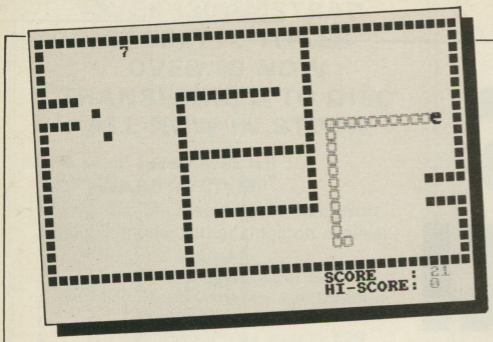
In the above example the computer can pick the right number about 45 per cent of the time. This may not impress some people, so consider the following trick. Shuffle a deck of 52 playing cards and invite someone to pick five cards randomly (don't let the computer see which cards he chooses).

From these five cards ask the person to select just one card. You now tell the computer what the other four cards are and it tells him the single card he selected – and now it's right every time.

How does it "know" the unknown card?

```
10 RANDOMIZE TIME
20 DIM nxt(1000)
30 right=0
40 tst=0
50 INPUT"How many numbers";n
60 pass=n/2.71828
70 PRINT:PRINT"Write down "n" numbers
 and I will"
75 PRINT"try to guess the biggest."
80 FOR i=1 TO n
90 nxt(i)=0
100 NEXT i
110 try=0
120 biggest=-2.14748365E+09
130 try=try+1
140 choice=INT(RND*n)+1
150 IF nxt(choice)>0 THEN 140
160 nxt(choice)=1
170 PRINT"Give me number ";choice
180 INPUT number
190 IF number>biggest AND try>pass TH
EN 220
200 IF number>biggest THEN biggest=nu
mber
210 IF try<n THEN 130
220 PRINT"I choose that number"
230 INPUT"Am I right ? Y/N";answer$
235 answer$=UPPER$(answer$)
240 IF LEFT$(answer$,1)="Y" THEN righ
t=right+1
250 tst=tst+1
260 PRINT"My average is ";right/tst*1
00;"%"
265 PRINT:PRINT"--- Next test ---"
270 GOTO 70
```

Program I



Mad Adder!

THERE'S nothing like hunger to make an adder madder. So to keep the peace try steering this one in his relentless search for food.

The screen is divided into two enclosed areas, access from one to the other being through a gate. Colliding with any part of the wall is fatal, and any snake who tries eating itself is bound to die!

The more he eats, the longer he grows and the more points you score. But beware, the food deteriorates with time, eventually to become another obstacle to be avoided at all cost.

The snake is directed by either the A, Z, <. > keys or joystick.

If you wish to alter the keys change the numbers in the brackets in lines 27-30, referring to page 16 Appendix III of the User Instructions.

The program is provided with plenty of REM statements but these can be omitted if desired.

VARIABLES

SP$	Stores the character associated with each screen position.
H$(P)	Snake's head depending on direction of travel.
F$(N)	Food.
X,Y	General screen coordinates.
FX,FY	Coordinates of food.
BX,BY	Coordinates of each body cell.
DX,DY	Coordinates of each additional obstacle.
L	Length of snake.
SC	Score.
HS	Hi-Score.
I,J,K	General variables.

Line	
19	Prints the snake's head, enables the interrupt to decrease food value if appropriate.
20	Checks to see if food is eaten.
21	Checks for collision.
22	Deletes head.
23	If there is no body cell to print clears variable containing head position and skips to line 27.
24	Prints new body cell.
25	Stores the position of each body cell for deletion later.
26	When snake is long enough branches to subroutine to delete the end body cell.
27-36	Sets new coordinates for the head according to the key/joystick command.
37	Returns to start.

```
1 REM
2 REM            MAD  ADDER
3 REM            by M.Andrews
4 REM   ******************************
5 REM  (c) Computing with the Amstrad
6 REM   ******************************
7 REM
8 REM
9 MODE 1:BORDER 10:INK 0,10:INK 1,24:I
NK 2,26:INK 3,6
10 DIM SP$(40,21),F$(9),H$(4),BX(200),
BY(200),DX(50),DY(50)
11 GOSUB 44:REM Define characters
12 GOSUB 57:REM Draw screen
13 GOSUB 78:REM Initialise
14 GOSUB 88:REM Display food
15 EVERY 150 GOSUB 96:REM Decrease foo
d
16 REM      ++++++++++++++++++++
17 REM      +   MAIN  PROGRAM   +
18 REM      ++++++++++++++++++++
19 LOCATE X,Y:PEN 3:PRINT H$(P):EI:DI
20 IF SP$(X,Y)=F$(N) THEN GOSUB 109
21 IF SP$(X,Y)=A$ OR SP$(X,Y)=B$ THEN
115
22 FOR T=1 TO 100:NEXT:LOCATE X,Y:PRIN
T " "
23 IF SC=0 THEN SP$(X,Y)="":GOTO 27
24 SP$(X,Y)=B$:LOCATE X,Y:PEN 2:PRINT
B$
25 BX(I)=X:BY(I)=Y:L=L+1:I=I+1:IF I=20
0 THEN I=0
26 IF L>SC THEN GOSUB 38
27 IF INKEY(31)=0 OR JOY(0)=8 THEN P=1
28 IF INKEY(39)=0 OR JOY(0)=4 THEN P=2
29 IF INKEY(69)=0 OR JOY(0)=1 THEN P=3
30 IF INKEY(71)=0 OR JOY(0)=2 THEN P=4
31 IF P=1 THEN X=X+1
32 IF P=2 THEN X=X-1
33 IF P=3 THEN Y=Y-1
34 IF P=4 THEN Y=Y+1
35 IF X=0 THEN X=40:Y=15
36 IF X=41 THEN X=1:Y=7
37 GOTO 19
38 SP$(BX(J),BY(J))="":LOCATE BX(J),BY
(J):PRINT " "
39 L=L-1:J=J+1:IF J=200 THEN J=0
40 RETURN
41 REM      ++++++++++++++++++++
42 REM      + DEFINE CHARACTERS +
43 REM      ++++++++++++++++++++
44 SYMBOL 242,&7E,&F7,&FF,&FF,&C0,&F0,
&7F,&3E
45 SYMBOL 243,&7E,&EF,&FF,&FF,&3,&F,&F
E,&7C
46 SYMBOL 244,&72,&F3,&F3,&B3,&F7,&F7,
&FE,&7C
47 SYMBOL 245,&3E,&7F,&EF,&EF,&CD,&CF,
&CF,&4E
48 SYMBOL 246,&7E,&FF,&C3,&C3,&C3,&C3,
&FF,&7E
49 A$=CHR$(233):B$=CHR$(246):C$=CHR$(2
41)
50 H$(1)=CHR$(242):H$(2)=CHR$(243)
51 H$(3)=CHR$(244):H$(4)=CHR$(245)
52 F$(1)="1":F$(2)="2":F$(3)="3":F$(4)
="4":F$(5)="5"
53 F$(6)="6":F$(7)="7":F$(8)="8":F$(9)
="9":RETURN
54 REM      ++++++++++++++++++++
55 REM      +   DRAW SCREEN    +
56 REM      ++++++++++++++++++++
57 FOR X= 1 TO 40:Y= 1:GOSUB 73:NEXT
58 FOR Y= 2 TO  6:X= 1:GOSUB 73:NEXT
59 FOR X= 2 TO  4:Y= 6:GOSUB 73:NEXT
60 FOR X= 1 TO  4:Y= 8:GOSUB 73:NEXT
61 FOR Y= 9 TO 20:X= 1:GOSUB 73:NEXT
62 FOR Y= 2 TO 14:X=40:GOSUB 73:NEXT
63 FOR X=37 TO 39:Y=14:GOSUB 73:NEXT
64 FOR X=37 TO 39:Y=16:GOSUB 73:NEXT
65 FOR Y=16 TO 20:X=40:GOSUB 73:NEXT
66 FOR X= 1 TO 40:Y=21:GOSUB 73:NEXT
67 FOR X=15 TO 25:Y=11:GOSUB 73:NEXT
68 FOR X=15 TO 23:Y= 6:GOSUB 73:NEXT
69 FOR Y= 6 TO 20:X=14:GOSUB 73:NEXT
70 FOR X=17 TO 25:Y=16:GOSUB 73:NEXT
71 FOR Y= 2 TO 16:X=26:GOSUB 73:NEXT
72 RETURN
73 SP$(X,Y)=A$:LOCATE X,Y:PEN 1:PRINT
a$
74 RETURN
75 REM      ++++++++++++++++++++
76 REM      +   INITIALISE     +
77 REM      ++++++++++++++++++++
78 PEN 1:IF SC>HS THEN HS=SC
79 LOCATE 27,22:PRINT "SCORE   :    "
80 LOCATE 27,23:PRINT "HI-SCORE:    "
81 LOCATE 36,23:PEN 2:PRINT HS
82 X=33:Y=6:P=INT(4*RND(1)+1)
83 SP$(X,Y)=H$(P):L=0:I=0:J=0:K=0:SC=0
84 RETURN
85 REM      ++++++++++++++++++++
86 REM      +  DISPLAY  FOOD   +
87 REM      ++++++++++++++++++++
88 N=INT(7*RND(1)+3)
89 FX=INT(38*RND(1)+2):FY=INT(18*RND(1
)+2)
90 IF SP$(FX,FY)<>"" THEN 89
91 SP$(FX,FY)=F$(N):LOCATE FX,FY:PEN 3
:PRINT F$(N)
92 RETURN
93 REM      ++++++++++++++++++++
94 REM      +  DECREASE FOOD   +
95 REM      ++++++++++++++++++++
96 N=N-1:IF N>0 THEN SOUND 1,500,10,5
97 WHILE N=0
98 SP$(FX,FY)=A$:LOCATE FX,FY:PEN 1:PR
INT A$
99 SOUND 1,1000,5,7
100 K=K+1:DX(K)=FX:DY(K)=FY:N=INT(7*RN
D(1)+3)
101 FX=INT(38*RND(1)+2):FY=INT(18*RND(
1)+2)
102 IF SP$(FX,FY)<>"" THEN 101
103 WEND
104 SP$(FX,FY)=F$(N):LOCATE FX,FY:PEN
3:PRINT F$(N)
105 RETURN
106 REM      ++++++++++++++++++++
107 REM      +    EAT  FOOD     +
108 REM      ++++++++++++++++++++
109 SOUND 1,20,10,5:SP$(FX,FY)=H$(P)
110 SC=SC+N:LOCATE 36,22:PEN 2:PRINT S
C
111 GOSUB 88:RETURN
112 REM      ++++++++++++++++++++
113 REM      +       CRASH      +
114 REM      ++++++++++++++++++++
115 INK 0,7,15:BORDER 7,15
116 FOR T=1 TO 500:SOUND 132,1000,5,5:
NEXT
117 INK 0,10:BORDER 10
118 LOCATE 10,25:PEN 3:PRINT C$;"  S P
 A C E   B A R  ";C$
119 REM      ++++++++++++++++++++
120 REM      +    PLAY  AGAIN    +
121 REM      ++++++++++++++++++++
122 W$=INKEY$:IF W$<>" " THEN 122
123 LOCATE 10,25:PRINT SPC(22)
124 SP$(FX,FY)="":LOCATE FX,FY:PRINT "
 "
125 IF L>0 THEN GOSUB 129
126 IF SP$(X,Y)=A$ THEN LOCATE X,Y:PEN
 1:PRINT A$:PRINT CHR$(7)
127 IF K>0 THEN GOSUB 131
128 FOR T=1 TO 500:NEXT:GOTO 13
129 FOR R=1 TO L:GOSUB 38:PRINT CHR$(7
)
130 FOR T=1 TO 40:NEXT T,R:RETURN
131 FOR R=1 TO K
132 SP$(DX(R),DY(R))="":LOCATE DX(R),D
Y(R):PRINT " "
133 NEXT:RETURN
```

