A Database Publication

Computing with the AMSTRAD

No. 7
July 1985
£1

The independent magazine for CPC464/664 users

Link your Amstrad to the world!

Great offer on Page 9

Save civilisation in our gripping

MISSILE COMMAND

Full listing of this action-packed game

Check your biorhythms with our colourful calendar

Complete listing for Amstrad Z80 assembler

How to slash screen loading time

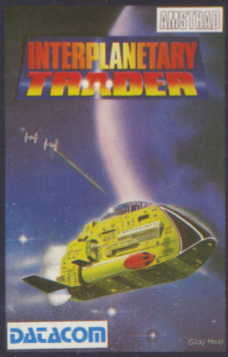
Amstrad teach-in

Design your own character set ● Basic bugs and how to avoid them ● Mysteries of indirect register addressing unveiled ● All about FOR . . . NEXT loops

WELCOME TO THE WORLD OF DATACOM

EXECUTION - They told me it would be bad but I never thought it would be like this...must keep a clear mind... can't afford to panic...time is fast running out...don't think my nerves will stand much more of this!!! A brain straining memory bashing game of words.

SNAIL PACE - Unbearable excitement. 30 'thoroughbred racing snails' battle to the finish. Super features include computer calculated odds, form guide, excellent sound effects, smooth m/c graphics. Rivetting excitement guaranteed.



INTERPLANETARY TRADER - The most addictive, mind blowing, feature packed space adventure you will ever experience on your AMSTRAD CPC 464. Defend your cargo against space pirates...navigate asteroids, black holes, magnetic storms...Total concentration is required in your quest to become a GALACTIC MEGABILLIONAIRE!!!

DATACOM PUBLICATIONS:
407F Hockley Centre,
Birmingham B18 6NF,
Tel: 021-233 1800

Trade enquiries welcome

All games just

£5.95

Each

These games will be available from your dealer or **POST FREE*** direct from: DATACOM PUBLICATIONS using the coupon provided (or just write in)

* U.K. only - Overseas add £1 p & p

Name.....

Address.....

.....

Please rush me ☐ Tapes of Interplanetary Trader ☐ Tapes of Execution ☐ Tapes of Snail Pace

Tel:.....

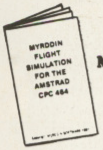
I enclose a cheque/P.O. for £.....

3D LANDMARKS YOU CAN FLY AROUND

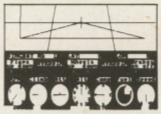
SUPERB REAL TIME SIMULATION

MYRDDIN FLIGHT SIMULATION

AMSTRAD CPC 464

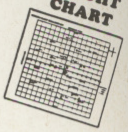


MANUAL

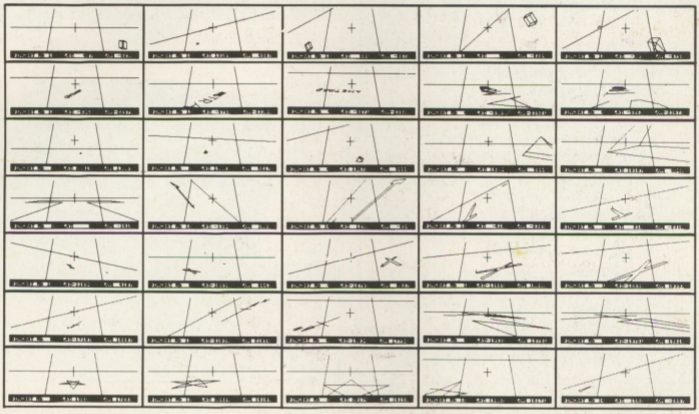


FULL SCREEN DISPLAY

FLIGHT CHART



Here are some screens from a typical flight showing the view from the cockpit (top half of screen) produced as printouts of the actual simulator.



A real time simulation with 3D graphics uses a massive 64000 x 64000 longitude & latitude flying area, making each flight completely different. Developed under pilot instruction to give realistic flight effect. The view through the cockpit gives moving 3D graphics.

Comprehensive instrument panel with moving needle meters & digital displays. 15 aircraft types with varying control sensitivities & speeds of between 100 – 500 knots.

3 runways available for refuelling, take off & landing. Ground and landmark orientation correct with all flying attitudes (rolls etc.).

The 3D graphics are still accurate when you fly upside down.

3D landmarks you can fly around.

Comes complete with manual & fully detailed chart of landmarks & airfields.

Joystick or keyboard operation.
Cassette version £11.95
Disc version £15.95
If your local dealer doesn't have it in stock yet, order from us direct.
For despatch within 48 hrs.
(usually 24 hrs.).

MYRDDIN SOFTWARE, PO BOX 61, SWINDON, WILTS.
Telephone: (0793) 40661

Please send me ..... Flight Simulator(s) by return of post for the Amstrad CPC 464

Name .....

Address .....

..... Postcode .....

Cheque enclosed for cassette £11.95/£15.95 disc (inc. P.P.)
OR Debit my Access A/C No:-

OR Telephone through your Access Order.

Signed.....

CWA 7

Vol. 1 No. 7 July 1985

Managing Editor: **Derek Meakin**
Features Editor: **Pete 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: 256871 MONREF G
Quoting Ref. 79:MAG001
Prestel Mailbox: 614568383

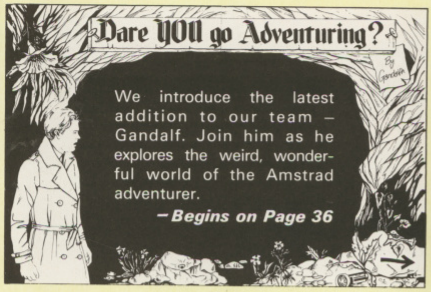
Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

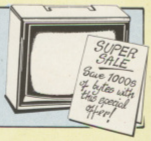
Subscription rates for
12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

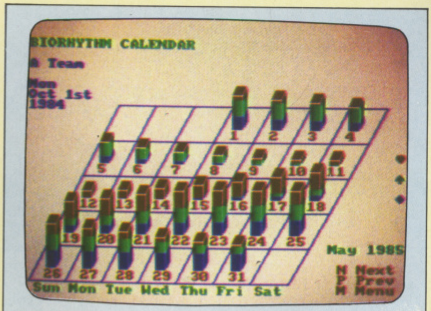
ABC Member of Audit Bureau of Circulations

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

Contents







Now YOU can fly with the legendary Red Arrows – in the most challenging flight simulation ever!

It's the most exciting flight simulator ever written for a home computer – the product of many months of dedicated work by some of Britain's top programmers, enthusiastically aided by the talents of aircraft designers, engineers, mathematicians – and the Red Arrow pilots themselves.

Every ounce of power contained in the micro, and its enhanced sound and graphics capabilities, is used to give the utmost realism to re-creating the most spectacular aeronautical displays ever seen in the skies of Britain.

You start by practising take offs and landings. Then, once you have won your wings, you fly in formation as part of the Red Arrows team. There's no margin for error as you fly a mere six to 10 feet from each other – at speeds of between 300 and 350 miles an hour!

But the real drama begins as you plunge into the death-defying manoeuvres that have been thrilling crowds at air shows for the last 21 years.

On the panel in front of you are all the instruments you need – plus a screen giving you an external view of the complete formation you are flying. Slip out of line for a second and the eagle-eyed Red Leader will be on the radio ordering you back into position.

The program comes with a detailed flight handbook that will soon give you the confidence to take YOUR place alongside the ace pilots of the Red Arrows, even if you've never flown before!

Be a VIP visitor with the Red Arrows!

Everyone who buys a Red Arrows computer program will be invited to enter an exciting competition. The winners will be given a VIP visit to the Red Arrows base at RAF Scampton, the wartime home of the Dambusters. Your visit will include two nights' accommodation at a luxury hotel. And while you are at Scampton you will be invited to sit at the controls of a Hawk.

Now on sale at:

BOOTS **COMET** **Currys** **Dixons**
Greens John Menzies **RUMBELOWS** **spectrum**
WHSMITH and other leading computer stores

ORDER FORM

	Tape (£8.95)	5¼" Disc (£11.95)	3" Disc (£12.95)	3½" Disc (£12.95)
Amstrad	☐	N/A	☐	N/A
Atari	☐	☐	N/A	N/A
BBC B	☐	☐	☐	N/A
Comm. 64	☐	☐	N/A	N/A
Electron	☐	N/A	N/A	☐
Spectrum	☐	N/A	N/A	☐

I wish to pay by:

☐ Access/Mastercard/Eurocard No.

☐ Barclaycard/Visa No.

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____

Address _____

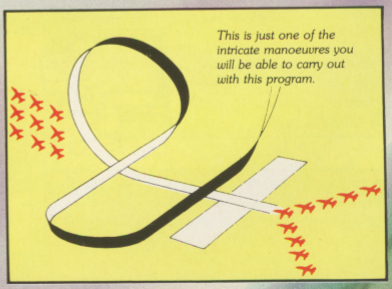
_____

Signed _____

Send to: Database Software, 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: **061-480 0173** *(24 hours)* *Don't forget to quote your credit card number and full address.*

CA7



This is just one of the intricate manoeuvres you will be able to carry out with this program.

Put yourself in the pilot's seat of the most manoeuvrable fighter in the RAF!


RED ARROWS
A gripping, realistic computer simulation for the
Commodore
Spectrum
Amstrad
Electron
BBC Micro
Atari

REALTIME SOFTWARE

3D STARSTRIKE

Actual Screen Shots (Amstrad)

AMSTRAD CPC 464 £6.95

ZX SPECTRUM 48K £5.95

- ★ STEREO SOUND
- ★ FAST 3D GRAPHICS
- ★ GREAT VALUE FOR MONEY

PLEASE RUSH ME!

☐ Starstrike (Amstrad) £6.95

☐ Starstrike (Spectrum) £5.95

Name .....

Address .....

.....

CWA 7

Cheques/P.O.s to:

Realtime Softwear, Prospect House, 32 Sovereign Street, Leeds LS1 4BT

AMSTRAD STARSTRIKE

In the shops NOW!

Exclusive offer from Computing AMSTRAD

Link YOUR Amstrad to the wide world of communications!

HERE's a great offer from *Computing with the Amstrad* that will enable you to link your micro into your telephone – and join the exciting world of tele-computing!

What we have put together is a sophisticated combination of hardware and software that provides you with all the vital input/output facilities you need to become one of the ever-growing number of communications enthusiasts.

The software is the most versatile communications package yet developed for the Amstrad. It includes a powerful terminal emulator that will allow you to access thousands of other computers, spanning micros to mainframes.

You will be able to contact bulletin boards, throughout Britain and overseas. And via the newly-launched MicroLink, you will be able to join Britain's biggest electronic mail service, Telecom Gold, where you will be able to send messages in a fraction of a second – for less than the cost of a stamp! You will even be able to turn your Amstrad into a telex machine.

As if all that wasn't enough, we've also incorporated a superb four-colour Prestel terminal, opening up to you more than 300,000 pages of information, electronic magazines and clubs.

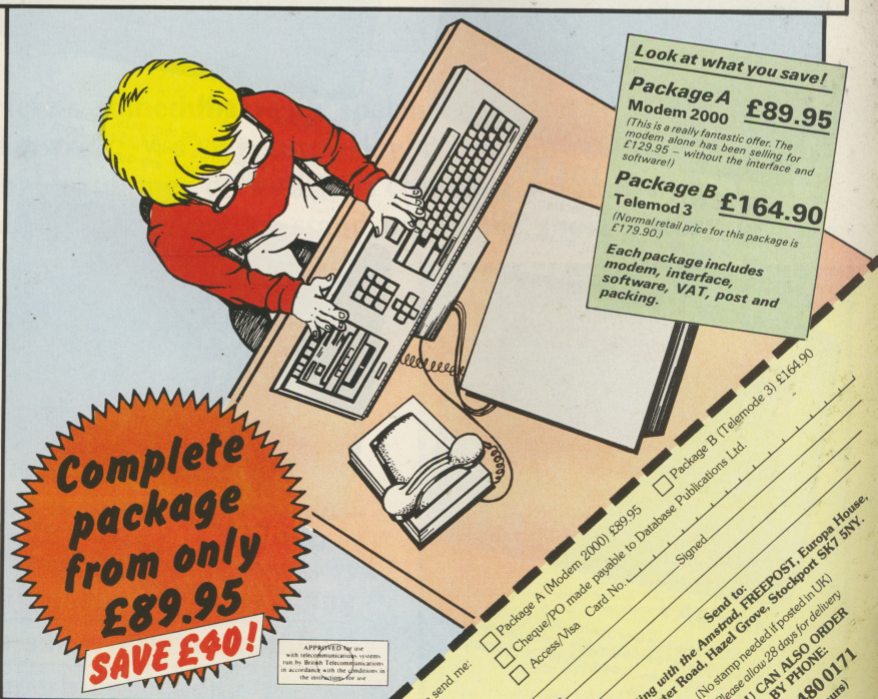
The whole package has been designed to be so simple that even the most inexperienced beginner can use it. Yet its ease of use disguises a communications potential powerful and flexible enough to satisfy even the most demanding hacker.

And the hardware matches the software in its adaptability, featuring established modems that have been tried and tested on other micros.

You have two choices. There is the Modem 2000, offering 1200/75 full duplex, 1200/1200 half duplex baud rates. Or the Telemod 3, which in addition has an extra 300/300 full duplex baud rate used by many bulletin boards – plus a very useful auto-answer facility. Either way, you'll be getting a high quality modem that is fully BT-approved.

And here's a most vital point. Both modems plug directly into BT's standard 600 series jackplug, which means your communications will be free from the frustrating corruption so often associated with the earlier acoustic modems.

Yes, now's the time your Amstrad joined the communications revolution – and this offer is the best way to do it. It's a revolutionary offer – at a really revolutionary price.