Give your fingers a rest . . .

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

See our special offer on Page 77.

Enlist the aid of this Pilot to guide you through a quiz

By GORDON MILLS

IF ever a program could be described as of mixed parentage, our version of Pilot can. The form books would describe it as "Pilot, out of N. Dealey, via Chuck Carpenter, adapted by Gordon Mills for the Apple and BBC Micro and translated for the Amstrad by the A Team".

It is rather surprising that no version of Pilot seems to be available for the Amstrad CPC464 despite the fact that the Pilot language was originally written for teachers to help them produce computer-aided learning programs.

Perhaps one reason for this is that the "Super" Pilot such as that available on the Apple is even more complicated to learn than Basic.

One advantage of a simple Pilot is that beginners in programming can be more easily initiated into the techniques of simple program structures. The disadvantages are that it does not allow subroutines/procedures nor any form of mathematical calculation.

However, for the teacher who wishes to make up simple quizzes or tutorial material on a question/response basis, this simple Pilot may save considerable time when compared with a similar program written in Basic.

The version described here is based on the Apple version (lines 500-4900 are almost identical). It requires only a few simple command letters with each command followed by a colon as shown in Table I.

After typing in Program II this should be saved. On running the program, to write a new one press 1 (new program). Then enter the name which you wish to give it. After a few seconds, a zero followed by a question mark will appear.

T:	Text line.
A:	Accept user input.
M:	List of strings to check if there is a Match with the last user input.
J:	Jump to another line.
C:	Clear the screen.
R:	Remark or comment line.
E:	(or END:) End the Pilot program.
S:	Allows a score to be kept (that is requires a number).
DONE	To end writing a program and return to menu.

Table I: Table of commands

This is your cue to start typing in Pilot. It is important not to put in unnecessary spaces and to ensure all letters are capitals.

The text command T: can be used to put any normal keyboard character on screen. Crude graphics can be displayed by using *, + and a full-stop to build patterns spread over a number of lines.

All the main commands can be modified by Y or N immediately following the command symbols and before the colon. This causes the command to operate only if:

The modifier is Y and the last Match is positive,

Or:

The modifier is N and the last Match is negative.

Additional symbols used are * to precede labels of lines to which the programmer wishes to jump and $ to precede string labels in lines accepting user input – these can be used later in the program in text lines. There has to be a space between the colon : and the $ of the label.

Examples of their use are given in Program I or Program Ia and in the instructions contained within Program II.

When using the Match command more than one string can be checked providing each is separated by a comma. Note also that the last symbol on a Match line must always be a single quote (').

As there is a limitation of 33 characters per line, if the number of matches to be checked is greater than can be accommodated on one line, then the continuation Match lines must start with MN: as shown in line 6 of Program 1.

Apart from improving the integrity and structure of the Carpenter program, additional features include:

- A comprehensive menu: on running the program, press 1 to start a new program. All other commands are self-explanatory.
- Improved editing by allowing the insertion and deletion of lines in a Pilot program.
- An option for the Pilot program to be listed on a printer.
- Paged screen listing of the Pilot program.
- Maximum number of Pilot program lines is 400. This could be enlarged further if required by increasing the value of MX in line 130. Alternatively, initialising/editing delays could be reduced by decreasing MX.
- Instructions and commands together with a sample Pilot program included within the Basic program.
- The addition of a Score command, S: which can be followed by any number (positive or negative). If the

Score number is less than 999, then that number is added to the current score: if 999 or greater, the score is reset to zero. At the conclusion of the program, the score is printed out.

- A facility for Pilot programs to be saved to and loaded from cassette.

When writing or editing a program, to tell the micro that you have finished type DONE on the next line (without a colon) and you will be returned to the menu. Don't forget that you will need the command E: to end the program you are writing.

One of the simplest uses for Pilot in education is to present simple knowledge tests. Program I is intended to illustrate how program structures can be developed from the simple to the more complex, thereby providing an ideal teaching medium for beginners.

Question 1 allows one attempt only. Question 2 shows how a multiple choice question can be written.

Questions 3 and 4 allow the user any number of attempts but it is important to include, as in Question 4, the option (?) to escape from what would be an endless loop if the answer is not automatically given after the user response.

Question 5 shows how to obtain two answers from one question. This involves a much more complex structure than the previous questions. The five types of questions demonstrated in Program I can serve as a model for similar questions in any discipline.

If you find Program I complex try Program Ia. This has only three questions and contains only one jump.

This Mini Pilot interpreter, although written in Basic, is surprisingly fast.

The only delays – of several seconds – occur on initialising and when inserting blank lines or deleting lines (under EDIT).

Also, the modular construction allows the expert programmer to readily experiment with additional commands if desired.

Now is your chance to experiment with a simplified Pilot. It may be that more elaborate versions are on the way, but this is a cheap method of finding out whether you think a more expensive Pilot with graphics and mathematical facilities could be a worthwhile acquisition.

I hope you and your friends – or pupils – don't groan too much at the riddles in Program I.

Program I

```
0 *1
1 C:
2 T:IF AN INDIAN WOMAN IS A SQUAW
3 T:WHAT IS AN INDIAN BABY?
4 A:
5 M:PAPOOSE,PAPPOOSE,PAPPOOS,
6 MN:SQUAWKER,
7 SY:3
8 TY:GOOD
9 T:THE JOKE ANSWER IS SQUAWKER
10 T:NOW PRESS RETURN
11 A:
12 *2
13 C:
14 T:HOW MANY KINGS HAVE BEEN
15 T:CROWNED IN ENGLAND SINCE
16 T:1500 AD ?
17 T:
18 T:A - ONE
19 T:B - 15
20 T:C - 16
21 T:
22 T:TYPE A OR B OR C
23 A:
24 M:A,
25 TN:ANSWER A IS CORRECT BECAUSE
26 T:ONLY JAMES I WAS ALREADY KING!
27 SY:2
28 SN:-1
29 T:PRESS RETURN
30 A:
31 *3
32 C:
33 T:WHAT WAS THE LARGEST ISLAND
34 T:BEFORE AUSTRALIA WAS
35 T:DISCOVERED ?
36 A: $ANSWER
37 M:AUSTRALIA,'
38 TN: $ANSWER IS WRONG
39 SN:-1
40 T:CORRECT IS STILL AUSTRALIA!
41 SY:3
42 TN:NOW PRESS RETURN
43 AN:
44 JN:*3
45 T:GOOD. NOW PRESS RETURN
46 A:
47 *4
48 C:
49 T:IF YOU REALLY DON'T KNOW THE
50 T:ANSWER TO THE NEXT QUESTION,
51 T:TYPE ?
52 *4A
53 T:MAKE ONE WORD FROM THE LETTERS
54 T:NEW DOOR
55 A:
56 M:ONE WORD,ONEWORD,'
57 TY:GOOD. NOW PRESS RETURN
58 AY:
59 SY:3
60 JY:*5
61 M:?,'
62 SY:-1
63 SN:-2
64 TN:TRY AGAIN. PRESS RETURN
65 AN:
66 JN:*4
67 TY:THE ANSWER IS THE 2ND AND 3RD
68 TY:WORDS OF THE RIDDLE!
69 TY:PRESS RETURN
70 A:
71 C:
72 JY:*4A
73 *5
74 R:A QUESTION WITH TWO ANSWERS
75 C:
76 T:WHAT CAN BE ADDED TO NINE TO
77 T:MAKE SIX ?
78 T:TYPE ONE OF TWO POSSIBLE
79 T:ANSWERS
80 A:
81 M:-3,- 3,MINUS3,MINUS 3,'
82 SY:1
83 JN:*5A
84 T:GOOD. NOW TYPE ANOTHER
85 A:
86 M:S,'
87 JN:*5B
88 *5C
89 SY:2
90 T:EXCELLENT. MAX SCORE IS 14
91 T:PRESS RETURN TO FIND YOURS
92 A:
93 EY:
94 *5B
95 T:CORRECT IS S. (S + IX = SIX)
96 *5D
97 SN:-2
98 T:PRESS RETURN
99 A:
100 J:*5
101 *5A
102 M:S,'
103 SN:-2
104 SY:1
105 TN:TRY AGAIN
106 JN:*5
107 T:GOOD. WHAT IS THE OTHER ?
```

```
100 A:
109 M:-3,- 3,MINUS3,MINUS 3,'
110 JY:#5C
111 T:CORRECT IS -3 (MINUS 3)
112 J:#5D
```