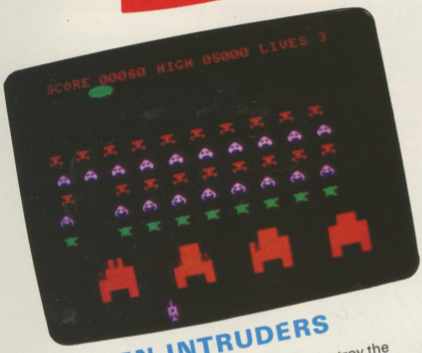


Please send me: ☐ Package A (Modem 2000) £89.95 ☐ Package B (Telemode 3) £164.90
☐ Cheque/PO made payable to Database Publications Ltd.
☐ Access/Visa Card No. _____ Signed _____
Name _____
Address _____

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

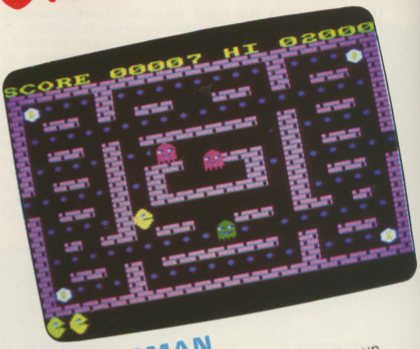
YOU CAN ALSO ORDER BY PHONE: 061-480 0171 (24 hours)

4 GREAT GAMES FOR THE PRICE OF ONE!



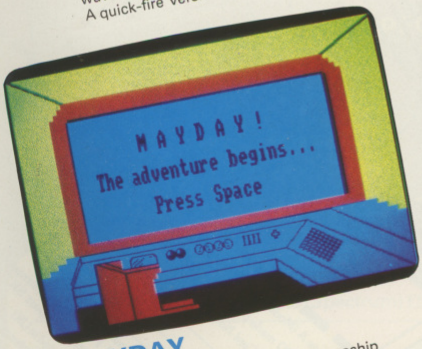
ALIEN INTRUDERS

With only your laser for protection, destroy the waves of aliens who threaten to engulf you. A quick-fire version of an all-time great!



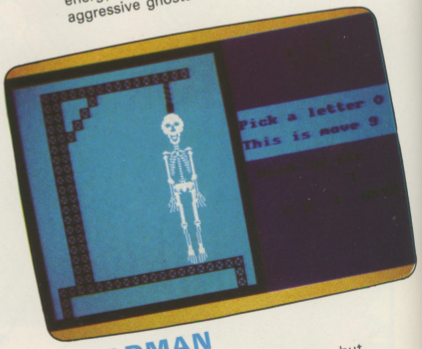
SNAPMAN

Steer your man around the maze, gobbling up energy pellets while keeping away from the aggressive ghosts who are out to get you.



MAYDAY

Guide the sole survivor of a stricken spaceship through the wreckage of his craft. Recover vital medical supplies . . . or a planet is doomed!



DEADMAN

The time honoured game of suspense – but with some novel endings. Guess the hidden letters or something nasty could happen to you.

Computing with the Amstrad presents

CLASSIC GAMES on the Amstrad

Here's something really special from *Computing with the Amstrad!* We've commissioned four rip-roaring programs that no games collection is complete without – the kind of games that really stand out in the short history of microcomputing.

This value-for-money package includes two top-rate machine code arcade classics plus a traditional word game and a futuristic adventure.

There's hours of enjoyment and something to suit everyone in this superb collection.

Please send me Classic Games on the Amstrad Vol. 1 (✓)

Cassette £5.95 6048 ☐
3" Disc £9.95 6049 ☐

Payment: please indicate method (✓)

☐ Access/Mastercard/Eurocard

☐ Barclaycard/Visa
No. ⌊⌊⌊⌊ ⌊⌊⌊⌊ ⌊⌊⌊⌊ ⌊⌊⌊⌊

☐ Cheque/PO made payable to Database Publications Ltd.
No. ⌊⌊⌊⌊ ⌊⌊⌊⌊ ⌊⌊⌊⌊ ⌊⌊⌊⌊

Name _____ Signed _____

Address _____

_____

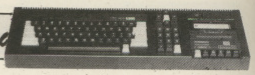
_____

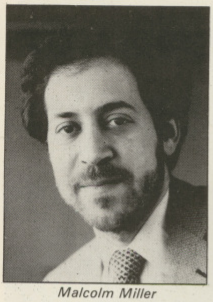
Send to:
***Computing with the Amstrad*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

You can also order by phone **061-480 0171** *24 hours*

Don't forget to quote your credit card number and full address. CA7


THE Only REALISTIC
BASKETBALL
ACTION PROGRAM FOR YOUR CBM 64
NO. 2. U.S. HIT
OUT NOW ON CASSETTE & DISK
NOW
AVAILABLE ON
SPECTRUM 48K
AMSTRAD CPC464
ATARI 600/800XL
CASSETTE BOOKS
including FREE
ENAMEL METAL BADGE
OFFER
ONE-ON-ONE
ONE-ON-ONE
From Electronic Arts
JULIUS ERVING
LARRY BIRD
PERSONAL COMPUTER GAMES HIT!
Fast Load Chrome Tape
A Basketball Action program for the COMMODORE 64 U.S. No. 2 HIT
ariolasoft
INCLUDES SPECIAL DISCOUNT VOUCHER
THIS PROGRAM HAS BEEN PERSONALLY DESIGNED BY TOP U.S. BASKETBALL SUPERSTARS – DR JULIUS ERVING & LARRY BIRD AND IS BASED ON THEIR REAL GAME MOVEMENTS.
STAR GAME
Basketball fans should love this simulation of the game—It has some of the most lifelike animation ever seen in a sports game—
■ To see your player spin 180 degrees, dribble past the defender and shoot close in, gives a vicarious thrill quite unmatched in any other sports simulation I've seen.
■ All this adds up to one heck of a great game.
Bob Chappell
Personal Computer News
FEATURES ● PLAY THE COMPUTER OR A FRIEND ● 4 SKILL LEVELS ● BUILT IN FATIGUE FACTORS – HOT & COLD STREAKS – FOULS – A STOP CLOCK – AND A REFEREE ● INSTANT ACTION REPLAY ● JOYSTICK CONTROL* ●
*and/or keyboard control (Spectrum & Amstrad versions)
AVAILABLE FROM ALL GOOD SOFTWARE RETAILERS – IF IT'S NOT THERE PLEASE ORDER IT
ariolasoft
HIGH PERFORMANCE PROGRAMS





Malcolm Miller

Marketing achievement in line for an award

THE success of the CPC464 has put Amstrad in line for another award. But this time it isn't the computer itself that has been nominated, but the achievements of sales and marketing director Malcolm Miller.

By capturing 16 per cent of the home computer market in only six months, Amstrad became one of the biggest success stories of British industry last year.

Now the Marketing Society has recognised this achievement by shortlisting Amstrad in the consumer product category of its annual awards.

In effect the Society has already said Amstrad's marketing performance was one of the UK's four best last year, by nominating it against only three others in this section of the awards.

New disc drive opens the way to using IBM's software

A MAJOR boost to Amstrad's hopes of making an impact on the business market is on its way from West Germany.

A scientist and an accountant in Bonn have designed a 5¼in second disc drive they claim will enable the CPC464 to run a significant proportion of the more than 300 IBM CP/M programs on the market.

The device is said to enable the CPC464 to run IBM software off the shelf.

It would also allow the machine to run an enormous range of other CP/M business software, the majority of which is supplied on 5¼in disc.

The add-on will be marketed in this country and overseas by Screens Microcomputer Distribution.

Director Nigel Sinclair-Miller told *Computing with the Amstrad* that he is testing an enhanced prototype of the product and hopes to make it available this summer.

"At current exchange rates the price will be less than £300", he said.

"I hope to have at least 5,000 models in stock in the very near future and we already have a worldwide dealer base to retail the product".

The drive will support the transfer of text files between the first DD1 drive to the new 5¼in drive and vice versa, says Sinclair-Miller.

But it won't be able to transfer CP/M programs from one file to another.

Machine code aid

A NEW machine code programming package for the CPC464 and 664 has been released by Picturesque.

Called Code Machine, it consists of separate assembler and monitor programs, and is based on Picturesque's previous product for the Spectrum, but with many improvements.

The programs are fully relocatable in memory and are supplied on cassette with an option to make a working back-up copy onto disc.

Code Machine comes with an 80 page manual and costs £19.95.

Accent on serious applications

MINI Office, the chart-topping software package for the Amstrad is being used as a teaching aid by leading UK computer retailers W.H. Smith.

Shortlisted for two major categories in the 1985 British Microcomputing Awards, Mini Office has been selected to bring the message home to W.H. Smith staff that computers have a serious application.

The company is currently using the program in its mobile training classroom which is on a nationwide tour.

This will mean that by the time the vehicle has completed its circuit thousands of W.H. Smith's computer and related staff will have used the package.

"We have a lot of staff and we want them to know what the goods they are selling are capable of", said a training division spokesman.

"They are not employed just to wrap up goods, but inform the customer of the computer's various applications. The computer is not just a toy".

W.H. Smith chose Mini Office because its four programs emphasise how easily a computer could be turned into an inexpensive office tool.

"At its revolutionary low price of £5.95, Mini Office is a truly cost-effective way of introducing the business concept of computers to our staff", said the man from W.H. Smith.

"Naturally, we are delighted that the computer retailing giant has demonstrated its faith in our product in this way", says Derek Meakin, head of Database Software, who created Mini Office.

ON THE RIGHT LINES...

IN supplying the new CPC664 to the retail trade, Amstrad plans to follow the successful formula which resulted in sales of 200,000 CPC464 models last year.

The company will sell direct to its traditional High Street multiple outlets and mail order houses, channelling independent retailers through Europa Electronics.

Europa is Amstrad's solely owned subsidiary set up to service the independent trade sector. Dickie Mould, Amstrad's sales director, was recently appointed managing director of Europa Electronics.

Bright business future

THE arrival on the market of Amstrad micros heralds a bright future for small businesses that want to computerise their operations, says Merrick Allsopp, managing director of Cornix Software.

"Machines capable of performing important business functions are now available at an affordable price", he told *Computing with the Amstrad.*

Allsopp is so confident of the success of the CPC464 and 664 that he has diverted a major part of his company's resources to producing business programs for the Amstrad micros.

As he puts it: "The final piece in the jigsaw is a range of equally-affordable software that will offer a professional performance within the small business environement".

Cornix has so far produced two new business programs for the Amstrad – Simple Accounts, which costs £29.95 and covers book-keeping, and Management Analysis, which costs £19.95 and is designed to extract information from the accounts on which management policy can be based.

Says Allsopp: "We plan a complete range of business software on both cassette and disc.

"Within the range will be programs performing similar functions but designed to meet the varied requirements of small businesses, particularly in the area of accounting at all levels".

Amstrad acts as memory for Chris

DESPITE being handicapped by crippling infirmities that would have crushed a lesser man, Chris Moore faces each day cheerfully, aided by his motorised wheelchair, citizens' band radio . . . and his CPC464.

Toughened by his years as a merchant seaman, soldier and HGV driver, Chris, of Walsall Wood, West Midlands, was a virtual stranger to illness until disaster struck suddenly.

A routine medical uncovered a liver problem.

Says Chris: "Tests confirmed I had a terminal liver disease with sundry other complaints.

"Just to cap it all, I have encephalopathy – deteriorating brain cells!"

Chris had to give up work immediately.

Worry about being jobless with a wife and three children to support brought on a mild stroke which resulted in a weakness in his left side.

But Chris continued to fight his disabilities.

"By constant harassment and a large amount of bluff I managed to get the doctor to sign me off as fit to work", he recalls.

"Just over 12 months later and after several near-misses with my wagon I was permanently grounded".

Having been involved with citizens' band radio since the early 1970s, the prospect of being confined to bed didn't unduly worry Chris.

Chris Moore . . . in radio-equipped wheelchair

"After legalisation of CB, I decided to have a go at the radio amateurs' exam, which I managed to pass first time.

"Unfortunately I neglected to study for the Morse Code exam and don't seem to be able to memorise it now".

Thanks to the Disabled Aid Fund, which lent him an electronic printer/typewriter, he started writing articles for various magazines catering for the disabled.

His column Grey Beard's Gleanings is a popular regular feature of Citizens' Band magazine and is devoted to helping disabled CB enthusiasts pursue their hobby.

Says Chris: "Eventually it was decided I should have a computer to back up my ever-failing memory.

"With the aid of a generous discount from Amstrad and the help of the DAF I was able to get the colour monitor version of the CPC464".

So far Chris has been unable to afford a printer, so he has concentrated on other uses for the machine. It allows him to keep his radio amateur logbook to the required standard.

Says Chris: "Obviously disc drives and a good printer will have to be acquired before I can fully utilise all of my Amstrad's skills.

"One of my first purchases will be a package that will translate Morse Code into plain English and print it on the monitor".

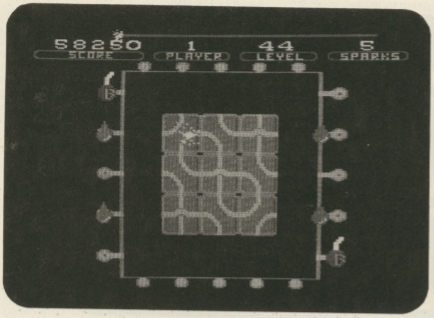
Cash keeps rolling in

MONEY Manager, from Connect Systems, enables Amstrad users to record individual financial transactions over a "rolling" 12-month period, and to view this data selectively.

It can be tailored to the user's circumstances, providing facilities for up to 100 transactions a month, up to nine accounts, and up to 50 income/expenditure classes divided into logical groups.

Each transaction has fields for date, reference, description, and a further free data item.

Price of the cassette version is £14.95.



TELESOFTWARE TAKES THE AIR

AMSTRAD owners with the kind of bright ideas that could make big money with proper financial backing have been the target of a novel advertising campaign.

The high tech section of 3i – a venture capital loan company – used selected local radio stations to broadcast one minute commercials in the form of software to be downloaded by Amstrad users.

The advert consisted of an electronic questionnaire to be answered by the Amstrad-owning, would-be entrepreneurs.

"We're mainly interested in propositions from people with ideas for computer hardware, serious software, process control and automation, semi-conductors and telecommunications", a 3i spokesman told *Computing with the Amstrad*.

"We'd like to give them a helping hand to become rich".

Explosive situation

THE first chance to defuse an explosive situation has been given to Amstrad users by Incentive Software.

It comes in the form of their new arcade game, Confuzion, which is set in a huge, 64-storey industrial plant used for the production and storage of deadly bombs.

The object for the user is to destroy the complex by eliminating the entire explosive stockpile. But chances are limited and depend on how skilful the play is.

Confuzion, price £6.95, also features bonus levels, extra sparks, speed spark, skill-stepping and multiplayer options.

Play with Robson

BRYAN Robson, the Manchester United and England captain, is the latest in a seemingly endless line of soccer celebrities to lend his name to a computer football game.

And this one is aimed at Amstrad owners.

Known as "Bryan Robson's Super League", the publicity release put out about it claims that it is "extremely exciting" with the league championship in doubt until the last game.

However it is different than most of the others currently on the market – Bobby Charlton's to name just one – in that it not only includes a cassette but also a board, dice, counters and other accessories.

While the Amstrad is called upon to control the fixtures and the way the games are played, the board is used for "away travel and the buying and selling of stars".

The object of the game is to win the league.

And it seems as though the publicity firm who wished to notify *Computing with the Amstrad* about the game has scored an own goal right from kick off. They forgot to mention how much it costs.

A busy man is Barnes

REMEMBER the name Barnes . . . he is the hero chosen by Cable Software to feature on a new series of complex arcade games for the Amstrad.

First to be released is Digger Barnes. In the game, Barnes has to dig himself out of trouble – trapping an increasing number of pursuing monsters in his self-made holes. Once in, he has to kill them off with a blow from his shovel.

As he escapes to higher levels, using ladders and lifts, the game becomes increasingly difficult.

A possible follow-up is Lancelot Barnes, in which the hero uses his jousting skills to escape the enemy.

Digger is written entirely in machine code, features high resolution sprite graphics, multi-screens, colour and sound and costs £7.95.

Sales team reorganised

AMSTRAD has reorganised its sales and marketing team to provide more efficient service for customers who buy from independent retailers.

Marketing director Malcolm Miller has been appointed sales and marketing director.

And former sales director Dickie Mould has become managing director of distributor Europa Electronics, Amstrad's subsidiary servicing small dealerships.

Miller's new department includes Mark Simons as sales manager with responsibility for all accounts, sales administration manager Steve Jones, software sales manager Mike Mordecai, export sales manager Joe Oki and overseas coordinator Sally Burdell.

Miller told *Computing with the Amstrad*: "We are growing rapidly and profitably and expect the newly-launched CPC664 and audio models to contribute strongly in the coming year".

FIG Forth released

KUMA Computers has released a version of FIG Forth for the CPC464.

Added features include floating point, string features, real time clock, high-level access to firmware, colour graphics, sound, function key support, stack printing utility, cursor-controlled screen editor, tape-based virtual memory system, primitive word generation, and full compatibility with ROMs and RSXs.

Supplied with the £19.95 cassette is a 120-page manual which includes an introduction to Forth, programming techniques, extensions and a glossary of Forth words.

Pixel test

VORTEX Software is making three games available for the CPC464 this summer.

Highway Encounter is a 3D, 32-screen, pixel movement test of skill and reactions and will cost £8.95.

The firm is also converting Android Two and TLL for the Amstrad and these will retail at £7.95 each.

Tame your loops with a crafty counter

By PETE BIBBY

OBSERVANT readers may have noticed that Mike Bibby has been replaced by Pete Bibby as the author of this series. It's not a misprint, nor a sign of a putsch at Computing with the Amstrad.

It's more a sort of negative nepotism. Mike's done the first six months articles and handed them over to me with a "now get out of that!"

The result is that I've been reading through the back issues trying to find out what he's covered. And very interesting it is. So, from now on I'm assuming that you know all about CLS, NEW, LIST and PRINT and that if I tell you to GOTO you won't take offence.

You should know your RUNs from your REMs, and be able to use INPUT WHILE WENDing your way round loops and taking the TAB LETs as usual. IF you understand all that, THEN you shouldn't have any problems following the rest of the series.

I'll be sticking to Mike's easy-to-follow style and the emphasis will still be on a hands-on approach. And that's where you come in.

Don't just read about it, try it out on the micro. And don't stick to the programs I give you. Try varying them or making up your own. And don't take anything I say for granted, test it out in practice. If you want to know "What if..." try it out on the Amstrad.

And now to begin. Type in Program I and see what it does.

```
10 REM Program I
20 counter=1
30 PRINT counter
40 LET counter=counter+1
50 IF counter <=25 THEN GOTO 30
```

Program I

It's not exactly the world's most fascinating piece of programming but it does cover some important points. With what you've done already you shouldn't have any difficulty in seeing how it works.

Line 20 sets up the variable *counter*, giving it the value 1 while line 30 displays it. Line 40 adds one to *counter* storing the result back in *counter*. This means that *counter* is now one bigger than it was when it was displayed.

Now comes the clever bit. The IF of the next line checks to see if *counter* is less than or equal to 25. If it is, the GOTO after the THEN sends the program back to line 30 and the whole process begins again.

When *counter* is greater than 25 the loop comes to an end. Technically the program is said to drop out of the loop. It then goes on to process the next line. In this case there isn't one, so things grind to a halt.

The program is an example of what's known as a conditional loop, the condition being that *counter* must be less than or equal to 25 for the loop to cycle. The result is that the numbers 1 to 25 are printed out and then the program ends. Can you guess the value of *counter* when the program ends? Try adding:

```
60 PRINT "At the end counter is
"counter
```

and you'll find that the result is 26. You may have thought that it should be 25 but a quick look at the condition of line 50 should disabuse you.

As the condition is less than or equal to 25, when *counter* is 25 the program will still obey the GOTO. It loops back to line 30 and prints *counter*. The next line adds one to this and so *counter* now has a value of 26 when it comes to the test of line 50. As 26 is greater than 25 the test is failed, the condition is false and the GOTO ignored. The program drops out of the loop and, finding no more lines to process, ends.

Try changing line 50 to:

```
50 IF counter <25 THEN GOTO 30
```

or

```
50 IF counter=25 THEN GOTO 30
```

and try to figure out what's happening. As you can see, seemingly small changes in the conditions of loops can have big effects on the way the program works.

You'll find that a lot of the problems you get when you use IFs and GOTOs to create loops lie in the use of the wrong conditions. You have to get them dead right which, in more complicated programs, isn't always easy.

Now having read all that lot you should have no difficulty in understanding why Program II says HELLO 25 times.

```
10 REM Program II
20 counter=1
30 PRINT "HELLO"
40 LET counter=counter+1
50 IF counter <=25 THEN GOTO 30
```

Program II

Use your knowledge to produce five HELLOs, then five thousand. Or maybe three hundred and five. Should the condition be less than 306 or less than or equal to 305 or will both do the job or will neither? Try it and see. When you've finished playing around, run Program III.

The output from this is exactly the same as that from Program I, yet it's one line shorter. How is this done?

The answer lies in the FOR of line

20 and the NEXT of line 40. These form what is known as a FOR . . . NEXT loop.

All the program lines that come between the FOR and the NEXT that

```
10 REM Program III
20 FOR counter=1 TO 25
30 PRINT counter
40 NEXT counter
```

Program III

follows it become part of the loop. In fact they're known as the body of the FOR . . . NEXT loop. And, as we've come to expect with the loops that we've already dealt with, it cycles over and over.

Each time round the lines making up the body of the loop are performed. In this case there's only one line sandwiched between the FOR and the NEXT. This is line 30 which prints the value of *counter*.

Obviously, the value of *counter* is varying as the loop progresses but how? Previously we've always had something like:

```
counter=counter+1
```

but there's nothing like that in program III. And how does the loop know when to finish? And, while we're at it, what's the initial value of *counter?*

The answers lie in line 20 where the FOR is followed by:

```
counter=1 TO 25
```

It's this line that tells the Amstrad the initial value of *counter* which, in this case, is one. As well as this it automatically adds one to its value each time round the loop and also determines how many times it will cycle. Let's look at it more closely.

As we've seen, the FOR and NEXT bracket the body of the loop, the lines that are to be repeated, Notice that the FOR is followed by a variable, in this case it's *counter.* This is grandly known as the loop control variable, and it's this that decides how many times the loop will cycle.

The control variable starts out at the value assigned to it after the FOR and each time round the loop it is increased in value by one. The loop stops when the control variable exceeds the value specified by the figure after the TO.

In other words, the control variable acts as a counter. If we wanted to do something 10 times we could use our fingers as a control variable. Looking at it as a FOR . . . NEXT loop, we'd probably have something like:

```
FOR finger=1 TO 10
do something
NEXT finger
```

By the time we've used all 10 fingers we'll have done whatever it was we were doing ten times. The finger has acted as a control variable ranging from 1 TO 10.

In Program III we have the line:

```
FOR counter=1 TO 25
```

This tells the Amstrad that it is to repeat all the lines up until the following NEXT. The loop control variable is to be *counter,* its initial value will be 1 and each time round the loop this is to be increased by one. The loop is to cycle until *counter's* final value is greater than 25.

The first time round the loop *counter* is 1. Then the NEXT sends the program back to the FOR and *counter* becomes 2, then 3 as the loop progresses and so on until it's 25.

Now when the NEXT sends the program back to the FOR the 1 that is added to *counter* gives it the value 26. This exceeds the limit put on it by the TO so this time the loop is skipped, the Amstrad going to the line after the NEXT. In this case there isn't one, so the program finishes.

The result is that all the numbers between 1 and 25 are printed, as the loop cycles FOR values of *counter* from 1 TO 25. Try adding:

```
50 PRINT "The final value of counter
is "counter
```

and you'll see that it really has a final value of 26.

Compare Program I with Program III. I think that you'll agree that Program III is a lot clearer. With the FOR . . . NEXT structure it's easy to see that the loop will cycle 25 times. With Program I you're never quite

```
10 REM Program IV
20 FOR counter=1 TO 25
30 PRINT "HELLO"
40 NEXT counter
```

Program IV

sure if it's going to be 24 or 25 times. So the lesson is that FOR . . . NEXT loops make things a lot clearer. They're also a lot easier to adjust than loops cobbled together with GOTOs.

Take a look at Program IV which does the same job as Program II but uses a FOR . . . NEXT loop.

You still get your 25 HELLOs on screen just as with Program II. Notice, however, how easy it is to get 500 or 5,000 of them. And to get 305 all we do is change one line.

```
20 FOR counter=1 TO 305
```

gives us what we want. As you can see it's a lot easier than messing about with Program II.

FOR . . . NEXT loops are not only simple, they're also powerful. Take a look at Program V and you'll see what I mean.

```
10 REM Program V
20 FOR counter=1 TO 25
30 PRINT counter, counter*2, counter*
counter
40 NEXT counter
```

Program V

Again the loop cycles 25 times, but now the line that makes up the body of the loop is different. It not only prints the value of *counter,* it also doubles it and shows its square. Try altering the line so that it gives the cube of *counter* or adds five each time round. Simple isn't it?

Before we leave Programs III and IV just see if you can spot any major difference between the way that they work. Don't worry if you can't, it's fairly obscure.

The answer is that Program III not only uses *counter* to control the number of loop repititions but also uses *counter* in the body of the loop. Program V does the same thing.

Program IV however doesn't use *counter* in the main body of the loop, it just uses it to keep track of the cycles. As I said, it's an obscure point but one worth bearing in mind.

If you use the loop control variable inside the main body of the loop be careful that you don't change it or you'll upset the loop's counting.

A line like:

```
25 counter=counter+3
```

in Program III will show you what I mean. Although it's fairly obvious in this case, in longer programs it can occasionally be a problem. Beware!

So far all our FOR . . . NEXT loops have had the control variable going up in steps of 1 each time round the loop. It doesn't always have to be like this. You can make the change any number you want using the aptly-

named STEP command. Program VI uses it to display all the even numbers between 0 and 20.

```
10 REM Program VI
20 FOR counter=0 TO 20 STEP 2
30 PRINT counter
40 NEXT counter
```

Program VI

It's a simple command to use. As you can gather from the program, the figure after the STEP decides how much the control variable is to be incremented at the end of each cycle. In this case *counter* is increased by 2 each time round.

What would happen if the STEP was 4? Try and figure it out before you attempt it on the micro.

Program VII shows what happens when the STEP is 3.

```
10 REM Program VII
20 FOR counter=0 TO 20 STEP 3
30 PRINT counter
40 NEXT
```

Program VII

There's no problem with most of the output. It starts at 0, and then goes up in threes until it reaches 18. But why stop at 18? If you add:

```
50 PRINT "The final value of counter
is "counter
```

to the end of the program you should see why.

When *counter* is 15 the NEXT adds 3 to it, making it 18 and then sends the loop back to line 20. Line 30 prints out the 18 and the program again comes to the NEXT.

Here *counter* is again increased by three, making it 21 and the program goes back to the FOR.

Now when the micro checks, *counter* is outside the allowed range (0 to 20) so the loop is skipped. Hence 18 is the last number you see printed.

Try changing the STEP to 5 or 8 and see if you can explain the results.

You may have noticed that in the last program the NEXT of line 40 didn't have a *counter* by it but it still worked. From this you can see that you don't always have to label the NEXT with the loop control variable. Having said that I'd advise you to always do so, it makes debugging programs a lot easier.

All this is all right if you want the numbers produced by a loop to increase, but what if, for reasons best known to yourself, you want the numbers to decrease in value?

Program VIII shows one way of doing this.

```
10 REM Program VIII
20 FOR counter=0 TO 15
30 PRINT 15-counter
40 NEXT counter
```

Program VIII

Here *counter* is still going from zero upwards by one each time, its range spanning 0 to 15. However, instead of just printing out the value of *counter,* now we take it away from 15 each time round the loop. The first time round *counter* is 0 so 15 is displayed. The second time *counter* has increased to 1 so 14 (15-1) appears with 13 (15-2) following it on the next cycle.

As *counter* ranges from 0 to 15, so the result of the subtraction goes from 15 to 0. Can you make it go down in twos?