Program Ia

```
0 R:QUIZ
1 C:
2 #FIRST
3 T:WHAT IS THE CUBE ROOT
4 T:OF 125 ?
5 A:
6 M:FIVE,5,'
7 TN:GO AND GET A CALCULATOR !
8 JN:#FIRST
9 T:WELL DONE. BRAIN BOX.
10 T:PRESS RETURN TO CONTINUE
11 A:
12 C:
13 T:WHICH IS THE ODD ONE OUT?
14 T:A - 1
15 T:B - 4
16 T:C - 8
17 T:D - 14
18 T:E - 19
19 T:
20 T:ENTER A,B,C,D OR E
21 A:
22 M:D,'
23 TN:WRONG !
24 TN:THE ODD ONE OUT IS D.
25 TN:IT SHOULD HAVE BEEN 13.
26 TY:CORRECT
27 TY:SEE IF YOU CAN GET THE
28 TY:NEXT QUESTION RIGHT.
29 T:PRESS RETURN TO CONTINUE
30 A:
31 C:
32 T:WHAT IS THE DECIMAL EQUIVALENT
33 T:OF THE BINARY NUMBER 1001101
34 T:(HINT: IT'S +VE)
35 A:
36 M:77,'
37 TY:CONGRATULATIONS !
38 TN:HARD LUCK, THE ANSWER IS 77.
39 T:
40 T:END OF TEST.....
41 T:.....
42 T:
43 END:
```

Program II

```
10 REM MINI PILOT
20 REM (C) Computing with
30 REM     the Amstrad.
100 REM INITIALISE
105 OPENOUT"dummy"
110 MEMORY HIMEM-1
120 CLOSEOUT
125 D0$=":":D1$=",":CP$="":M$="N"
130 MX=399:LL=33:R=19:JF=0:LF=7:NL=20:
Q=0:SC=0
140 DIM P$(1,MX),I$(1,R),S$(7),M$(9)
150 S$=STRING$(35,"A"):A$=S$
200 REM DATA
210 FOR I=1 TO 9:READ M$(I):NEXT
220 DATA NEW PROGRAM,LIST,EDIT,RUN,REP
LACE,SAVE,LOAD,END,INSTRUCTIONS
230 FOR I=1 TO 7:READ S$(I):NEXT
240 DATA A,T,M,J,C,E,S
499 REM CONTROL SECTION
500 GOSUB 1000:GOTO 600
510 GOSUB 1100:GOTO 600
520 GOSUB 1200:GOTO 600
530 GOSUB 1300:GOTO 600
540 GOSUB 1400:GOTO 600
550 GOSUB 1500:GOTO 600
560 GOSUB 1600:GOTO 600
570 GOSUB 1700:GOTO 600
580 GOSUB 1800:GOTO 600
590 GOSUB 1900:GOTO 600
600 ON Z GOTO 510,520,530,540,550,560,
570,580,590,500
999 REM TITLE
1000 GOSUB 5000:V=1:H=1:GOSUB 5200:GOS
UB 4900
1010 V=2:H=9:GOSUB 5200:PRINT"MINI-PIL
OT (GMM)"
1020 V=3:H=1:GOSUB 5200:GOSUB 4900
1030 FOR I=1 TO 9:V=4+I:H=10:GOSUB 520
0:PRINT I
1040 H=14:GOSUB 5200:PRINT M$(I):NEXT
1050 V=23:H=1:GOSUB 5200:PRINT"PRESS N
UMBER OF ITEM REQUIRED";
1060 GOSUB 6000:IF G<49 OR G>57 THEN G
OTO 1060
1070 Z=VAL(A$):RETURN
1100 REM NEW
1105 LT=LL:Z=1:V=1:H=1:IF A$="EDIT" TH
EN V=3:GOTO 1125
1110 CP$="":GOSUB 5000:C=0:GOSUB 5200:
PRINT"PROGRAM NAME? ";:LL=LF:GOSUB 610
0
1115 CP$=A$:IF CP$="" THEN 1110
1120 LL=LT:GOSUB 4500
1125 V=V+1:FOR A =C TO MX: GOSUB 4800:
IF A=MX THEN GOSUB 4300
1130 PRINT A"?";:GOSUB 6100
1135 IF A$="EDIT" THEN Z=3:A=MX:GOTO 1
175
1140 IF A$="DONE" THEN Z=10:A=MX:GOTO
1175
1145 IF LEFT$(A$,1)="*" OR LEFT$(A$,1)
="!" THEN P$(0,A)="":P$(1,A)=A$:GOTO 1
175
1150 M=0:FOR B=1 TO LEN (A$)
1155 IF MID$(A$,B,1)<>D0$ THEN GOTO 11
65
1160 P$(0,A)=LEFT$(A$,B-1):P$(1,A)=MID
$(A$,B+1,LEN(A$)-B+2):B=LEN(A$):M=1
1165 NEXT B:IF M THEN GOTO 1175
1170 V=V+1:GOSUB 5200:PRINT"COMMAND ER
ROR":A=A-1:V=V+1:GOSUB 5200
1175 NEXT A:RETURN
1200 REM LIST
1205 Z=10:IF CP$="" THEN RETURN
1210 GOSUB 5000:PRINT"PRESS S FOR SCRE
EN/P FOR PRINTER";:srn=0
1215 GOSUB 6000:IF G<>80 AND G<>83 THE
N GOTO 1215
1220 GOSUB 5000:IF G=80 THEN GOSUB 650
0
1225 V=1:H=8:GOSUB 5200:PRINT#srn,CP$;
NC=0:FOR A=0 TO MX:NC=NC+1
1230 IF P$(0,A)="" AND P$(1,A)="" THEN
 A=MX:GOTO 1255
1235 GOSUB 4800:H=1:GOSUB 5200
1240 IF P$(0,A)<>"" THEN PRINT#srn,A;:
H=7:GOSUB 5200:PRINT#srn,P$(0,A);D0$;P
$(1,A):GOTO 1250
1245 PRINT#srn,A;:H=6:GOSUB 5200:PRINT
#srn,P$(1,A)
1250 IF G<>80 THEN GOSUB 4600
1255 NEXT A:GOSUB 6600:GOSUB 4700:RETU
RN
1300 REM EDIT
1310 GOSUB 5000:PRINT"PRESS I TO INSER
T":PRINT:PRINT:PRINT"OR A TO ALTER/ADD
"
1320 PRINT:PRINT"OR D TO DELETE MORE T
HAN ONE LINE"
1325 PRINT:PRINT"OR R TO RENAME PILOT
PROGRAM"
1330 GOSUB 6000:IF G<>73 AND G<>68 AND
 G<>65 AND G<>82 THEN GOTO 1330
1335 IF G=68 THEN GOSUB 4400:GOTO 1390

1340 IF G=65 THEN GOSUB 4400:GOTO 1380
1345 IF G=73 THEN GOSUB 4400:GOTO 1360
1350 GOSUB 5000:PRINT"TO REPLACE "CP$;
" TYPE: ":LT=LL:LL=LF:GOSUB 6100:LL=LT
1355 CP$=A$:Z=10:RETURN
1360 PRINT:PRINT"NO. OF (BLANK) LINES
TO INSERT?"
1365 GOSUB 6100:NB=VAL(A$):IF NB<1 OR
NB>(MX-C) THEN PRINT"INVALID":GOTO 136
5
1370 FOR I=MX-NB TO C STEP-1:FOR I1 =
```

```
0 TO 1:P$(I1,I+NB)=P$(I1,I):NEXT I1:NE
XT I
1375 FOR I = C TO C+NB-1:P$(0,I)="":P$
(1,I)="":NEXT
1380 GOSUB 5000:IF C THEN PRINT C-1;"
";P$(0,C-1);D0$;P$(1,C-1)
1385 PRINT C;P$(0,C);D0$;P$(1,C):Z=1:A
$="EDIT":RETURN
1390 GOSUB 5000:PRINT"DELETE FROM ";C;
" TO ? ";:GOSUB 6100
1392 ND=VAL(A$):IF ND<C+1 OR ND>MX THE
N GOTO 1390
1394 FOR I= C TO ND:P$(0,I)="":P$(1,I)
="":NEXT
1396 FOR I=ND+1 TO MX:P$(0,I-ND-1+C)=P
$(0,I):P$(1,I-ND-1+C)=P$(1,I):NEXT:GOT
O 1380
1400 REM RUN
1405 IF CP$="" THEN Z=10:RETURN
1410 FOR I= 1 TO R:FOR P=0 TO 1:I$(P,I
)="":NEXT:NEXT
1420 F=0:GOSUB 5000
1430 PRINT:PRINT"RUN OF ";CP$;GOSUB 70
00
1440 FOR A=0 TO MX:IF LEN(P$(0,A))=2 A
ND RIGHT$(P$(0,A),1)<>M$ THEN GOTO 147
0
1450 FOR J= 1 TO 7:IF LEFT$(P$(0,A),1)
=S$(J) THEN ON J GOSUB 3000,3100,3300,
3400,3500,3600,3700
1460 NEXT:IF Q THEN A=MX
1470 NEXT A:Q=0
1480 PRINT:PRINT:PRINT"END OF RUN":PRI
NT:IF SC THEN PRINT"SCORE = ";SC
1490 SC=0:GOSUB 4700:Z=10:RETURN
1500 REM REPLACE STORED PROGRAM
1510 Z=6:GOSUB 6700:Z=10:RETURN
1600 REM SAVE TO CASSETTE
1610 CLS
1650 GOSUB 6800:Z=10:RETURN
1700 REM LOAD FROM CASSETTE
1710 CLS
1760 GOSUB 6900:Z=10:RETURN
1800 GOSUB 5000:PRINT"BYE-BYE":END
1900 REM INSTRUCTIONS
1910 GOSUB 5000:PRINT"T: TYPE WHATEVER
 IS ON THIS LINE"
1920 V=2:H=4:GOSUB 5200:PRINT"(MAX=";L
L;"CHARACTERS/LINE)"
1930 H=1:V=4:GOSUB 5200:PRINT"A: WAIT
FOR USER INPUT"
1940 V=6:GOSUB 5200:PRINT"M: MATCH CHA
RACTERS WITH INPUT"
1950 H=4:V=7: GOSUB 5200:PRINT"FROM A:
 (,BETWEEN ITEMS AND"
1955 V=8:GOSUB 5200:PRINT"END LINE WIT
H ')"
1960 H=1:V=10:GOSUB 5200:PRINT"J: JUMP
 TO LABEL (STARTING *)"
1970 V=12:GOSUB 5200:PRINT"C: CLEAR TH
E SCREEN"
1975 V=13:GOSUB 5200:PRINT"R: REMARK O
R COMMENT"
1980 V=14:GOSUB 5200:PRINT"S: + OR - N
UMBER TO ADD TO SCORE"
1990 GOSUB 4700:Z=10
2000 GOSUB 5000:PRINT"Y OR N ARE MODIF
IERS AFTER MAIN"
2010 V=2:GOSUB 5200:PRINT"COMMANDS (AN
D BEFORE:) TO GIVE"
2020 V=3:GOSUB 5200:PRINT"ACTION IF LA
ST MATCH WAS"
2030 V=4:GOSUB 5200:PRINT"CORRECT(Y) O
R INCORRECT(N)"
2040 V=6:GOSUB 5200:PRINT"* IS USED BE
FORE EACH LABEL"
2050 V=8:GOSUB 5200:PRINT"$ MUST PRECE
DE A STRING LABEL IN"
2060 V=9:GOSUB 5200:PRINT"A: LINES IF
IT IS TO BE USED"
2070 V=10:GOSUB 5200:PRINT"LATER IN (S
AY) T: LINES"
2080 V=12:GOSUB 5200:PRINT"E: OR END:
STOPS THE PILOT RUN"
2090 V=14:GOSUB 5200:PRINT"DONE IS USE
D TO END INPUT MODE"
2100 GOSUB 4700
2110 GOSUB 5000:PRINT"SAMPLE PROGRAM":
PRINT
2120 PRINT"  0?R:NAME INPUT":PRINT"  1
?*BEGIN":PRINT"  2?T:TYPE YOUR NAME"
2130 PRINT"  3?A: $NAME":PRINT"  4?M:S
UPERMAN,SUPERWOMAN,'"
2140 PRINT"  5?TY:DON'T BE FACETIOUS":
PRINT"  6?JY:*BEGIN"
2150 PRINT"  7?T:HELLO, $NAME":PRINT"
 8?END:*
2160 PRINT:PRINT"ALWAYS PUT A SPACE BE
FORE $":GOSUB 4700:RETURN
3000 REM ASK
3010 PRINT"    ?";:P$=P$(1,A):L=LEN(P$
)
3015 K=1
3020 IF MID$(P$,K,1)="$" THEN I$(0,F)=
MID$(P$,K):GOSUB 6100:I$(1,F)=A$:I$=A$
:F=F+1:K=L:K=K+1:IF K(=L THEN 3020 ELS
E RETURN
3030 K=K+1:IF K(=L THEN 3020 ELSE GOSU
B 6100:I$=A$:PRINT:RETURN
3100 REM TYPE
3120 P$=P$(1,A):L=LEN(P$):K=1
3125 IF MID$(P$,K,1)="$" THEN 3150
3130 K=K+1:IF K(=L THEN 3125
3135 GOSUB 3220:RETURN
3150 U=K
3155 IF MID$(P$,U,1)=" " OR MID$(P$,U,
1)="" THEN Y=U:GOTO 3180
3170 U=U+1:IF U<=(L+1) THEN 3155
3175 GOSUB 3220:RETURN
3180 U=Y:V$=MID$(P$,K,U-K)
3190 H=F-1
3200 IF V$(>I$(0,H) THEN H=H-1:IF H)=0
 THEN 3200 ELSE GOTO 3220
3210 PRINT MID$(P$,1,K-1);I$(1,H);MID$
(P$,U,L-U+1):RETURN
3220 PRINT P$:RETURN
3300 REM MATCH
3310 M$="N":N=1
3320 P$=P$(1,A):L=LEN(P$):FOR E= 1 TO
L
3330 IF MID$(P$,E,1)<>D1$ THEN 3370
3350 IF MID$(P$,N,E-N)=Z$ THEN M$="Y":
E=L:GOTO 3380
3360 N=E+1
3370 IF MID$(P$,E,2)=" " OR MID$(P$,E,
1)="" THEN E=L
3380 NEXT E:RETURN
3400 REM JUMP
3410 Q=0:D=0
3420 IF P$(0,D)(>"" THEN 3440
3430 IF MID$(P$(1,A),1,LL)=MID$(P$(1,D
),1,LL) THEN A=D:D=MX:RETURN
3440 D=D+1:IF D(= MX THEN 3420 ELSE V=
V+1:GOSUB 5200
3450 PRINT:PRINT"JUMP TO UNFOUND LABEL
 FROM LINE *";A:Q=1:V=V+1:RETURN
3500 REM CLEAR SCREEN
3510 GOSUB 5000:RETURN
3600 REM END
3610 A=MX:RETURN
3700 REM SCORE
3710 TS=VAL(P$(1,A)):IF TS)998 THEN SC
=0:RETURN
3720 SC=SC+TS:RETURN
4300 GOSUB 5000:PRINT MX;" IS THE LAST
 POSSIBLE LINE.":PRINT
4310 PRINT"   YOU MAY CONTINUE":PRINT
"BY SAVING THE PILOT PROGRAM"
4320 PRINT"AFTER INPUT OF THIS LINE ";
MX:PRINT:PRINT"THEN AMEND LINE 130 OF
THE"
4330 PRINT"PILOT INTERPRETER TO INCREA
SE":PRINT"THE VALUE OF `MX' BEFORE REL
OAD"
4340 GOSUB 4700:GOSUB 5000:Z=10:RETURN
4400 GOSUB 5000:PRINT"STARTING AT LINE
 ? ";:GOSUB 6100:C=VAL(A$)
4410 IF (C(1 AND A$<>"0") OR C>MX THEN
 4400
4420 RETURN
4500 FOR A= 0 TO MX:P$(0,A)="":P$(1,A)
```

```
="":NEXT: RETURN
4600 REM PAGING LIST
4610 IF NC<NL THEN RETURN
4620 NC=0:GOSUB 4700:GOSUB 5000:V=1:H=
8:GOSUB 5200:PRINT CP$:RETURN
4700 REM PRESS SPACE BAR
4710 V=23:H=1:GOSUB 5200:PRINT"PRESS S
PACE BAR TO CONTINUE";
4720 GOSUB 6000:IF G<>32 THEN 4720
4730 G=0:RETURN
4800 REM SET TABS FOR LINE NUMBERS
4810 V=V+1:H=1:IF A<100 THEN H=2
4820 IF A<10 THEN H=3
4830 GOSUB 5200:RETURN
4900 FOR I = 1 TO LL+5:PRINT"*";:NEXT:
RETURN
5000 CLS:RETURN
5200 LOCATE H,V:RETURN
6000 A$="":WHILE A$="":a$=INKEY$:WEND:
G=ASC(A$):RETURN
6100 REM INPUT CHECK
6110 S$=""
6120 GOSUB 6000:X=ASC(A$)
6130 S=LEN(S$):IF X=127 AND S THEN S$=
LEFT$(S$,S-1):PRINT CHR$(8)CHR$(32)CHR
$(8);:GOTO 6120
6140 IF X=127 THEN GOTO 6120
6150 IF X=13 THEN PRINT:A$=S$:RETURN
6160 IF X>31 AND X<91 AND S<LL THEN S$
=S$+A$:PRINT A$;
6170 GOTO 6120
6500 REM PRINTER ON
6510 sra=8:RETURN
6600 REM PRINTER OFF
6610 sra=0:RETURN
6700 REM PLACE FILE
6710 GOSUB 5000:PRINT"DO YOU WANT TO D
ELETE THE EXISTING FILE":PRINT:PRINT"O
F "CP$" ?"
6720 PRINT:PRINT"PRESS Y OR N"
6730 GOSUB 6000:IF G<>89 AND G<>78 THE
N GOTO 6730
6740 IF G=78 THEN Z=10:RETURN
6800 REM CASSETTE SAVE
6810 FOR A=MX TO 0 STEP-1:IF P$(0,A)="
" AND P$(1,A)="" THEN NEXT
6820 OPENOUT(CP$)
6830 FOR I=0 TO A
6840 WRITE#9,P$(0,I)
6850 WRITE#9,P$(1,I)
6860 NEXT
6870 CLOSEOUT
6880 RETURN
6900 REM CASSETTE LOAD
6905 INPUT"PROGRAM NAME? ",CP$:PRINT"W
AIT":GOSUB 4500
6910 OPENIN(CP$)
6915 A=0
6920 WHILE NOT EOF
6930 INPUT#9,P$(0,A)
6940 INPUT#9,P$(1,A)
6950 A=A+1
6960 WEND
6970 CLOSEIN
6980 RETURN
7000 FOR I=1 TO 2000:NEXT:RETURN
8000 END
```

Give your fingers a rest . . .