Program IX shows an easier way of counting down.

```
10 REM Program IX
20 FOR counter=15 TO 0 STEP -1
30 PRINT counter
40 NEXT counter
```

Program IX

The secret lies in the STEP parameter of –1. Each time round the loop –1 is added to *counter.* The result of adding –1 each time round the loop is that *counter* is decreased.

Notice that the figures on either side of the TO have changed to cope with this. They now go from an initial value of 15 to a final one of 0.

Try using other negative steps and see what happens. They're quite simple to use, but make sure that the start and finish values match the steps.

If you don't understand that try Program X and you'll see what I mean.

```
10 REM Program X
20 FOR counter= 2 TO 1
30 PRINT "That's silly!"
40 NEXT counter
```

Program X

Try:

```
STEP -1
```

or

```
FOR counter=1 to 2
```

to correct it.

By now you should be fairly at home with FOR . . . NEXT loops, so program XI should hold no fears for you. You should be able to figure out that the last number displayed will be 9.

```
10 REM Program XI
20 FOR counter=1 TO 10 STEP 2
30 PRINT counter
40 NEXT counter
```

Program XI

Notice that when the STEP is positive the loop finishes when the control variable exceeds the limit set.

If you've any doubts:

```
50 PRINT "The last value is "counter
```

should convince you of the fact as 11 is certainly greater than the 10 after the TO.

However, when the STEP is negative as in Program XII things are different. The loop finishes when the control variable is less than the figure after the TO.

```
10 REM Program XII
20 FOR counter=10 TO 1 STEP -2
30 PRINT counter
40 NEXT counter
```

Program XII

Again:

```
50 PRINT "The last value is "counter
```

added to the end of the program should convince you. There's no doubt that 0, the final value of *counter* is less than 1, the limit set in line 20.

And that's more or less the end for this month. Until next time why not write your own programs using negative and positive steps? And why not use fractional values instead of the whole numbers we have been doing? And if, after all this, you have any time left, can you explain Program XIII?

```
10 REM Program XIII
20 FOR outer=1 TO 3
30 PRINT "This is outer loop "outer
40 FOR inner=1 TO 3
50 PRINT TAB(15)"This is inner loop "
inner
60 NEXT inner
70 NEXT outer
```

Program XIII

What's happening here? All will be explained next month.

ACCESS & VISA ORDERS: 0227 266289

CUT OUT & KEEP THIS AD!

AMSTRAD USERS!

Having difficulty getting software for your CPC464 or 664? Then let us help you! **The Software Supplies Company** is a specialist trading division of a long-established mail-order software supplier with a worldwide customer base of thousands. Shop by mail-order in confidence and experience a speedy and personalised service. Overseas orders a speciality. New catalogue giving full product details available soon, free to all customers on our mailing list.

Please make all cheques payable to **The Software Supplies Co.** Please add 50p P&P for all orders under £20, and £1 for heavy items marked with a dagger (†). For Europe add £1.00 per item, and £2 each for heavy items (marked †). Airmail elsewhere at cost. Please read footnote.

- **Business Educational and Recreational Software for all home and business computers —**
- **Keen prices — excellent choice**
- **Send for lists (state machine type) or, if you want something quickly, phone your order through now! We're first with the newest!**

APPLICATIONS

Advanced Amsword(†)	Amsoft	19.95
Amscalc(†)	Amsoft	19.95
Amscalc Xtra(†)	Amsoft	34.95
Amsword(†)	Amsoft	19.95
Cashbook Accounting(*)(†)	Gemini	d65.95 59.95
Database	Gemini	d25.95 19.95
Database	Kuma	14.95
Decision Maker(*)(†)	Amsoft	d29.95 24.95
DFM Database + Labels	Dialog	14.95
Easi Amsword Word Processor	Amsoft	9.95
Easi VAT Accounts	Kuma	39.50
Entrepreneur(*)(†)	Amsoft	d29.95 29.95
Flexifrend (Home Budgeting)	Camel Micros	7.50
Home Budget	Kuma	19.95
Home Accounts Manager	Dialog	14.95
Home Accounts	Gemini	d25.95 19.95
Investat	Dialog	29.95
Mastercalc 464(†)	Campbell	d29.95 24.95
Masterfile 464(†)	Campbell	d29.95 24.95
Masterfile Extension	Campbell	10.00
Microgen(†)	Amsoft	d49.00
MicroScript(†)	Amsoft	d49.00
MicroSpread(†)	Amsoft	d49.00
Mini Office	Database	5.95
Money Manager	Connect Systems	14.95
Project Planner(*)(†)	Amsoft	d29.95 24.95
Report Generator	Gemini	d25.95 19.95
Stock Aid	Dialog	29.95
Tasprint 464	Tasman	9.90
Tasword 464	Tasman	d19.95
Transact	Dialog	29.95
VAT File(*)	Gemini	d25.95 19.95

UTILITIES/LANGUAGES

Abersoft FORTH(*)	Amsoft	24.95
Artist + Sprite Designer	CRL	9.95
Artwork	Kuma	9.95
Assembly Language Course(†)	Honeyfold	12.50
Azimuth 2000 (head alignment)	Interceptor Micros	8.99
BASIC Programming Course(†)	Honeyfold	10.50
Concise BASIC Spac(†)	Amsoft	11.95
Concise Firmware(†)	Amsoft	19.95
CPC464 Meddler (copier)	Nemesis	4.50
Devpac Ass/Disassembler	HiSoft	24.95
Font 464	HiSoft	7.95
FORTH	Kuma	17.95
Graphics Designer	Amsoft	11.95
Grasp (Graphics Package)	Camel Micros	8.50
Guide to Amstrad BASIC 1(†)	Amsoft	19.95
Guide to Amstrad BASIC 2(†)	Amsoft	19.95
Guide to CP/M(†)	Amsoft	tba
Guide to LOGO(†)	Amsoft	tba
Introduction Manual(†)	Amsoft	9.95
Introduction to Sound 1	Amsoft	11.95
LOGO Turtle Graphics	Kuma	19.95
Machine Code Tutor	New Generation	14.95
Music Composer	Kuma	9.95
Operation Caretaker (demag/align)	Global Software	9.95
Pascal	HiSoft	29.95
Screen Designer(†)	DJL Software	11.95
Speech Synthesiser (+ speakers)(†)	DK'Tronics	39.95
The Quill	Gilsoft	16.95
Toolbox (Graphics & Sprites)	Camel Micros	4.95
Zen Assembler	Kuma	19.95
ZAPP Z80 Assembler Prog Package	Hewson	14.95

EDUCATIONAL

Animal Vegetable Mineral	Bourne Educational	8.95
First Steps with Mr Men	MirrorSoft	8.95
French is Fun (+ audio)(*)	CDS Software	7.95
German is Fun (+ audio)(*)	CDS Software	7.95
Happy Letters	Bourne Educational	8.95
Happy Numbers	Bourne Educational	8.95
Happy Writing	Bourne Educational	8.95
Here and There with Mr Men	MirrorSoft	8.95
Italian is Fun (+ audio)(*)	CDS Software	7.95
Map Rally	Bourne Educational	8.95
Osprey!	Bourne Educational	8.95
Rhyme Land	Anirog	tba
Spanish is Fun (+ audio)	CDS Software	7.95
Star Watcher	Amsoft	d23.95 19.95
Time Man One	Bourne Educational	8.95
Time Man Two	Bourne Educational	8.95
Word Hang	Bourne Educational	8.95
World Wise	Bourne Educational	8.95

GAMES

3D Invaders	Quark Software	8.95
3D Star Strike	Real Time	6.95
3D Time Trek	Anirog	7.95
5-A-Side Soccer	Amsoft	8.95
Admiral Graf Spee	Temptation	8.95
Adventure Quest	Level 9	9.95
Alien Break-In	Romik	6.95
Alien 8(*)	Ultimate	9.95
American Football	Mind Games (Argus)	9.95
Amsgolf	Computersmith	d12.95 8.95
Android One	Vortex	7.95
Android Two(*)	Vortex	7.95
Angelique: A Grief Encounter	Nemesis	6.50
Animated Strip Poker(*)	Knightsoft	8.95
Archon(*)	Ariolasoft	10.95
Arnold Goes To Somewhere Else!	Nemesis	6.50
Atom Smasher	Romik	8.95
B Jacks Superstar Challenge(*)	Martech	7.95
Backgammon(*)	CP Software	8.95
Battle for Midway	PSS	9.95
Blagger	Alligata	7.95
Bridge Player -2	CP Software	9.95
Bridge	Kuma	8.95
Bridge-It	Epicsoft	8.95
Caesars Travels	Mirrorsoft	8.95
Castle Blackstar(*)	CDS Software	6.95
Centre Court	Epicsoft	d12.95 8.95
Chopper Squad	Interceptor	6.00
Chuckie Egg	A&F Software	7.95
Classic Adventure	Abersoft	8.95
Classic Racing	Amsoft	7.95
Climate	Tynesoft	8.95
Codename Mat	MicroMega	d12.95 8.95
Colossal Adventure	Level 9	9.95
Combat Lynx	Durrell	11.95
Crazy Golf	Mr Micro	8.95
Cubit	Mr Micro	8.95
Daley Thompsons Decathlon	Ocean	8.95
Dark Star	Design Design	7.95
Death Pit	Durrell	7.95
Defend or Die	Alligata	7.95
Detective (Cluedo)	Amsoft	8.95
Digger Dannes	Cable Software	8.95
Dragons Gold	Amsoft	7.95
Draughts(*)	CP Software	8.95
Dungeon Adventure	Level 9	9.95
Electro Freddy	Softspot	d12.95 8.95
Emerald Isle	Level 9	8.95
Erbert	Microbyte	5.95
Eric the Viking	Mosaic	9.95
Everyone's a Wally(*)	Mikro-Gen	9.95
Fantasia Diamond	Hewson	7.95
Fantastic Voyage	Quicksilva	6.95
Fighter Pilot	Digital Int	8.95
Flight Path 737	Anirog	6.95
Football Manager	Addictive	7.95
Forest at World's End	Interceptor	5.95
Fruity Frank	Kuma	6.95
Galactic Plague	Indescomp	8.95
Galaxia	Kuma	8.95
Gems of Stradex	Kuma	8.95
Ghostbusters	Activision	10.99
Ghouls	Micropower	7.95
Gilligans Gold	Ocean	6.90
Grand Prix Driver	Brittania	8.95
Hard Hat Mack(*)	Ariolasoft	10.95
Hareraiser Prelude	Haresoft	8.95
Hareraiser Finale	Haresoft	8.95
Harrier Attack	Durrell	d12.95 8.95
Haunted Hedges	MicroMega	8.95
Heathrow ATC	Hewson	7.95
Herbert's Dummy Run(*)	Mikro-Gen	9.95
Heroes of Karn	Interceptor	5.95
Hold Fast	Kuma	8.95
House of Usher	Anirog	6.95
Hunchback	Ocean Software	d12.95 8.95
Hunchback II	Ocean Software	8.95
Hunter Killer	Protek	d12.95 8.95
Interdictor Pilot	Supersoft	17.95
Jammin'	Tasket	8.95
Jet Set Willy	Software Projects	7.95
Jetboot Jack	English	7.95
Jewels of Babylon	Interceptor	5.95
Johnny Reb	Lothlorien	6.95
Jump Jet(*)	Anirog	9.95
Killer Gorilla + Gauntlet	MicroPower	9.95
Knight Lore	Ultimate	9.95
Kong Strikes Back	Ocean	8.95
Laser Warp	Mikro-Gen	8.95
Lords of Time	Level 9	9.95
Magic Sword (+ book)	Database	8.95
Manic Miner	Software Projects	7.95
Master Chess	Mikro-Gen	d12.95 9.95
Match Fishing(*)	Alligata	9.95
Message from Andromeda	Interceptor	5.95
Mighty Magus	Quicksilva	6.95
Millionaire	Incentive	6.95
Minder	DK'Tronics	9.95
Moon Buggy	Anirog	7.95
Mr Wong's Loopy Laundry	Artic Computing	8.95
Mutant Monty	Artic Computing	d12.95 8.95
Oh Mummy	Amsoft	8.95
One-On-One(*)	Ariolasoft	9.95
Pinball Wizard	CP Software	9.95
Punchy	Mr Micro	d12.95 8.95
Pyjamarama	Microgen	d12.95 8.95
Quackajack'	Amsoft	d12.95 8.95
Redcoats	Lothlorien	6.95
Return to Eden	Level 9	9.95
Ring of Darkness	Wintersoft	7.95
Roland Ahoy	Gem Software	d12.95 8.95
Roland Goes Digging	Gem Software	8.95
Roland Goes Square Bashing	Gem Software	8.95
Roland in the Caves	Gem Software	d12.95 8.95
Roland in Space	Gem Software	d12.95 8.95
Roland in Time	Gem Software	d12.95 8.95
Roland on the Ropes	Gem Software	d12.95 8.95
Roland on the Run	Gem Software	d12.95 8.95
Shadow Fire(*)	Beyond	9.95
Shadow of the Bear	Kuma	8.95
Sherlock Holmes(*)	Melbourne House	14.95
Sir Lancelot	Melbourne House	5.95
Snooker	Gem Software	d12.95 8.95
Snowball	Level 9	9.95
Software Star	Addictive Software	7.95
Sorcery	Virgin	8.95
Space Hawks	Durrell	8.95
Spanner Man	Gem Software	d12.95 8.95
Special Operations	Lothlorien	6.95
Splatt	Incentive	d12.95 8.95
Star Avenger	Kuma	8.95
Star Commando	Terminal	8.95
Star Clash	Amsoft	8.95
Steve Davis Snooker	CDS Software	7.95
Stockmarket	Argus Press	8.95
Sultan's Maze	Gem Software	8.95
Super Pipeline III(*)	Tasket	8.95
Superchess 3.5	CP Software	9.95
Survivor	Anirog	7.95
Tank Busters	Design Design	7.95
Technician Ted	Hewson	7.95
The Hobbit	Melbourne House	14.95
The Trial of Arnold Blackwood	Nemesis	6.50
The Wise & Fool of A Blackwood	Nemesis	6.50
Tornado Low Level (TLL)(*)	Vortex	7.95
Tripods	Redshift	11.50
Uncle Claude	Alligata	7.95
Whirlinurd(*)	Ocean	6.90
World Cup Football	Artic	7.95
Xanagrams	Dean Software	8.95
Zodiac	Anirog	6.95
Zorakk The Conqueror	Icon Software	6.95

The list above doesn't include **everything** for the CPCs – there's an ever increasing list of disc-based software, for instance. So if you are hunting for an old classic, or something scorchingly new, or simply obscure . . . do try us! Note that a single asterisk (*) indicates new or imminent release. Cheques of credit card orders for the latter will not be processed until goods are actually ready for despatch. Please note that the US Gold titles are not expected until June, starting with BeachHead. But believe it when you actually read the reviews!