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

See our special offer on Page 77.

Final snag from Andromeda

I HAVE just completed Interceptor Software's adventure Message from Andromeda. It is a very original and exciting adventure and will test your intelligence up to its limit. Even at the end, where you think nothing could go wrong, the slightest mistake will cost you the game.

However near the end of the adventure there is a snag. When you reach the computer, insert the key and turn it you will be asked to type the password. I will not tell you what it is though, because that would spoil the adventure.

I do suggest hunting around the limestone maze, for you will find some very interesting clues to help you. I have seen no use for the metal bar or the coin, but all the other objects hidden or not are all essential to carry out and complete the adventure. – **N.D. Hampton, Littlehampton, West Sussex.**

■ Andrew Spence reviewed this Adventure in our February issue and it looks as though you enjoyed it almost as much as he did.

TAG colour

AFTER reading about the problem of setting the graphics printing (TAG) colour in the first two issues of your excellent magazine, I sat down with the Concise Firmware Specification to solve it. The following program resulted.

While it may not be possible to directly alter the graphics pen number from Basic, the operating system provides a routine, GRA SET PEN, at $BBDE that sets the graphics pen to the value held in the accumulator. Hence, all that was needed was to set up the accumulator and call the routine.

This program will set up such a routine:

```
10 h=HIMEM:MEMORY h-6
20 FOR a=1 TO 6
30 READ a$:POKE h+a,VAL("&"
 + a$)
40 NEXT a
50 setpen = h+1
60 DATA DD,7E,00,C3,DE,BB
```

When the program is in memory, the graphics pen can be set using:

```
CALL setpen,(pen number)
```

If I have any criticism of Computing with the Amstrad *it is that it is full of beginner's articles. I have nothing against beginners, but could we please have more articles like Amstrad Expansion (Issue 1) or maybe a simple explanation of the layout of the display-file in memory and the operation of the 6845 CRTL?*

Other than this slight gripe, Computing with the Amstrad *is an excellent magazine and well worth the money. May it have every success in the future.* – **Philip Marson, Bristol.**

Characters on parade

OUR family have recently adopted an Amstrad computer and we have had quite a bit of fun getting acquainted. We have also taken out an order with our local newsagents for Computing with the Amstrad, *which we have found to be interesting, useful and entertaining.*

You might be interested in the small program below which displays all the computer characters, including 0 to 31, using the CHR$(1)

```
100 CLS:MODE 1:i=-1
110 ON ERROR GOTO 200
120 FOR j = 1 TO 20
130 FOR k = 1 TO 20
140 i = i+1
150 PRINT CHR$(1) + CHR$(i)
160 IF k<>20 THEN PRINT "
";
170 NEXT k
180 PRINT:PRINT
190 NEXT j
200 GOTO 200
```

spaced out over the screen in Mode 1. Line 210 prevents "Ready" messages scrolling the first line of characters above the top of the screen.

– John Warner, Malvern, Worcs.

Erase old tapes

THIS may be of interest to other readers with Save problems.

Although getting all the right screen messages, and hearing the little bytes being transferred, I have found that either the SAVE has failed or the computer is unable to sense the program in CAT or LOAD, about 70 per cent of the time.

Initially I suspected duff tapes, but they recorded audio or Spectrum programs without problem on testing.

The conclusion seems to be that if the tape has been previously used to record programs from other makes of computer, or audio, the signal present either corrupts Save, or unerased portions between new Amstrad recordings, confuses the Amstrad so that it does not detect the program.

The answer seems to be to always use new tapes or to erase old tapes by recording without input on an ordinary audio recorder, before using them for the Amstrad.

A bit of an inconvenience but it seems to work. – **Dorene Cox, Dagenham, Essex.**

■ It sounds as though you have a tape recorder with a defective erase head. The erase head accompanies the recording head on the tape whenever you press play and record and should remove any 'garbage' before the recording is made.

Ready to dig deeper

THE problem Alex McKay was having with changing print colours using the TAG command (Computing with the Amstrad, *February issue) can be resolved by:*

```
PLOT 0,0,N:MOVE X,Y:PRINT""
```

(where N is the pen required).

Although long winded it's readable and easy to follow. Perhaps a better way lies in the Firmware Specification book.

I'd like to see some powerful utilities written for the CPC464 such as sprite commands (in MODE 1) which would enable the fast and smooth movement of sprites using the RSX, plus a sprite designer.

It's not enough to just give us these, but I would like to see a detailed breakdown of just what's happening so that I can make modifications, if necessary, at a later date.

The article going along the right lines (albeit slowly) is Mike Bibby's Guide to Machine Code.

I'm sorry if I seem a little impatient but I've had my CPC464 some ten weeks now and I'm eager to 'dig deep' into its workings. – **Michael Drury, Barnsley, South Yorkshire.**

Through first

I HAVE solved Forest at World's End! Interceptor Micros are doing a really good job.

Your magazine is a miracle maker, it has got my mum interested in the Amstrad through First Steps! But I still ask myself, where is a page for adventures? – **Tom Tofield, Ashford, Middlesex.**

■ Perhaps we ought to start a Mums' Corner as well as an Adventure section!

Rude monitor

I RECENTLY bought a CPC464 and after two weeks of exploration was half way through loading a program from your magazine ready for my first attempt at saving it on a tape when the monitor spat at me.

There was a discharge from somewhere at the back of the monitor and everything froze. It would not enter, run, save nor accept any further information.

If this happens again how can I retrieve my one and a half hours work? After reluctantly powering down and resetting everything I discover the machine now seems to function OK.

It concerns me that the computer and monitor are imported from Korea with no quality control, and what seems to be one small component failure could harm what seems to be generally an excellent machine.

Could it be that Amstrad require the consumer to be his own quality control, thus cheapening Amstrad's operation?

I pen this letter in the hope that others with a like experience may contact you or Amstrad and help prevent the misery I felt at seeing one and a half hours work disappear down that big white plughole as you switch off. – **A.G. Reader, Dagenham, Essex.**

■ It sounds very much like an electrical mains spike robbed you of your program. They don't happen often fortunately, but when they do they are most annoying.

One answer is to save your work more often than every one and a half hours, particularly if you have carried out any major alterations. Alternatively you can try a special spike suppressor. It comes in the form of a plug – your dealer should have one.

There may be a fault on the mains socket at home. It's worth a check, but get an expert to do it.

FX80 problem

I RECENTLY purchased a copy of Computing with the Amstrad. *I have only had my machine for a day, but Mr Fisher's problem, mentioned in your Postbag page, is simply overcome.*

I have an Epson RX80 connected to my Amstrad and I do not get double line feeds as I have altered the Epson DIP switch.

There are two DIP switches on the RX80, and if pin 3 of the smaller switch (it has 4 pins) is on then it will give double line feeds.

What Mr Fisher needs to do is look in his Epson operation manual for the FX80 DIP switches, and make sure that the line feed pin (no 3 on the RX80) is OFF. There is no need to cut any wires on the interface at all.

It is always worth while reading the manual first! – **J.H. Treacy, London.**

■ Thanks for your comments. It seems the problem is easily solved on the RX80. However Mr Fisher was asking about the FX80, and our reply is correct as it stands – it's always worth reading the letter first!

The DIP switch you mention (actually number four on the FX80) does not solve the problem – it's to do with the way the interface is set up.

Of course it's worth a try with the DIP switches first, but you won't always get away with it!

Pretty bug-free

A LETTER from John Hayes in the February issue of Computing with the Amstrad *suggested that the word wrapping feature of the Basic, in which a string that is too long to fit on the current line is moved completely to start on the next line, is in some way a bug (the word was mentioned twice), rather than the feature of the Basic that is fully documented in the Concise Basic Specification.*

For example:

```
10 MODE 1
20 LOCATE 38,10:PRINT "Long
string"
RUN
Long string
```

The way in which we suggest that a user should override this feature is by use of PRINT USING "&" thus:

```
10 MODE 1
20 LOCATE 38,10:PRINT USING
"&";"Long string"
```

This will have the desired effect.

I hope you will be able to include a note to this effect in your next issue as we aren't ashamed to admit to valid bugs (well not too ashamed), but this particular, so-called, bug has been identified by a number of people, generating a popular mis-conception that CPC464 contains a large number of bugs, when, in fact, there are very few. – **Cliff Lawson, Software Engineer, Amsoft.**

■ All we can say is that one man's bug is another man's feature – fully documented or not. Certainly if anyone said to us: "Have you come across the Amstrad's long string bug?" we'd know what he was talking about.

Anyway, thanks for your help in showing readers how to get round this feature.

Saving data

AS a recent purchaser of a CPC464 and a newcomer to Locomotive Basic, I have a very simple problem which I am unable to solve by reference to any of your publications.

In programs where calculations are carried out it is often necessary to be able to save the result as data for future calculations, such as a running total of a day's

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

takings. All methods I have tried have resulted in the figures being wiped on saving to cassette.

Please what is the answer? Or where can I find it? – **C.B. Archer, Oxted, Surrey.**

■ The answer to your problem is to open and close a dummy output file at the start of your program, before you do any initialising. This allocates a buffer for the cassette system and is achieved by typing in the following three lines:

```
10 OPENOUT "dummy"
20 MEMORY HIMEM-1
30 CLOSEOUT
```

Going down

I HAVE discovered how to make the Amstrad's screen scroll downwards using Basic.

Printing CHR$(11) makes the cursor move up one line. When the cursor reaches the top of the screen, anything on the screen moves down. The following program shows this:

```
10 MODE 1
20 LOCATE 7,1
30 PRINT "COMPUTING WITH
THE AMSTRAD"
40 PRINT CHR$(11);
50 GOTO 40
```

I hope this will be of help to other readers. One last thing, how about an "adventure problem" page and congratulations on an excellent production. – **John McCrory, Belfast.**

Line feed

WITH reference to your reader's enquiry regarding the Epson link-up – between the Amstrad and the FX80 printer – I encountered the same problem.

It appears that this can be rectified through the programming as the Tasword Word processor makes specific allowance in its instruction book to make the necessary adjustment – and it works.

The nearest I have got to correcting this problem – outside Tasword – is to produce an artificial spacing between the lines that approximates to a single line feed. You can achieve this by keying in instructions for 7/72 inch line spacing ie:

```
PRINT #8, CHR$(27);"1";
```

However I am sure there is a better answer. – **John Bennett, Wembley.**

Shield yourself from those monitor jitters

I FIND your magazine a great help in leaning to use the CPC464, especially as it is my first computer.

However my main reason for writing is to help all GT64 users to get rid of the violent undulations, screen distortion or as Paul O'Neil described it in your March issue a hum-bar producing oscillations.

I spent the first six weeks of owning my CPC464 trying to solve this problem, which is more than annoying especially late into the night as seems to be the norm for us computer addicts.

The problem is that it doesn't always happen straight away on switch on and unfortunately is widespread throughout all GT64 screens which haven't been modified.

So even if you get a replacement the new one will almost certainly have the same fault.

Because of the above facts unfortunately the so called computer experts in our department store in Southampton refused to believe that the fault even existed.

So after many phone calls and visits to this store I became desperate and contacted Amstrad technical department myself.

They confirmed that this undulation was a problem with their GT64 monitors and it was easy to rectify. They told me to return it and they would have it back, modified within a week – the modification consists of an extra shield fitted to the transformer.

Well nearly four weeks later it returned and at long last my monitor is rock steady with no more violent undulations to strain the eyes.

If you can't do without it for up to four weeks then either obtain a loan monitor which I did eventually, or do as I saw in another letter to a different magazine, remove the transformer to the outside of the casing. – **P. Basher, Southampton.**

Monitor choice

I INTEND to buy a colour TV to double as a monitor for the CPC464 and as a second TV for the family.

While I appreciate I could just purchase any TV and the MP1 power supply/modulator and 'hey presto' colour, I would prefer to buy one of the TV/monitors with direct RGB input currently on the market, since this I believe would give a better quality picture.

I and I am sure many other CPC464 users would appreciate your unbiased advice as to the compatibility of these TV/monitors with our favourite computer. – **P.R. Quemby, Stilton, Peterborough.**

■ As we have no experience of the TV/monitors you mention, we are unable to express an opinion. However we are sure our readers will oblige.

Short on words

I HAVE just obtained the disc drive for the Amstrad CPC464. While the equipment is almost without fault, I must say that I am disappointed with the documentation.

The ribbon cable exits from the interface unit at the side and interferes with the video and power cables. If the cable had come out from bottom of the interface this problem would not have happened.

Additionally the manual tells you to make a copy of the 3in disc, but surprise, surprise, Amstrad have NOT supplied a second disc.

Considering the complexity of the facilities that you get with the disc drive and CP/M and Dr Logo, the documentation is very poor. It seems that you now have to buy £40 worth of extra documentation (CP/M User Guide and Logo User Guide).

The strange thing is that you get six programs (including an assembler and debugging aid) with absolutely NO documentation.

Perhaps you could run an article or two on these missing areas and warn other folks to order some discs with the disc drive. – **C.W. Hill, Chelmsford, Essex.**

■ We think your only real grounds for complaint would have been if it was stated that the package did contain a 3in disc and the manuals for Logo, etc. Even as you read this reply, our A team hacks will be working on the articles you want.

Lack of resolution?

ONE reason why I bought the Amstrad was for its advertised resolution of 640 x 200 pixels. On reading the manual I was

further impressed when the resolution was said to be 640 x 400 pixels.

However, I am now confused and somewhat disappointed because in Chapter 5 it has now gone down to just 320 x 200 pixels! Which, if any, is right?

By the way, where can I get hold of software other than Amsoft stuff? Try printing a 'Software Available' list as well – it would be a great help! – **Toby Pankhurst, Littleton, Winchester, Hants.**

■ If you check Page 3 of Chapter 5 you will see that the resolution changes depending which mode you are working in.

Software is available from houses such as CRL, Level 9, Addictive, Bourne, Kuma, Camel and Virgin Games, to mention but a few.

Hysterical DMPI

I HAVE recently bought an Amstrad CPC464 and find it vastly superior than any other home computer. Part of this praise must be allocated to Amsoft for their support and encouragement.

However Amstrad have a failing in the direction of printers. Having recently seen the DMP1 in action I have concluded it would be better to stand next to a pneumatic drill. It's output is hysterical and the manual supplied resembles earlier versions of the Beano.

As a school student studying computers for 'O' and 'A' levels I need a printer with speed, clarity of characters and the ability to draw graphs and perform full screen dumps.

I have my eye on the Epson RX80. Could you tell me whether it can fulfil the aforementioned requirements and does it need a special cable? – **S. Lola, Forest Gate, London E.7.**

■ We don't think you'll have any complaints with the Epson, but you will have to purchase a special lead to make it compatible with the Amstrad.

Al's beat

Faster equation may be answer

REGARDING Alan McLachlan's program in the March Computing with the Amstrad, *he doesn't say exactly which part of the program he wishes to speed up. If he would like the movement of the finished drawing to be faster, I suspect he would have to perform the ink switching using machine code. The actual drawing of the circles (ellipses), however, may be speeded up by about 50 per cent.*

Although the most common method for drawing a circle is to use the polar co-ordinate method

x=r.cos(a) y=r.sin(a)

the sine and cosine functions on home computers tend to be rather slow. A circle may also be defined in terms of x and y by the equation:

$$x^2+y^2=r^2$$

which can be re-arranged to give:

$$y=\text{sqr}(r^2-x^2)$$

as in the program below.

This method is faster as it doesn't involve the use of sine and cosine functions. It is, however, dependent on the radius of the circle. With a radius of about 350 it works out slower than the polar coordinate method. – **J.P. Roberts, Gateshead, Tyne and Wear.**

```
10 MODE 0
20 CALL &BC02
30 ORIGIN 320,200
40 r%=150
50 r2%=r%*r%
60 s=0.5
70 FOR x= -r% TO r%
80 c=ABS(x) MOD 15+1
90 y=SQR(r2%-(x*x))
100 PLOT x,y,c:PLOT x,-y,15-c
110 PLOT x*s,y,c:PLOT x*s,-y,15-c
120 PLOT x,y*s,c:PLOT x,-y*s,15-c
130 NEXT
140 FOR i%= 0 TO 15
150 INK i%,0
160 NEXT
170 WHILE 1=1
180 FOR i%=1 TO 15
190 INK i%,26
200 FOR x= 1 TO 100:NEXT
210 INK i%,0
220 NEXT
230 WEND
```

Roland's keys

HERE are some keys that are not included on the inlay of Roland on the Ropes:

Pause – Enter
Space – Restart
ESC – Abort game

– **J. Smith, Southoe, Cambs.**

Buffer flusher

LET me begin by congratulating you on an excellent, unbiased publication, and long may you remain on our bookstalls.

I have watched with interest your readers' problems with the keyboard buffer, and would like to offer the following solution.

To begin with, there is no single CALL which will flush the keyboard buffer, or if there is then Amstrad are treating it as a trade secret.

There is, however, a small routine which will allow the user to flush the buffer quickly.

I would ask those who own a copy of the Firmware Specification to turn to page 14.5, which documents the routine at address &BB09.

The routine is KM READ CHAR, which tests to see if a character is available from the buffer, and if a character is found acts upon it.

If no character is available then the routine returns without waiting for a key to be pressed.

As documented, the way to flush the keyboard buffer is to repeatedly call the routine until it returns without a character. The following short routine does just that.

Please note that as we are only flushing the buffer it is not necessary to do anything with any characters which may be in the buffer.

```
      ORG 30000
READ: CALL &BB09
      JR C,READ
      RET
```

The routine occupies just six bytes of memory, and may in fact be relocated anywhere in memory, although I have put it at 30000 purely for demonstration purposes, and because it is unlikely to get in the way of anything else.

On calling the routine at &BB09 any characters present in the buffer are placed in the accumulator and the carry flag is set. Therefore we carry on reading the buffer until carry is false.

This means that no character is pending – that is, the buffer is empty.

On returning from &BB09 the other flags will be corrupted, so if used from a machine code program it may be necessary to PUSH AF onto the stack and POP AF before returning to the calling program.

I trust that this routine will suffice for your reader.

For the benefit of those readers who are using Mike Bibby's program Hexer, the codes to be entered in response to the BYTE prompt are:

CD 09 B9 38 FB C9

The start address should be

30000, which is &7530.

To use the routine from within a basic program simply CALL 30000.

The routine has no disastrous side effects and leaves all pre-set function keys intact.

I trust that Mr Worth of Glasgow will find the routine does the job as he would like. – **Peter Paton, Burnley, Lancs.**

Tilted Trapper

I THINK there is a fault or bug somewhere in the Trapper program (Computing with the Amstrad, *January) page 26.*

I tried to get the monster down but the backslash key / does not seem to do anything. Z X and / up all work but not / down. The program was typed in ditto to that on page 26.

I then passed the cassette to another CPC464 user who tried it on his Amstrad and the same thing happened, / did not work. – **R. Rodrigues, Moreton, Wirral.**

■ Our typesetters normally bend over backwards to help us but this time they bent the wrong way I'm afraid. The symbol printed in the article for "down" should have been '\' not '/'. This one got past our proof reading as well – but very few do.

Leaving the club

I WOULD like to congratulate you on an excellent magazine but also have a moan about the Amstrad user club. When buying the Amstrad we are bombarded with literature from Amstrad urging us to join the users' club and gain benefits like cheap software and a free magazine, all for just £20.

Soon after joining I discovered that Amsoft titles can be bought cheaper elsewhere, even cheaper than Amsoft's discount prices.