If the hotline answerphone is in use and you'd rather not place an order on the machine please leave your name and number and someone will attempt to return your call on weekday afternoons or early evenings. **Remember to state machine type.**

SOFTWARE PRODUCERS: Please let us know of new products — planned or released — for inclusion in a detailed catalogue and our adverts.

the software supplies company

p.o. box 19, whitstable, kent ct5 1tj

SOFTWARE Competition!

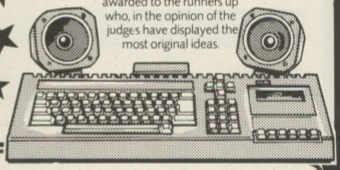
Anyone can enter and submit as many games as they wish and WIN more than one prize. Simply send any game you have written to Cascade before 30th August 1985 and you're well on your way to winning a super prize. Entries can be in machine code, basic or basic with machine code sub-routines.

2 FIRST PRIZES

Fabulous Amstrad DD1-1 Disc Drives (with CP/M and Dr. Logo) and interface, for the best machine code game and the best basic game.

10 RUNNERS-UP PRIZES

SUPERB STEREO SPEECH SYNTHESIZERS for your Amstrad which can make your computer talk with the minimum of programming. These 10 speech synthesizers will be awarded to the runners up who, in the opinion of the judges have displayed the most original ideas.



PLUS 100 CONSOLATION PRIZES

cascade microlern Head Cleaners specially designed to remove damaging oxide deposits left on the playback head by program tapes.

HOW TO ENTER

Entry couldn't be easier, once you are satisfied that the game's program runs as intended, send your tape/disc(s) to Cascade in a suitable envelope, accompanied with the completed form below.

***NOTE:-**
The first 2 prizes (DD1-1 Disc Drives) will be awarded to the best machine code game and the best basic game.

RULES

1. There is no limit to the number of entries an individual can make.
2. All entries must be accompanied with the entrant's own name and address.
3. All entries will be examined and the judges will award the first prizes to the entrants who, in their opinion have written the most interesting and creative game. All runner up prizes will be awarded to those entrants the judges deem to be the next best entries in order of merit.
4. Prizes must be taken as offered – there can be no cash alternative.
5. Prizes do not include installation.
6. Entries arriving after August 30th 1985 will be disqualified as will damaged or mutilated games.
7. No responsibility can be accepted for entries lost or delayed in the post or otherwise.
8. All competition entries are the sole property of the competition organizers, games can only be returned accompanied with a S.A.E.
9. No correspondence will be entered into with the competition organizers.
10. The decision of the judges in all matters affecting the competition will be final and legally binding.
11. All winners will be notified by post and the results published in a later edition of "Computing with the Amstrad".
12. All programs must be submitted in an unprotected form.

Cut out the form and return to Cascade Games Ltd., 1-3 Haywra Crescent, Harrogate, North Yorkshire. HG1 5BG.

cascade

Name _____

Address _____

_____

Postcode _____ Tel. _____

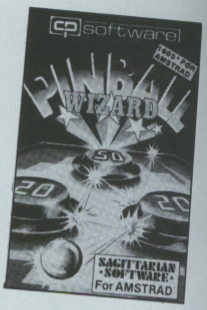
Tape(s) ☐ Disc(s) ☐ Number of entries ☐

The signature of a parent or guardian is required where the entrant is under 18 years of age.

Signature _____ Date _____

Program name _____

Pinball gets up-to-date



THE game of pinball was king of the arcades long before any of today's electronic wizardry arrived on the scene. Sagittarian Software has recently released **Pinball Wizard,** a computerised version of the arcade favourite.

The table appears quite squat when compressed into the Amstrad's screen.

A long row of "rollovers" fills the top of the screen. Below these are two sets of bumpers laid out in diamond shapes.

The only other features of any note are two sets of targets, one set marked "High" the other set "Low".

Hitting one of the high targets doubles the scoring value of the bumpers. A subsequent hit on one of the low targets returns the bumpers to their original scoring value.

Before beginning a game, you must select the speed of ball you wish to play with.

The controls are simple: any key on the bottom row from Z to B will move the left flipper, and from N to / will move the right flipper. Either of the Shift keys can be used to fire the ball.

Having enjoyed a mis-spent youth, I have played the odd game of pinball in the past. One thing you soon learn is that the flippers are not solely for belting the ball with.

By catching the ball with the flipper up you can stop it dead. Releasing the flipper allows the ball to roll down towards the end.

By "flipping" at any point as the ball rolls down its length you can aim it at any point on the table.

To some degree these tricks can be carried out quite successfully in this computerised version.

Another area in which the game scores highly is the movement of the ball.

The angles of the ricochets and the changes in speed as it cascades through the bumpers are all impeccable.

Unfortunately this wonderful programming has been somewhat wasted due to the lack of any variation in the game.

The reason the real pinball machines have survived for so long is because the manufacturers are always producing new and more lavish machines.

One of their more popular ideas has been the multi-level tables. Something along these lines could have been implemented with great effect on the Amstrad.

Sagittarian Software has got the makings of a great game, but in its present form pinball wizards may find the shine starts wearing thin after a few hours.

Jon Revis

One way to the stars...

WITH **Star Watcher,** from Amsoft, you get two programs plus a 50-page manual for your money.

The manual attempts to explain the "way the sky is mapped". However the First Point of Aries is "not of critical concern"?.

The key phrase is in Section 3.1 entitled Stars Don't Move.

"We all know that stars don't move – the earth turns and generates the illusion. In fact the stars are moving, but far too slowly (to be noticed by us).

"The illusion of sum (sic) and star movements is complicated by one additional issue – the earth also travels around the sun".

There are at least four errors in this garbled sentence.

For example the illusion of the sun's movement is CAUSED rather than complicated by the earth's tilt and orbit relative to the sun.

Also "winking" stars (or eclipsing binaries) such as Algol are noticeable BECAUSE they are moving.

The manual cites as one reference the "Guinness Book of Astronomy Facts and Feats".

This book contains vastly more information – it gives not only the Sky Coordinates of more than 1,000 stars but also their approximate distances (accurate to 10%) plus the radial velocities of the nearest 30.

I bought a copy for £2 whereas, for an order of magnitude more, you get the coordinates of 1,482 stars that could move but don't.

Admittedly you can ask for a view at any time in any direction "at any point on the earth's surface" (except the North and South Poles) but that's all you get.

"Another exciting feature allows you to see the pattern of starts (sic) which is obscured by sunlight during the day".

Talk about tautology and surely moonlight (reflected sunlight) is almost as obscure as that quote.

Personally I WANT to see the sun against the pattern of stars. This is not too difficult and it is after all a star.

Then I could practise solar and stellar navigation, a practical application of an animated computer display that a book of star patterns cannot provide.

To further damn with faint praise, the views do come up quickly – the authors claim 1,000 stars per second – yet a Basic program could only do two stars per second.

Actually I have seen a Basic program handle 500 stars at 50 per second – a quite acceptable rate – including views of the sun and planets *and* identifying the planets and main stars by name via a pointer.

The main feeling is disappointment at the almost total lack of imagination used to display such information on a computer.

I want the stars to move – either at Warp Factor 7 or by speeding up time.

These displays are so static you might just as well buy a book by Patrick Moore – it would certainly explain celestial coordinates more accurately and contain far more information. Or buy a Planisphere.

Not recommended.

Dave Alliss

Jack is one to note..

FROM the title of the game you could be forgiven for thinking that **Jet Boot Jack,** from English Software, was a space-style shoot-em-'up game.

Actually it is about a little chap making his way through

a record factory collecting musical notes.

He does have some jet boots, though. They help him out-run the bugs and gremlins that aim to stop him acquiring the greatest music collection in the world.

Each of the ten screens is composed of five floors, some of which contain sliding platforms, and all of the floors are connected by lifts.

Musical notes are suspended above every floor and Jack collects these by walking through them.

Collect every note on the screen and you progress to the next level.

Jet Boot Jack, looking like an American cop, is a brightly-coloured and detailed character. He can move left and right, duck to dodge stalactites hanging from the roof, and jump up and down.

One occasion you'll want to jump up and down is when a bug is hanging from the ceiling of the level directly below your feet.

The vibration dislodges him and he falls to his death, providing some bonus points.

When moving around it is much safer to move the Jack in a permanent stoop as in this way you don't need to pay too much attention to the stalactites.

Unfortunately ,it would appear that the programmer thought of this little ploy, because should Jack be held down for more than a couple of seconds be begins to bob up and down as he moves along.

The level of power in Jack's boots is displayed constantly at the bottom of the screen.

As it decreases, an alarm begins to sound and you must replenish his supply.

Jack recharges his boot-power by dipping his head in one of the drips of wet vinyl that hang from the roof. He's obviously never heard of Duracell.

The demo routine displays each of the 10 screens for a few seconds. However by pressing the spacebar, the pause game option, you'll have time to make note of any likely troublespots.

Just recently I have been tiring of ladders-and-levels

style games. However, Jet Boot Jack whipped up my enthusiasm with a vengance.

Every one of the screens is both detailed and complex.

Your failure to quickly assess the troublespots on each new level could easily mean stepping on to the wrong conveyor and being whisked headlong into a stalactite or a hanging gremlin.

This is an excellent game, an opinion borne out by the fact that it has already sold well on both the Commodore 64 and the BBC Micro.

Carol Barrow

Drawing for duffers

ARTWORK by Kuma, is a package designed to allow even the most inexperienced user to create graphic displays.

The program displays a cross-hair cursor, which is moved around the screen using the cursor keys.

Straight lines can be drawn by pressing the Shift key as the cursor is moved.

Alternatively, pressing Ctrl in conjunction with the cursor keys allows a rubber banding technique to be employed.

Using these methods, it is very easy to produce simple or complex line drawings.

Unfortunately the program does not allow the use of joysticks, which would have been useful for drawing lines other than straight ones.

The colour of lines can easily be changed by pressing the Copy key. This results in all the available colours being displayed across the top of the screen from which a new

colour is selected.

The number of colours available for drawing will of course depend on the particular screen mode in use: you can select any of the three modes.

So it is possible to produce multicolour drawings in Mode 0, or if you prefer, high

Lancelot and the weirdos

SIR Lancelot, from Melbourne House, is a levels-and-ladders game in which you control the hero through a series of individual screen formats with a view to moving on to the next.

You start the game in Merlin's Magic Teleport Chamber, where you can choose from 23 alternative exits.

You move Sir Lancelot around the screen using either the joystick or the pre-selected keys.

The object of the exercise is to collect a variety of flashing objects ranging from skulls to keys, most of which are unidentifiable as anything in particular.

You must, however, collect all the items in order to return to the teleport chamber from where you can select your next screen. Each has its own routes by which all the items can be collected.

Once a route is found, the screen should pose no problems as the same route can be used on future visits.

It all sounds very easy, but, as with most of this type of game, there are numerous weirdos and meanies out to get you.

These come in many shapes and sizes, ranging from tanks to bouncing balls, from ghosts to mutant cauldrons, complete with stirring ladle.

Needless to say, contact with any one of them means curtains with one life down, three to go.

The screen is also on a time limit – so running out of time has the same effect.

Your resultant spirit ascends gracefully to the top of the screen accompanied by a chorus of the Death March – I wish they'd find something more original.

Once the program has loaded, you are greeted with a spectacular, colourful opening screen accompanied by J.S. Bach's Whatsit in D, opus 99

Should you fail to interrupt within 30 seconds, the screen scrolls tediously through the high score table and enters a demonstration mode.

The latter's a nice idea, giving you a brief insight into what is to come, but the high score table is painfully slow.

The game is extremely addictive and excellent use is made of sound and colour.

Full advantage has been taken of the Amstrad's graphic capabilities and particularly entertaining was Sir Lancelot's animated legs and sword-arm.

David Andrews

resolution two colour drawings in Mode 2.

In addition to simple line drawing, there are a number of other routines for drawing shapes such as circles, ellipses and boxes.

These may be varied in size, and moved around the screen with the cursor keys.

A fast fill option allows enclosed shapes to be filled in any colour.

Text may also be added to your drawings – useful for labelling diagrams or simply adding your name to the finished masterpiece.

An unusual feature which I particularly liked was the ability to scroll the screen in any direction. Using this facillty, you can easily move your drawing to any position on the screen.

Finished pictures may be saved to cassette or disc, and later retrieved for viewing or alteration.

The instructions also give some examples of saving sections of a screen for use in your own programs.

There are of course quite a lot of keys to use. The instruction manual details the use of all the routines.

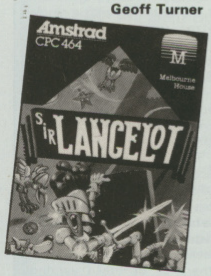
In addition, the program features a very useful Help option.

This provides on-screen information if you get stuck and works without disturbing the drawing you're working on so can be called at any time.

Overall, I found the package interesting and easy to use.

If you have a serious interest in art, or just simply enjoy doodling, you should find this an interesting program.

Geoff Turner



Stimulating simulation

GOING "solo" on one's first computer-simulated flying lesson is normal procedure, but "flying blind" at the same time is something else.

Of course I crashed when attempting it, but what the heck – there was plenty more aircraft where that came from.

I flew blind because Myrddin Software's **Flight Simulation** had such an irritating view through the cockpit windscreen.

There wasn't a proper sky or landscape, simply an expanse of yellow background in the upper half of the screen.