This is appalling. Amsoft must sell the software to the distributors dirt cheap and so cut the throats of their owners' club.

Admittedly I get the club magazine but this can now be bought from any newsagents for 95p. Therefore I feel that I would have been better off by not joining the club and just ordering the magazine.

Before writing this letter to you I have given Amsoft the right of reply and have written to them four times in all, yet they do not seem to have the courtesy to reply. Also I have written asking for help but again have received no reply.

In future I shall just write to a magazine such as yours and I shall certainly not renew my membership with Amsoft.

However after all the moaning I feel I must say what a brilliant machine the CPC464 is. It's just a shame Amstrad don't offer the back up service that they boast. – **Philip Harman, Tunbridge Wells, Kent.**

Hint from Al

I TYPED in all the games from your first three issues, plus the word processor and screen dump, and couldn't get any of them to run without chucking out a variety of errors.

I suspected that they couldn't be all in your listings so I asked a mate of mine to help me. Yes they were all mine – mistakes and typing errors, and I'm very frustrated at not being able to spot them myself.

Now to the point. Please, please, if only for me (and a few other novices) will you do an article on how to check a program properly – you know, hints on how to make it easier to find errors.

My friend showed me how to put STOP in and a CHR$(7) beep to show where the program had got to.

By the way, the programs are all running now and they're great, even Digger with its bonus% error (got you on that one – well my mate did).

Keep it up, and hurry up with that article, we're all willing to learn from the experts. – **Cyril Bellis, Frodsham, Cheshire.**

■ By the time you read this you will have realised no doubt that Al's Beat has included this month a helpful hint of the kind you mean.

Forthcoming months should find a few more of these intended to make life a little easier.

Printing up the answer

I HAVE just written a machine code routine on my CPC464 that will perform 16 bit Integer multiplication. The routine itself works OK, but I am having problems printing the correct answer on the screen.

I am using the routine called TXT OUTPUT at BB5A to print the value contained in the HL register pair.

Although I know the value is correct I can find no way of representing this as a decimal instead of two separate hex numbers.

Is there a way to do this without having to use Basic?

Finally, can you recommend a book that details how to program the 6845 CRT display chip. – **A.P. Tompkins, Purley, Surrey.**

■ Here's a routine that should do the trick. It's designed to be used with Hexer (see Mike Bibby's machine code article in the March issue), though you can put it anywhere.

As shown, the number you want printing out is stored at &2FF8,9 and copied to "workspace" at &2FFA,B.

This version prints out all the leading zeros. For example, &200 becomes 00512.

You can get round this by printing spaces instead of zeros, but check for the special case when the number is itself zero, or you'll just print blanks!

```
 1                      ORG  3000H
 2                      LOAD 3000H
 3
 4
 5            Nuaber:   EQU  2FF8H
 6            Hold:     EQU  2FFAH
 7
 8
 9 3000 0E05  Start:    LD   C,5              ; elesents in table
10 3002 2AF82F          LD   HL,(Nuaber)
11 3005 22FA2F          LD   (Hold),HL
12 3008 213130          LD   HL, Table
13 3009 5E    GetPower: LD   E,(HL)
14 300C 23              INC  HL
15 300D 56              LD   D,(HL)
16 300E 23              INC  HL
17 300F E5              PUSH HL               ;next elesent on stack
18 3010 0600            LD   B,0              ;B counts subtractions
19 3012 2AFA2F          LD   HL,(Hold)
20 3015 E600  Subtract: AND  0
21 3017 ED52            SBC  HL,DE
22 3019 3003            JR   C,NoSub
23 301B 04              INC  B
24 301C 18F7            JR   Subtract
25 301E 19    NoSub:    ADD  HL,DE            ;restore reains
26 301F 22FA2F          LD   (Hold),HL
27 3022 78              LD   A,B
28 3023 C630            ADD  A,30H
29 3025 CD5ABB          CALL 0BB5AH           ;TextOut
30 3028 0D              DEC  C
31 3029 2804            JR   Z,Finish
32 302B E1              POP  HL               ;points to table
33 302C C3093D          JP   GetPower
34 302F E1    Finish:   POP  HL               ;tidy up stack
35 3030 C9              RET
36 3031 1027  Table:    DW   10000
37 3033 E803            DW   1000
38 3035 6400            DW   100
39 3037 0A00            DW   10
40 3039 0100            DW   1
41                      END
```

ALAN McLACHLAN prowls the whacky whimsical world of the Amstrad CPC464

YOU know folks, one of the most entertaining times of the day in the editorial office is the morning mail session. To the accompaniment of the sparrows coughing outside I do a quick scan of your letters, separating them into categories such as Postbag, submissions for publication, comments and suggestions, and listing queries.

This is done so that later, when everybody has come down off the ceiling (editor included) the A Team can tackle the letters together.

During the last few months we've received sackfuls of mail on all sorts of topics. Particularly enjoyable are those that praise the magazine content, those that offer constructive criticism and suggestions for improvement and in particular those that contain praise for the games written by amateurs not on our staff (as opposed to the amateurs on the staff).

Queries from people who have typed in listings incorrectly and so can't get the programs to run have introduced us to a host of wonderful characters whose existence we'd never suspected.

I'd like to introduce you to some of these by quoting some short sections from letters, and if on reading them you recognise yourself, please don't take offence as none is intended.

The first character is the **chess player** – *"I always check everything thoroughly, I've checked and double checked but . . ."*

A variant is the **railway employee:** *"I've checked every line and there isn't a single fault to be seen"*. He's probably been reading Mike's Z80 articles. We usually reply that he's lost track of the listing and he normally gets the point.

Our favourite is the **typist:** *"As I have years of experience using a typewriter the chance of any errors being mine are exstreamly unlikely"*.

That one's followed closely by the **postman:** *"I've checked every single letter and I can't find anything wrong"*.

More dubious is the **pot smoker:** *"I've passed it around among my friends and they all agree there's nothing wrong with it"*.

Rather more serious is the **doting parent:** *"I got my Johnny an Amstrad for Xmas and he's had a lot of fun with it until he typed in your Trapper. He has spend hours typing it in and it's not fair as he cannot get it to work. There must be a mistake in it as it keeps coming up 'Syntax Error' all over the place"*.

He's the one who says everyone's out of step but our Johnny.

A similar case is the **ice skater:** *"I've been backwards and forwards over it till I'm blue in the face, and I'm still no further"*.

And, of course, the inevitable **accuser:** *"Having tried all weekend to get your program to run, I'm sure you put deliberate errors in to make people buy your monthly tapes"*.

A more dreaded character is the **improver:** *"Your program does not work properly. The following additions will stop the Grumpies going outside the maze"*. (They never did on our original version, but you try convincing him).

As a Spielberg freak the editor is fond of the **film fan:** *"We've searched high and low for Gremlins, but can't find any. They seem to be hidden away in the listing and we're just not good enough to find them"*.

Of course there's the **pacifist:** *"We are not normally the type of family to complain but all we have is 'Arguments' all over the place"*.

We all like the **Rentokill man:** *"I've got rid of all my bugs but I think you must still have some, and I can't find them"*, and the **motor maniac:** *"I've run over it time and time again and I'm sure I'm right"*.

These are closely rivalled by the **search party:** *"We've checked every entry and there's nothing unusual to be seen"*, and the more evil **masochist:** *"I've made a painstaking check of my listing and can only assume your's to be in error"*.

There's the **out of work GP:** *"I must admit I have run out of patience"* (think about it) and the **racing driver:** *"I've been through it time and time again and it's driving me round the bend"*.

Saddest is the **optimist:** *"I have been told by your telephone receptionist to write in as I am desperate. I have typed in four of your games and can't get any of them to work.*

"I have saved them all on cassette and ask you to look at them for me and find out where I've gone wrong. I am an absolute beginner and don't know where to start looking for errors".

At least this last lady admitted that the faults were probably hers. Unfortunately we just don't have the time to provide a debugging service, and certainly not over the phone.

Out of curiosity I took this particular cassette home one weekend and found in the four programs no less than 47 typing errors and four missing lines.

It took me 10 hours looking carefully through the listing and checking every entry against that in the magazine. Apart from four errors which I couldn't find without some useful debugging techniques, all the mistakes were plainly visible.

OK, I agree that to an experienced eye they will be more obvious than to an inexperienced one, but they were there to be spotted with careful scrutiny.

I'm not going to keep the debugging techniques to myself and intend, with the assistance of the rest of the A Team, to continue with a series of articles on hints and tips to

find your own programming errors.

Here is an example to be going on with. Let us assume you have typed in line 30 of Reaction Timer (page 62, February 1985 issue) as:

```
30 MODE 1:INK 0,1:INK 1,24:INK
2,20:INK 3,6:PEN 1:DIM a$(8):GOSUB100
```

Your micro will respond "Syntax error in 30" and the mistake should be fairly obvious – GOSUB 100 needs a space.

But what if there were more statements in the line and it were not so obvious? A simple hint is to split the line in two:

```
30 MODE 1:INK 0,1:INK 1,24:INK 2,20
31 INK 3,6:PEN 1:DIM a$(8):GOSUB100
```

Now you will get "Syntax error in 31". This can now be split:

```
32 INK 3,6:PEN 1
33 DIM a$(8):GOSUB100
```

and the result will be "Syntax error in 33".

Gradual elimination will narrow things down until your are left with the offending statement on its own. Your micro can't tell you what is wrong with it but with careful checking and perhaps experiment you should come up with the solution.

This is a fairly easy technique which can be used for many error messages. We'll look at some more another time.

By the way, if all your debugging sessions have failed and you have been the postman, the pot smoker, the ice skater and any other of my friends, then drop us a line. But a few words of advice:

- Don't expect an immediate reply. We're as fast as we can be, but we are snowed under.
- Do include a self-addressed envelope complete with stamp.

We can't promise to solve all your problems, but we try our best or at least admit defeat. The request for an SAE also applies to any other type of mail if you want a personal reply.