It resembled a desert dust-storm and may well have explained why a pyramid featured among the match-stick-style, 3D landmarks.

These materialised at first as a web of mysterious black lines, plus a single red one, which pulsated like an intensive-care heart monitor gone mad.

It was having the same effect on me, so I utilised the screen's static charge to hold a paper shield in place and concentrated on the instrument panel below.

Oddly enough, this procedure enabled me to become familiar with the various dials and meters rather quickly, so I resumed visual flying with far less irritation.

The instruments adequately represented those of a simple light aircraft and consisted of fairly crude dials with moving needles, which were backed up by a digital readout where necessary.

The sound denotes only piston-engined aircraft and might improve with stereo output, though the revs increased authentically when I opened up the throttle – even warning me of impending doom when I nosedived while looking at the chart!

I went smack into the middle of the Amstrad logo landmark (apologies to Alan Sugar), but still couldn't read the darned thing!

There are 15 levels, or types of aircraft, known as Dingbats,

which range in speed up to 500 knots.

Higher levels showed a faster, sometimes too severe response for me, so I settled on Dingbat 3 in which to make my first successful landing.

The latter accomplishment takes nerves of steel as well as skill and I admit to flunking a couple of touchdowns when my nerve gave out.

There is also a cheat device (the exclamation mark), which enabled me to reverse course for repeated landing practice and I made full use of it.

When I finally rolled to a stop, there was a moment or two of sheer disbelief, a further pause while my heartbeats quietened down, followed by an irresistible urge to climb out of the cockpit and emulate the Pope.

This program has much criteria for success, such as the need for fast reflex action, observation, forethought and planning.

It rewards those who persevere, but initially it can be disappointing – especially at the rather high purchase price of £11.95 – and I do wish it would let me "loop the loop".

Ray Miller-Lawson

It's wild out West

"HOWDY, Pardner" – that's how you'll be greeted by the introduction to **The Wild Bunch,** a text adventure from Firebird Software.

As a result of some unfortunate timing, you have been wrongfully accused of murder and your only chance to clear your name and escape a long prison sentence is to catch up with the real murderer – a member of the dreaded "Wild Bunch"!

You must evade the agent sent to bring you in and pursue your quarry by gambling, fighting and bribing your way from town to town.

With only $50 to your name, you have to purchase the necessary equipment to allow you to move around. And, with a horse costing $55, you need to get additional money if you don't want to do an awful lot of walking.

The best source of income seems to be from gambling in the saloons. This can be very profitable if your nerves are up to it.

Another form of income is the bounties on wanted men, and if you're quick on the draw in a gunfight and you've got the right man you can earn some ready cash.

It's a good idea to check your ammunition before challenging anyone to a gunfight 'cos them thar baddies don't miss.

Moving around can be a

risky business, too. All manner of nasties lurk in the deserts and prairies intent on doing you no good at all.

In a moment of desperation I tried to bribe a vulture with my last $7 but it didn't get me anywhere.

As you fight your way around the old West, your strength is gradually reduced – so be careful how you use it, because when it's all gone you're dead.

A few beers in one of the saloons will help keep your strength up, but be careful how many you have because too many can be disastrous.

The gunfights are fought graphically and there are sound effects to accompany both the gunfights and the text.

The on-screen instructions are quite adequate, although I found that a pencil and paper were invaluable for keeping track of who is who and where they all are.

You should find this an extremely entertaining game, which is easy to play.

Although I haven't yet caught up with a real murderer, I had a lot of fun trying.

Firebird software is very reasonably priced, so good hunting!

Allan Mitchell

Powerful filing program

THE very impressive capabilities of the Amstrad computer are brilliantly matched by **Masterfile,** Campbell Software's high-class and exceptionally powerful database program, which incorporates several features not normally found on similar applications.

The kind of information which can be used with this program is virtually limitless. Telephone numbers and addresses of friends, customers, business contacts and so on are handled with ease.

Setting up files is not a particularly simple task, but the 36 pages of information accompanying the program are very detailed, and will guide the novice user through the early stages.

Once set up, they are interrogated fairly easily, with facilities to sort numerically or alphabetically, in ascending or descending order of fields.

There is also a powerful search facility in which entries which conform to a stated criterion will be displayed.

Various criteria may be related together, and the arguments available include numerical comparisons such as equal and greater than, as well as more direct arguments such as present or absent.

Thus a search may identify quite complex subsets.

The most exciting feature of the program, however, is a relational aspect, previously not seen on home micros.

Suppose a file has been set up containing the correspondence received and sent by a club secretary. There will be several letters and other items sent to and from certain organisations.

To type in the full name and address each time, along with other information, would be a daunting task.

Masterfile allows a "parent" file to be set up, containing this central information, while those other entries which relate to it can be termed "child" records.

The interrogation of the child files can read those entries relevant to it from the parent file, so that full details can be pulled together.

In some, more complex, databases the idea is extended to grandparent and grandchild files.

These are not used here, although the theme is continued in the terminology, by searching for "childless parents" or "orphans". All a little strange to begin with!

The best databases I have used have read directly from discs, so that the eventual size of a file was only limited by the size of the disc.

Masterfile has a very generous 32k available, and most applications will not find the memory restraints a problem.

The program is a welcome indication of the arrival of excellent serious-applications software for the Amstrad.

Phil Tayler

REVIEWED SO FAR

Adventure Quest	Level 9
American Football	APS
Amstradraw	B.S. McAlley
Android One	Vortex
Arnold goes to somewhere else	Nemesis
Artwork	Kuma
Astro Attack	Amsoft
Amstrad Tutorial	Amsoft
Atomsmasher	Romik
Blagger	Alligata
Bridge Player	C.P.
Centre Court	Epicsoft
Chopper Squad	Interceptor
Colossal Adventure	Level 9
Country Cottages	Sterling
Crystal Theft	Wiccasoft
Cubit	Mr Micro
Defend or Die	Alligata
Detective	Argus Press
Dragon's Gold	Romik
Ev*bert	Microbyte
Fighter Pilot	Digital Integration
Flexi-friend	Camel
Flight Path 737	Anirog
Flight Simulation	Myrrdin
Football Manager	Addictive
Fruit Machine	Amsoft
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
Home Budget	Kuma
Hunchback	Ocean
Jet Boot Jack	English
Lords of Time	Level 9
Manic Miner	Software Projects
Map Rally	B.E.S.
Masterfile	Campbell
Message from Andromeda	Interceptor
Mr Wong's Loopy Laundry	Artic Computing
Music Composer	Kuma
Mutant Monty	Artic Computing
Number Painter	ASK
Osprey	B.E.S.
Pinball Wizard	C.P.
Punchy	Mr Micro
Pyjamarama	Microgen
Roland in the Caves	Amsoft
Roland Goes Square Bashing	Durell
Rollaball	Timeslip
Royal Quest	Timeslip
Screen Designer	DJL
Sir Lancelot	Melbourne House
Snooker	CDS
Snooker	Gem
Snowball	Level 9
Software Star	Addictive
Sorcery	Virgin
Splat	Incentive
Star Avenger	Kuma
Star Watcher	Amsoft
Stockmarket	Argos Press
Strip Poker	Knightsoft
Survivor	Anirog
Test Match	CRL
The Moors Challenge	Timeslip
The Wild Bunch	Firebird
Timeman One	B.E.S.
Timeman Two	B.E.S.
Trial of Arnold Blackwood	Nemesis
War Zone	C.C.S.
Wordhand	B.E.S.
Wordwise	B.E.S.
World Cup Football	Artic
Xanagrams	Postern
Z80 Assembler, Disassembler and Editor	Arnor

A fling on the market

HAVE you ever fancied trying your luck with a fling on shares but lacked the money or courage to try? If that is the case **Stockmarket** from Argus Press will suit you well, being an excellent implementation of dealings through a broker's office, with superbly drawn graphics.

These represent a telex terminal and modem, complete with On-line indicators. The dealings are made through this modem, which is of course hooked up to a monitor, the screen being displayed clearly in Mode 1.

There are four companies whose share values are included in the simulation, these producing lead, zinc, tin and gold. The initial values are set to approximately half their maximum value, so quick profits (or losses) may be made.

The players – up to six are allowed for – each have a turn at the terminal, with a single letter response deciding whether shares are to be bought or sold, or which company is being dealt with.

Numerical values are required of course, but the screen questions are clear and unambiguous.

The share values fluctuate according to the buying and selling activities of the players, together with random features. In addition newsflashes appear via the telex terminal, the benevolence or otherwise of these varying

dramatically with the difficulty level selected.

At the easiest level tax refunds abound, while capital gains tax crops up with depressing and monotonous frequency at the hardest! The implementation of the telex terminal is superb.

Feminists will no doubt object that all references are to "Sir", but again I would imagine that this is fairly true to reality. However this minor criticism aside, even ardent feminists will enjoy the excellent key response, quick and precise and the clear screen layout.

I tried my hardest to crash the program, but couldn't manage it – it appears to be very comprehensively error-trapped

I recommend this game to devotees of Monopoly and similar pursuits. It is – in short – the best version I have seen so far!

Pat Hillery

A pulse-pounding zapper

INSTEAD of having you stare at a blank screen while the cassette loads, **Android One** from Vortex, includes a screen dump in the Basic header showing what is in store – a massive hole knocked through a brick wall!

In this game you are an Android encouraged to rush headlong through a succession of mazes and walls with the help of a powerful laser gun. You must save the world by reaching a reactor in time to stop it from exploding.

Your path lies past a frightening array of other creatures.

A multi-stage game, each of the 14 screens has its own strategy for success.

That isn't to say I've actually discovered the best tactics for each screen, but at last I'm getting through to the later ones. I must say that I'd

certainly welcome an opportunity to select the starting screen, as going back to square one each game leaves me little chance of spending much time on the higher levels.

I'm sure a level choice, or a practice mode, would be appreciated by many players.

The movement of the Android and the various meanies – Bouncers, Groupies, Skaters and Wanderers – is beautifully smooth, and you can choose keyboard or joystick control.

The keys selected were quite good, although a joystick is a must for serious contenders for the hi-score table.

Completing each screen allows a very smooth horizontal scrolling on to the next, and it is also possible to move back into a previous room.

The sound effects are very satisfactory, raising the adrenalin levels of the players most effectively.

All in all, it is a very competent and addictive game, with a fairly original storyline. If you enjoy a good zap, you could do a lot worse than Android One.

Phil Tayler

ALAN McLACHLAN turns his attention to DATA statements

EVENIN' all, tame Plod here. My hints and tips over the last couple of months seem to be working because the number of your letters about listing problems has dwindled quite dramatically. Either that or the postman's dumping them in the river.

Anyway, let's get straight into some more useful advice to get those programs running.

I suspect the type of program most feared by beginners is one that consists of lines and lines of data statements.

I must admit that lots of my first programs would have been greatly improved with the use of this command, but I never really took the trouble to get the hang of it.

Although READ and DATA are great commands for reducing the amount of typing that faces a programmer, they can cause havoc to the poor chap trying to fathom out what's going on.

Let's demonstrate this with a simple program.

Type in Program I and run it.

```
10 REM PROGRAM I
20 CLS
30 FOR X= 1 TO 5
40 READ A,B
50 LOCATE A,B:PRINT CHR$(143)
60 NEXT
70 DATA 10,6,2,20,5,15,1,23,36,8
80 GOTO 80
```

Program I

Lines 30 to 60 create a loop which is executed five times.

Within this loop DATA in line 70 is READ in, in pairs, allocated to the variables A and B, and used to print a series of blocks – CHR$(143) – on the screen.

Line 80 is just a dummy line to suppress the "Ready" prompt.

I must admit it isn't exactly the most spectacular piece of programming you're likely to come across, but it's only meant to be a simple demonstration. Program II, which is longer, produces identical results.

```
10 REM PROGRAM II
20 CLS
30 LOCATE 10,6:PRINT CHR$(143)
40 LOCATE 2,20:PRINT CHR$(143)
50 LOCATE 5,15:PRINT CHR$(143)
60 LOCATE 1,23:PRINT CHR$(143)
70 LOCATE 36,8:PRINT CHR$(143)
80 GOTO 80
```

Program II

Granted the two programs have the same number of lines, but look at the amount of typing saved by using the first.

For the purpose of this article, though, the programs' actual effects matter less than the error messages produced by our first version should anything go wrong.

Let's assume you've accidentally mistyped a full-stop for a comma in the list of data in line 70.

That is:

```
70 DATA 10,6,2,20,5,15.1,23,36,8
```

It's easily done, particularly if you have already typed in more than 100 lines of data. And it's even more likely when you know you've 100 more of the things to come.

Line 40 is attempting to READ five pairs of numbers – 10 numbers in all. But it's only encountering four and a half pairs – 9 numbers.

What should have been a "read" of 15 followed by a "read" of 1 (2 numbers), is now being read as 15.1, one number with a decimal point.

Consequently your micro reports "DATA exhausted in 40". ("PROGRAMMER exhausted in dining room" might be more appropriate.)

Anyway, that message means that line 40 has run out of DATA to READ. But if you list line 40 you'll find it apparently in order – and it is.

The message therefore is not as helpful as it first appears, is it?

The real error lies in the DATA line some distance away, though it's reported as being in the READ line.

Right, so we know where the mistake is, but here we've only one line of data.

What if you had in fact 100 lines of data? The error would be very difficult to find and you'd need eyes like a hawk to spot that full-stop.

A useful method of finding an error in lists of data is to PRINT the data immediately after it has been READ, but before the program has had an opportunity to do anything with it.

Alter Program I so that it contains the data error in line 70, and REM line 50 to preserve it from harm while we're debugging (last month's tip). Next insert lines 45 and 46:

```
45 PRINT A,B
46 WHILE INKEY$="":WEND
```

Your modified Program I should now be identical to Program III:

```
10 REM PROGRAM III
20 CLS
30 FOR X= 1 TO 5
40 READ A,B
45 PRINT A,B
46 WHILE INKEY$="":WEND
50 REM LOCATE A,B:PRINT CHR$(143)
60 NEXT
70 DATA 10,6,2,20,5,15.1,23,36,8
80 GOTO 80
```

Program III

Now Run the program. Your data should appear as pairs of numbers needing a key press to display each successive pair. The numbers will appear on screen as in Figure I.

```
 10          6
  2          20
  5          15,1
 23          36
DATA exhausted in 30
```

Figure I

The black sheep stands out like a sore thumb. This tip is invaluable when there is a large amount of data to check.

You'll have even greater problems if you accidentally omit a data item and leave the comma in the right place. Change line 70 of Program III to:

```
70 DATA 10,6,2,20,5,15,,23,36,8
```

Notice the difference? There's no "DATA exhausted" message.

This is because there's enough information there, believe it or not. But it's the wrong information.

The Amstrad finds the two consecutive commas, says there's nothing there, and puts exactly that into the variable A – a zero.

In fact unless you are trying to use LOCATE you won't get any error message at all. But you'll almost certainly get some strange results that would be very difficult to track down.

The reason I mentioned LOCATE is easily demonstrated. Remove the REM from line 50, delete lines 45 and 46, then alter line 70 to:

```
70 DATA 10,6,2,20,5,,,23,36,8
```

or

```
70 DATA 10,6,2,20,5,0,0,23,36,8
```

Now run it again and see how the error message suddenly appears as "Improper argument in 40".

However listing line 40 reveals our old friend the READ statement, which, as we know, is absolutely correct. (This reminds me of those letters – "I've typed the line in over and over again so there must be an error in the magazine").

Once again the error is actually in the data in line 70. Your micro has done the dirty on you when it reports the error, but the mistake is definitely yours.

Here you are attempting to LOCATE a text screen position that doesn't exist – 0,0. Remember the top left corner is location 1,1.

The whole point of these examples is to demonstrate that there are lots of error messages that specify a correctly entered line, when in fact if you can follow your programming logic, that line will point to another, which is in error.

I told you in my May article that when the Amstrad encounters a Syntax error, it automatically lists the error line in edit mode.

This is the only error-type it does this for, but it would be quite handy to create a routine that will at least list that line for you.

To do this you use the ON ERROR . . . GOTO command, and the variable ERL (the error line). If you type in Program IV you'll see the sort of thing you can do. It's only meant to be a bit of fun, but someone may find it quite useful.

```
10 REM PROGRAM IV
20 CLS
30 ON ERROR GOTO 100
40 FOR X= 1 TO 4
50 READ A
60 NEXT
70 DATA 1,2,3
80 END
100 KEY 130,"LIST"+STR$(ERL)+CHR$(13)
110 KEY 129,"GOTO"+STR$(ERL)+CHR$(13)
120 PRINT"What a wally! you've made a
n error"
130 PRINT"Press key 1 for the error":
PRINT"Press key 2 to list the line"
140 END
```

Program IV

Here's how it works.

Line 30 suppresses the normal error routine so that I can create my own from line 100.

Lines 40-70 generate the same error as the one we've looked at already in Program I – DATA exhausted.

Line 80 is the normal end of the program. Bear in mind it may not be in this form if you are adding this routine to a finished program.

It could be an infinite loop using a WHILE . . . WEND or FOR . . . NEXT command.

Line 100 sets up the 2 key on the numeric key pad to LIST the line number.

The STR$ command is used to change what is normally a numeric variable into a string for use with the KEY command.

Line 110 sets up the number 1 key to go back to the error line and generate the previously supressed error message.

This may seem long-winded, but it's the only way I've found to get it printed on the screen.

Granted the variable ERR (error number) could be used in with ERL in line 100. But it only produces a number that you would still have to look up in the manual.

Line 120 prints on screen any initial error message you may care to make up.

Line 140 makes sure that having jumped to this routine, the program has an END. The error message generated if you omit this is rather complicated and will be the subject of a future article.

You can add line 30 to the beginning, and lines 100-140 to the end, of any program, provided of course the writer has not already incorporated some crafty error handling routine of his own.

Well I think you have enough to be going on with.

If the number of programming queries keeps dropping at the rate it has been, I might even be out of a job.

Still it would be an end to being up all night answering other people's queries.

Now I'll be up all night answering my own.

Haystack Peripherals CPC464 Software Haystack Peripherals

8 MIDGROVE, DELPH, OLDHAM OL3 5EJ Tel: 045 77 5229

	RRP	OUR PRICE
A'n'F		
Chuckie Egg	7.90	6.90
ABACUS		
Cash Planner	12.95	11.45
Mailing List	17.95	15.95
Non Vat Accounts	17.95	15.95
Payroll	29.95	26.95
Purchase/Sales Ledger	29.95	26.95
Stock Control	17.95	15.95
ADDICTIVE		
Football Manager	7.95	6.95
Software Star	7.95	6.95
ALLIGATA		
Blagger	7.95	6.95
Defend or Die	7.95	6.95
ACTIVISION		
Ghostbusters	10.99	9.99
AMSOFT		
American Football	9.95	8.95
Amsgolf	8.95	7.95
Amsword (Advanced)	19.95	17.95
Amsword (Easy)	9.95	8.95
Astro Attack	8.95	7.95
Centrecourt	8.95	7.95
Concise Basic Spec.	19.95	17.95
Concise Firmware Spec.	19.95	17.95
Cubit	8.95	7.95
DFM Database*	14.95	12.95
Decision Maker	21.95	19.95
Devpac Ass/Disass	24.95	21.95
Entrepreneur	21.95	19.95
Guide to Basic I	19.95	17.95
Guide to Basic II	19.95	17.95
Hisoft DEVPAC*	34.95	29.95
Hisoft PASCAL	34.95	29.95
Home Accounts Manager*	14.95	12.95
Invostat*	29.95	26.95
Jammin*	8.95	7.95
Jet Boot Jack	8.95	7.95
Master Chess	8.95	7.95
Masterfile	24.95	22.95
Mr Wongs Loopy Laundry	8.95	7.95
Pitman Typing Tutor	9.25	8.25
Project Planner*	24.95	22.95
Punchy	8.95	7.95
Roland/Caves	8.95	7.95
Roland/Sq. Bashing	8.95	7.95
Screen Designer	14.95	12.95
Splat!	8.95	7.95
Spreadsheet*	19.95	17.95
Starwatcher	17.45	15.45
Stock Aid*	29.95	26.95
Stockmarket*	8.95	7.95
Transact*	29.95	26.95
Xanagrams	8.95	7.95
AMSOFT/BES		
Animal, Vegetable, Mineral	8.95	7.95
Happy Letters	8.95	7.95
Happy Numbers	8.95	7.95
Happy Writing	8.95	7.95
Map Rally	8.95	7.95
Osprey!	9.95	8.95
Timeman One	8.95	7.95
Timeman Two	8.95	7.95
Wordhang	8.95	7.95
Worldwise	8.95	7.95
ANIROG		
Flightpath 737	6.95	5.95
Moon Buggy	7.95	6.95
Survivor	6.95	5.95
ARIOLASOFT		
Archon	10.95	9.95
Hard Hat Mack*	8.95	7.95
One On One*	8.95	7.95
ARTIC		
World Cup Football	7.95	6.95
ASK		
Number Painter	8.95	7.95
BLACK KNIGHT		
Timebomb	7.65	6.65
CCS		
Warzone	6.95	5.95

	RRP	OUR PRICE
CDS		
Castle Blackstar*	6.95	5.95
Steve Davis Snooker	7.95	6.95
C.P. SOFTWARE		
Backgammon*	8.95	7.95
Bridge Player	9.95	8.95
Draughts*	8.95	7.95
Pinball Wizard	8.95	7.95
Superchess	9.95	8.95
CRL		
Rocky Horror Show*	8.95	7.95
Test Match	6.95	5.95
CAMEL		
FlexiFrend	7.50	6.50
Grasp	8.50	7.50
DESIGN DESIGN		
Dark Star	7.95	6.95
Tank Busters	7.95	6.95
DIGITAL INTEGRATION		
Fighter Pilot	8.95	7.95
DK'Tronics		
Minder*	9.95	8.95
DOMARK		
A View to a Kill*	10.99	9.99
DURELL		
Combat Lynx	8.95	7.95
Death Pit	7.95	6.95
GARGOYLE		
Dun Darach*	9.95	8.95
GEMINI		
Database	19.95	17.95
Home Accounts	19.95	17.95
Report Generator	19.95	17.95
GILSOFT		
The Quill	16.95	15.25
HEWSON		
Fantasia Diamond	7.95	6.95
Heathrow ATC	7.95	6.95
Zaxp	14.95	13.45
HISOFT		
Font 464*	7.95	6.95
IMAGINE		
World Series Baseball*	8.95	7.95
INCENTIVE		
Confuzion	6.95	6.25
Millionaire	6.95	5.95
INTERCEPTOR		
Azimuth 3000	8.95	7.95
Chopper Squad	6.00	5.25
Forest at Worlds End	6.00	5.25
Heroes of Karn	6.00	5.25
Jewels of Babylon	6.00	5.25
Message from Andromeda	6.00	5.25
Warlord*	6.00	5.25
KNIGHTSOFT		
Animated Strip Poker	8.95	7.95
KUMA		
Artwork	9.95	8.95
Database	14.95	12.95
Easi VAT*	39.50	34.50
Forth	19.95	17.95
Fruity Frank	6.95	5.95
Galaxia	5.95	5.25
Gems of Stradus	7.95	6.95
Holdfast	5.95	5.25
Home Budget	19.95	17.95
Music Composer	9.95	8.95
Star Avenger	6.95	5.95
LEVEL 9		
Adventure Quest	9.95	8.45
Colossal Adventure	9.95	8.45
Dungeon Adventure	9.95	8.45
Emerald Isle	6.95	5.95
Lords of Time	9.95	8.45
Return to Eden	9.95	8.45
Snowball	9.95	8.45
LOTHLORIEN (WARMASTER)		
Johnny Reb	6.95	5.95
Redcoats	6.95	5.95
Special Operations	6.95	5.95

	RRP	OUR PRICE
MARTECH		
Brian Jack's Superstar Chall.*	7.95	6.95
MELBOURNE HOUSE		
Hobbit	14.95	12.95
Sir Lancelot	5.95	5.45
MICROGEN		
Everyone's a Wally*	9.95	8.95
Herberts Dummy Run*	9.95	8.95
Pyjamarama	8.95	7.95
MICROPOWER		
Ghouls	6.95	6.15
Killer Gorilla/Gauntlet	9.95	8.95
MIRRORSOFT		
Caesar's Travels*	7.95	6.95
First Steps/Mr. Men	8.95	7.95
Here + There/Mr. Men	7.95	6.95
MOSAIC		
Erik The Viking	9.95	8.95
MYRRDIN		
Flight Simulation	11.95	10.45
NEW GENERATION		
Machine Code Tutor*	14.95	13.45
OCEAN		
Daley Thompson's Decathlon	8.95	7.95
Gilligans Gold*	6.90	5.90
Hunchback II	8.95	7.95
Kong Strikes Back	8.95	7.95
ODIN		
Nodes of Yesod*	9.95	8.95
P.S.S.		
Battle for Midway	9.95	8.95
REALTIME		
3D Starstrike	6.95	5.95
REDSHIFT		
The Tripods	11.50	9.95
SHARDS		
Mystery/Java Star	7.95	6.95
SOFTWARE PROJECTS		
Jet Set Willy	8.95	7.95
Manic Miner	8.95	7.95
TASMAN		
TASCOPY	9.90	8.90
TASPRINT	9.90	8.90
TASWORD	19.95	17.95
TYNESOFT		
Supergran*	9.95	8.95
ULTIMATE		
Alien 8*	9.95	8.95
Knightlore	9.95	8.95
U.S. GOLD		
Beach Head*	9.95	8.95
VIRGIN		
Sorcery	8.95	7.95
VORTEX		
Android One	7.95	6.95
Android Two*	7.95	6.95
Highway Encounter*	8.95	7.95
Tornado Low Level*	7.95	6.95
WINTERSOFT		
Ring of Darkness	9.95	8.95

*** Please ring to confirm availability**

C15 Computer Cassettes (Box of 10) **4.50**

Pro-Ace Joystick **11.95**

PHONE FOR DETAILS OF OTHER SOFTWARE – WE CAN SUPPLY ALMOST ANY AMSTRAD TITLE AVAILABLE!

Overseas orders – Please add £1.00 per item. Send sterling money orders

BEWARE OF CHEAP BOOTLEG SOFTWARE – ALL OUR TITLES ARE ORIGINALS!

Please Send:	Price
TOTAL	

ALL OUR PRICES INCLUDE VAT AND CARRIAGE

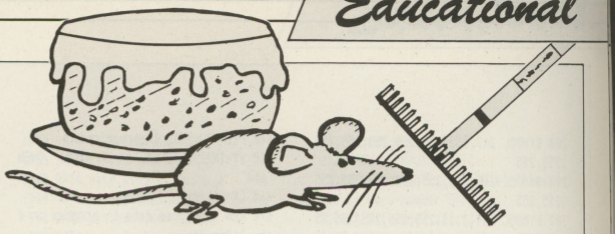
Cheques/PO's to:
HAYSTACK PERIPHERALS
8 Midgrove, Delph, Oldham OL3 5EJ
Tel: Saddleworth (04577) 5229

NAME .....

ADDRESS .....

.....

.....



Jumbo's word game

By STEVE LUCAS

ONE of the first concepts young children are introduced to when learning to read is phonics, the ability to recognise words which sound alike.

This program presents the children with four words, three of which have a similar ending. The children's task is to find the 'Odd Man Out'.

A mouse will move underneath boxes containing the four words and the child must press the space bar when the mouse is underneath the word it wishes to select.

In the meantime Jumbo the elephant is busy eating his buns at the bottom of the screen and the selection must be made before he has eaten them all.

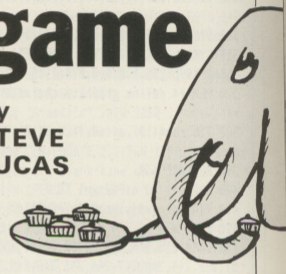
Each child is given three lives and will lose one life if he is too slow or gets the wrong answer. Each question answered correctly causes the mouse to move faster.