Well that's all for this month. I'm going to get some practice now at finding some of your errors, which are amazingly like my own. I'll put some of the techniques into practice in another session.

Ready Reference: String handling

Get the facts at your fingertips with the fourth of our reference charts

UPPER$ forces all the letters in a string into capitals.

```
letter$="abcdef"
PRINT UPPER$(letter$)
```

gives ABCDEF. Unsurprisingly, **LOWER$** does the reverse:

```
PRINT LOWER$("WXYZ")
```

returns wxyz.

LEN is a Basic function returning the length of the string enclosed in the brackets following it.

```
x=LEN("Spot")
PRINT x
```

gives 4, the length of the string Spot.

You can have a string variable inside the brackets:

```
moggy$="Bodger"
x=LEN(moggy$)
PRINT x
```

returns 6, the length of *moggy$*.

INSTR is used to give the position of one string inside another or 0 if it doesn't occur.

```
position=INSTR("tomato","mat")
PRINT position
```

gives 3, which is the start position of the substring "mat" in "tomato". The string to be searched comes first, the string that is sought comes after it.

```
PRINT INSTR("tomato","to")
```

gives 1 as the starting position of "to" within "tomato". This is only the first occurrence of "to". INSTR has another parameter which allows the search to start anywhere within "tomato".

```
PRINT INSTR(3,"tomato","to")
```

returns 5. The first number inside the brackets is the position that the search is to start from. Since this is 3, the search starts at the "m" of "tomato" and it is the position of the second "to" that is returned.

The start parameter has to be well chosen:

```
searched$="tomato"
sought$="mat"
? INSTR(5,searched$,sought$)
```

gives 0 as does

```
PRINT INSTR(9,searched$,sought$)
```

and:

```
PRINT INSTR("abcdef","xyz")
```

STR$ can be looked on as the opposite of VAL. It turns a number into a string.

```
number=10
PRINT STR$(number)
```

gives 10. Nothing much appears to have happened but what is returned is now a string. Try:

```
y=STR$(number)
```

and you'll see it is no longer a numeric. Using STR$ allows numbers to be used in string operations:

```
number=10
street$=" Downing Street"
number$=STR$(number)
address$=number$+street$
PRINT address$
```

VAL is used to give the numeric value of a string such as:

```
easya$="123"
PRINT VAL(easya$)
```

which returns the number 123. This number can be held in a numeric variable and have mathematical operations performed on it.

```
easya$="123"
x=VAL(easya$)
PRINT 2*x
```

gives 246 as the answer. Try:

```
PRINT 2*easya$
```

and see what you get. Strings have to be converted by VAL before you can do mathematics with them.

VAL only takes the numeric parts of the string, starting from the leftmost character. The rest is ignored as in:

```
easya$="123four"
PRINT VAL(easya$)
```

which gives 123, ignoring four. Also beware:

```
VAL("8+9")
```

which is 8, the same rule applying.

Now you've started computing with the Amstrad you'll want to read EVERY issue!

Give your fingers a rest

All the program listings from each month's issue are available on cassette for only **£3.75**

JANUARY: ***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.*

FEBRUARY: ***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 what they used to be. Test them out on this simple little program.* ***Origins:*** *Demonstrates the dramatic effects obtained by changing the graphics origin.*

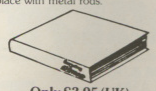
MARCH: ***Starfleet:*** *Earth's defences have been seriously weakened by a shortage of qualified starfleet pilots. Can you come to the rescue?* ***Swamp:*** *Hilarious arcade action as you join the Wotawallies in their desperate battle against the giant frogs.* ***Parachute:*** *Use simple addition to guide your parachute to earth in this educational program for the younger child.* ***Dump:*** *An invaluable screen dump utility for Modes 0 and 1.* ***Hexer:*** *A hexadecimal loader, ideal for the novice machine code programmer.* ***PLUS*** *an extra full-length game,* ***Four in a Row,*** *an enthralling 3D version of the popular game of luck, skill and strategy.*

APRIL: ***Mad Adder:*** *Appease our aggravated adder's appetite. Fun for all the family in this classic game.* ***Egg Blitz:*** *Eggs, boy scouts and a stunt pilot make up the fun in our special game for Eastertide.* ***Simple Sprites:*** *Create sprite-like effects to bring superb graphics action to your programs.* ***Caesar Cipher:*** *Learn how to crack the code with two listings that explore the earliest of ciphers.* ***Pilot:*** *An easy to learn language, ideal for computer aided learning.* ***Spelling:*** *Learning how to spell becomes fun with this addictive educational game.* ***Aleatoire:*** *Find out the odd ways mariners make decisions in our resident puzzler's program.* ***Palindrome Tester:*** *Slick string handling techniques for those who are backwards with words!*

New! We are pleased to be able to offer you annual subscription for the monthly listings cassette for only £40.

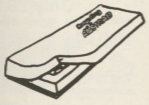
How to keep your collection complete

Bound in rich burgandy 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.



Only £3.95 (UK)

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

FREE! **Send in your subscription on the form below and we'll send you this month's cassette, worth £3.75, entirely free of charge!**

ORDER FORM

All prices include postage, packing and VAT and are valid to May 31.

Please enter your requirements by ticking boxes £ p

Annual subscription
UK £12
EIRE £13 (IR £16)
Overseas (Surface) £20
Overseas (Airmail) £40
Please send me my free April issue listings cassette
Commence with _____ issue
TOTAL _____

Back issues
£1.25 UK
£1.50 Overseas (Surface)
January
February
March
TOTAL _____

Tapes
£3.75 (UK & Overseas)
January (Smiley)
February (Digger)
March (Starfleet)
April (Mad Adder)
TOTAL _____

Cassette tape annual subscription
£40 (UK & Overseas)
Commence with _____ tape (state month)
TOTAL _____

Dust Cover
£3.95 (UK & Overseas)
TOTAL _____

Binder
£3.95 UK
£5.00 Overseas
TOTAL _____

Payment: please indicate method (✓) **TOTAL** _____

☐ Access/Mastercharge/Eurocard
Card No. _____

☐ Barclaycard/Visa
Card No. _____

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____ Signed _____

Address _____

Send to:
***Computing with the Amstrad*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

YOU CAN ALSO ORDER BY PHONE:

Orders (24 hours) 061-480 0173
All other enquiries 061-480 0171

Don't forget to quote your credit card number and full address.

Amstrad CPC464

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.

Stereo Output

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. Any sound that previously came out of the mono speaker will now be sent out via the interface in stereo. All programs that use the sound in anyway (i.e. commercial software) will now output through the interface, which is fitted with volume and balance controls.

Speech Synthesis

The Amstrad speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an almost unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chip's internal rom.

Text to Speech

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 than 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.

The machine code software is mainly developed to this mode of operation. 3.5K is used for tables which contain the rules & exceptions to the rules of the English Language.

e.g. I before E except after C) This therefore allows the user to enter words to be spoken in normal English.

Speakers

Supplied with the Speech Synthesizer are two high quality 4" speakers these have been designed to compliment the Amstrad Computer. They are fitted with 1 metre of cable and can be positioned for the best stereo effect. The synthesizer interface fits neatly on to the rear of the computer. It has a through connector to enable other interfaces (e.g. Disc Drive) to connect to the rear of the synthesizer for ease of expansion Please send S.A.E. for a copy of the instruction manual which will give full and comprehensive details.

New Basic Commands

There are 8 new Basic Commands which control all the functions of the interface. Making the Synthesizer very easy to use. You can even control the speed at which it will talk to you. Or use the synthesizer to create sound effects like a fourth sound channel.

10 PRINT " 'AMSTRAD' "

The above is an example of the Syntax for entering speech into the computer and shows how simple it is to use.

The instruction book gives comprehensive details and examples of how to use the interface both from machine code and basic.

How to Order

The Amstrad Speech Synthesizer costs only £39.95. You can obtain your synthesizer through any good computer store or by completing the order form and returning it to:
dk'tronics Limited, Shire Hill, Saffron Walden, Essex. OR by telephone quoting your barclaycard or access number. Orders normally despatched within 24 hours.

Please rush me

.....[QTY] *Amstrad Speech Synthesizer at £39.95 +£1.25 p&p*

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

or debit my Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signature.....

Name.....

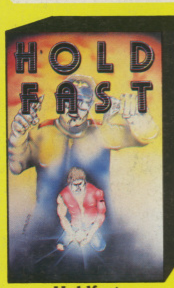
Address.....

dk'tronics

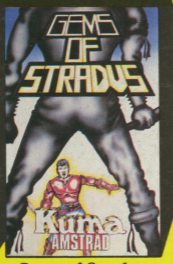
Saffron Walden, Essex CB11 3AQ
Tel: (0799) 26350 10 lines

the only choice

Kuma AMSTRAD CPC464 software



Holdfast



Gems of Stradus

Star Avengers

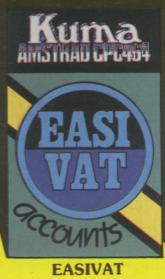
Galaxia

Music Composer

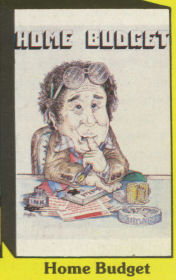
Logo

Database

ZEN Assembler



EASIVAT



Home Budget

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC 464 Micro-computer.

BOOK ● ***The Amstrad CPC 464 Explored.***

This superb book is designed to let every CPC 464 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.

Now available from selected branches of Co-op, Granada, LASKYS and John Menzies

Kuma Computers Ltd., Unit 12, Horseshoe Park, Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name .....

Address .....

.....

..... Phone.....

I own an Amstrad CPC 464 computer ☐

Trade Enquiries Phone 07357-4335

Visitors wishing to call at our Pangbourne Manufacturing and Distribution Centre are advised to phone 07357-4335 first for an early appointment.