When the program is run a menu is displayed which gives you a choice of three options.

- Use the data contained within the program.
- Write a data file to tape containing your own questions.
- Read in a tape of questions.

There are 60 questions in the game and these are chosen at random. In the unlikely event of a child completing them all the pointers are reset and the questions are set again in a different order and at a higher speed.

If you don't want the mouse to speed up when a child gets an answer right you should delete line 1130.



10-80	Initialise variables and select colours.
100-360	Define characters.
380-440	Define graphics.
460-570	Titles.
590-680	Menu.
700-750	Define windows.
760-1080	Main loop.
1100-1210	Get question right.
1220-1280	Too slow.
1290-1370	Wrong answer.
1380-1430	End of game.
1440-1450	Reset pointers.
1460-1500	Play tune.
1510-2190	Use data within program.
2210-2290	Load data file.
2300-2550	Create data file.
2560	Instructions.

VARIABLES

guess$	Your guess.
s%	Lives left.
a$	Keyboard input.
y	Number of question.
aa%(x)	Number of answer.
a$(x,y)	Words.
d%(x)	Questions already set.
mouse$,bun$,row$	Graphics.
bun%,bun%1	Number of buns.
ps%	Position of mouse.
st%	Speed.
jump%	Move mouse.
C%	Option in menu.

```
10 REM ** Jumbo's Word Game **
20 REM (c) Computing with the Amstrad
30 REM ** Steve Lucas **
40 REM ** define the colours used **
50 MODE 1:INK 0,1:INK 1,24:INK 2,20:IN
K 3,6
60 s%=3:REM ** set the number of lives
 allowed **
70 bun%=20:bun1%=bun%
80 st%=300:ps%=5:t%=0
90 REM ** define the character set
100 SYMBOL AFTER 229
110 SYMBOL 230,0,0,0,216,244,245,36,32
120 SYMBOL 231,0,0,7,159,191,127,51,16
130 SYMBOL 232,0,0,0,0,0,28,62,127
140 SYMBOL 233,128,128,0,0,0,0,0,0
150 SYMBOL 234,127,127,0,0,0,0,0,0
160 SYMBOL 235,252,252,0,0,0,0,0,0
170 SYMBOL 236,1,1,0,0,0,0,0,0
180 SYMBOL 237,196,132,132,128,128,128
,128,128
190 SYMBOL 238,31,31,31,31,31,31,31,31
200 SYMBOL 239,252,252,252,252,252,252
,252,252
210 SYMBOL 240,96,96,96,96,0,0,0,0
220 SYMBOL 241,240,240,240,252,252,204
,236,196
230 SYMBOL 242,247,3,1,0,0,0,0,0
240 SYMBOL 243,123,120,120,112,112,112
,96,96
250 SYMBOL 244,240,240,240,240,240,240
,240,240
260 SYMBOL 245,63,31,31,31,31,31,63,12
7
270 SYMBOL 246,0,0,128,224,244,224,244
,240
```

```
280 SYMBOL 247,252,255,255,255,255,255,255
,255,255
290 SYMBOL 248,255,255,255,255,255,255,255
,255,255
300 SYMBOL 249,3,15,255,255,255,255,255,25
5,255
310 SYMBOL 250,248,255,127,191,191,223
,255,255
320 SYMBOL 251,253,254,254,159,159,255
,255,255
330 SYMBOL 252,31,63,63,63,63,63,63,12
7
340 SYMBOL 253,0,0,0,0,0,0,0,240,240
350 SYMBOL 254,0,0,0,0,0,0,0,251,251
360 SYMBOL 255,0,0,0,0,0,0,0,7,15
370 REM ** define graphics characters
**
380 DIM row$(10),a$(60,4),aa%(60),d%(6
0)
390 RESTORE 430
400 FOR y=1 TO 6:FOR x=1 TO 7
410 READ d:row$(y)=row$(y)+CHR$(d)
420 NEXT x,y
430 DATA 255,254,253,32,32,32,252,2
51,250,249,249,247,246,245,248,248,248
,248,248,244,243,242,248,248,248,248,2
41,240,32,239,32,32,238,32,32,236,235,
32,32,234,233,32
440 bun$=CHR$(232):mouse$=CHR$(231)+CH
R$(230)
450 REM ** titles **
460 CLS:FOR x=1 TO 6
470 LOCATE 1,x:PEN 2:PRINT row$(x)
480 LOCATE 31,x+19:PRINT row$(x);
490 LOCATE 31,x:PEN 1:PRINT row$(x)
500 LOCATE 1,x+19:PRINT row$(x);
510 NEXT
520 WINDOW #1,10,30,10,15:PAPER#1,1:PE
N #1,0
530 CLS#1:LOCATE #1,2,2:PRINT#1,"Jumbo
's Word Games"
540 PEN #1,3:LOCATE #1,4,4:PRINT#1,"by
  Steve Lucas"
550 REM ** play music and introduce a
short delay **
560 GOSUB 1470:REM ** music
570 FOR x= 1 TO 3000:NEXT x
580 REM ** choose data held in program
 or data file **
590 WINDOW #2,2,39,8,17:PAPER #2,1:PEN
 #2,3:CLS#2
600 LOCATE #2,1,2:PRINT#2,"Which of th
e following do you want ?"
610 LOCATE #2,1,4:PEN #2,0: PRINT#2,"1
). To use the data in the program."
620 PRINT#2,"2). To load data from tape."
630 PRINT#2,"3). To create a data file."
640 PRINT#2,"4). Instructions."
650 a$=INKEY$:IF a$)"4" OR a$<"1" THEN
 650
660 C%=VAL(A$)
670 REM ** choose data in program or d
ata files **
680 ON C% GOSUB 1520,2210,2310,2570
690 REM ** define windows **
700 WINDOW #3,1,18,1,5:PAPER #3,2:PEN
#3,3
710 WINDOW #4,11,28,1,5:PAPER #4,1:PEN
 #4,0
720 WINDOW #5,21,38,1,5:PAPER #5,2:PEN
 #5,3
730 WINDOW #6,31,48,1,5:PAPER #6,1:PEN
 #6,0
740 WINDOW #7,1,48,6,9:PAPER #7,3:PEN
#7,2
750 WINDOW #2,1,48,18,25:PAPER #2,2:PE
N #2,0
760 WHILE s%>0
770 CLS:CLS#3:CLS#4:CLS#5:CLS#6:CLS#7:
CLS #2
780 REM ** random question **
790 y=INT(RND(1)*60)+1
800 IF d%(y)>0 THEN 790
810 d%(y)=1:IF t%>59 THEN GOSUB 1440 :
REM ** all the questions have been set
 **
820 IF s%<1 THEN GOSUB 1380:RUN
830 REM ** set question **
840 FOR x=1 TO 4
850 LOCATE #x+2,4,3:PRINT#x+2,a$(y,x)
860 NEXT
870 PEN #2,3:FOR x=1 TO 6
880 LOCATE #2,31,18+x:PRINT #2,row$(x)
890 NEXT x
900 PEN #2,3:LOCATE #2,18,8:IF s%>1 TH
EN PRINT#2,"You have ";s%;" lives left
" ELSE PRINT#2,"You have 1 life left";
SPC(5);
910 jump%=st%
920 LOCATE #7,ps%,2:PRINT #7,mouse$
930 LOCATE #2,1,16:PRINT#2,STRING$(38-
bun%,CHR$(32));STRING$(bun%,bun$);
940 WHILE bun%>0
950 WHILE jump%>0
960 jump%=jump%-1
970 guess$=INKEY$
980 IF guess$<>"" THEN GOSUB 1100:t%=t
%+1:CLS #3:CLS #4:CLS #5:CLS #6:GOTO 7
90
990 WEND
1000 jump%=st%
1010 LOCATE #7,ps%,2:PEN #7,3:PRINT#7,
mouse$:PEN #7,2:SOUND 1,478
1020 ps%=ps%+10:IF ps%>35 THEN ps%=5
1030 LOCATE #7,ps%,2:PRINT #7,mouse$
1040 bun%=bun%-1
1050 LOCATE #2,1,16:PRINT#2,STRING$(38
-bun%,CHR$(32));STRING$(bun%,bun$);
1060 WEND
1070 GOSUB 1220
1080 WEND
1090 RUN
1100 IF ps%=5 THEN ans%=1 ELSE IF ps%=
15 THEN ans%=2 ELSE IF ps%=25 THEN ans
%=3 ELSE ans%=4
1110 IF ans%<>aa%(y) THEN GOSUB 1290:R
ETURN
1120 REM ** the next line makes the ne
xt game harder ... delete the line if
you don't want this **
1130 st%=st%-10
1140 GOSUB 1470
1150 CLS#7:LOCATE #7,4,3:PRINT #7,"Pre
ss the <Space Bar> to continue."
1160 PEN #2,0:LOCATE #2,15,4:PRINT#2,"
Well done!"
1170 a$=INKEY$:IF a$<>" " THEN 1170
1180 CLS#7:PEN #2,3:PEN #2,0:LOCATE #2
,15,4:PRINT#2,STRING$(10,CHR$(32));
1190 bun%=bun1%
1200 ps%=5
1210 RETURN
1220 PEN #2,0:LOCATE #2,18,4:PRINT#2,"
Too slow. The word was ";a$(y,aa%(y))
1230 CLS#7:LOCATE #7,4,3:PRINT #7,"Pre
ss the <Space Bar> to continue."
1240 s%=s%-1
1250 PEN #2,3:LOCATE #2,18,8:IF S%>1 T
HEN PRINT#2,"You have ";s%;" lives lef
t" ELSE PRINT #2,"You have 1 life left
";SPC(5);
1260 a$=INKEY$:IF a$<>" " THEN 1260
1270 bun%=bun1%
1280 RETURN
1290 PEN #2,0:LOCATE #2,18,4:PRINT#2,"
Wrong ! The word was ";a$(y,aa%(y))
1300 CLS#7:LOCATE #7,4,3:PRINT #7,"Pre
ss the <Space Bar> to continue."
1310 s%=s%-1:bun%=bun1%
1320 PEN #2,3:LOCATE #2,18,8: IF s%>1
THEN PRINT#2,"You have ";s%;" lives le
ft" ELSE IF s%=1 THEN PRINT#2,"You hav
e 1 life left";SPC(5); ELSE PRINT#2,"Y
ou have no lives left !";SPC(5)
1330 a$=INKEY$:IF a$<>" " THEN 1330
1340 CLS#7:PEN #2,3:PEN #2,0:LOCATE #2
,18,4:PRINT#2,STRING$(28,CHR$(32));
1350 CLS #3:CLS #4:CLS #5:CLS #6
1360 ps%=5
1370 RETURN
1380 CLS #2:FOR x=1 TO 6: LOCATE #2,1,
```

```
10+x:PRINT#2,row$(x):LOCATE #2,34,10+x
:PRINT#2,row$(x):NEXT
1390 FOR x=1 TO 500:SOUND 1,x,1:NEXT x
1400 LOCATE #2,5,2:PRINT#2,"You have l
ost all your lives !!!"
1410 CLS #7:LOCATE #7,4,2:PRINT#7,"Pre
ss the Space Bar for another game."
1420 d$=INKEY$:IF d$<>" " THEN 1420
1430 RETURN
1440 t%=0:FOR x=1 TO 60:d%(x)=0:NEXT x
1450 RETURN
1460 REM ** play tune **
1470 RESTORE 1490
1480 FOR x=1 TO 15:READ d:SOUND 1,d,20
,7:FOR y=1 TO 200:NEXT y,x
1490 DATA 478,478,426,478,379,358,478,
478,253,284,319,358,379,426,478
1500 RETURN
1510 REM ** use the data held within t
he program **
1520 RESTORE 1580:CLS#2:LOCATE #2,2,3:
PRINT#2,"Reading the data into the arr
ays now."
1530 FOR z=1 TO 60: FOR y=1 TO 4
1540 READ a$(z,y):NEXT y
1550 READ aa%(z):NEXT z
1560 RETURN
1570 REM ** data for 60 questions **
1580 DATA cat,bat,sat,let,4
1590 DATA eat,meat,rub,seat,3
1600 DATA cot,dot,not,set,4
1610 DATA hill,call,ball,fall,1
1620 DATA miss,put,nut,hut,1
1630 DATA hiss,miss,mat,kiss,3
1640 DATA cah,pan,tin,ran,3
1650 DATA ham,put,Sam,Pam,2
1660 DATA tap,map,lap,bat,4
1670 DATA hid,rid,put,Sid,3
1680 DATA had,rid,mad,sad,2
1690 DATA rod,cod,pod,rat,4
1700 DATA plate,bake,cake,rake,1
1710 DATA late,ball,hate,gate,2
1720 DATA Jill,Bill,call,pill,3
1730 DATA shell,well,pull,bell,3
1740 DATA pull,bull,hull,all,4
1750 DATA hop,shut,pop,top,2
1760 DATA shed,bed,red,hop,4
1770 DATA pop,lip,pip,rip,1
1780 DATA dad,bad,hop,lad,3
1790 DATA lad,hit,sit,pit,1
1800 DATA fan,sun,fun,bun,1
1810 DATA lid,hid,bun,kid,3
1820 DATA face,pace,plate,race,3
1830 DATA meat,heat,seat,sit,4
1840 DATA fair,hair,chair,chap,4
1850 DATA lid,bin,pin,tin,1
1860 DATA cut,fish,hut,put,2
1870 DATA take,lake,hake,hunt,4
1880 DATA car,far,hall,bar,3
1890 DATA wake,pin,rake,make,2
1900 DATA pan,pen,hen,men,1
1910 DATA ant,pant,men,rant,3
1920 DATA mast,last,fish,past,3
1930 DATA lock,tank,rock,sock,2
1940 DATA lamp,vest,ramp,camp,2
1950 DATA vest,rest,pest,lock,4
1960 DATA nine,mine,pine,kite,4
1970 DATA pram,ham,land,tram,3
1980 DATA land,sand,hand,lit,4
1990 DATA hop,pin,pop,top,2
2000 DATA mark,park,bark,land,4
2010 DATA hop,pin,top,pop,2
2020 DATA land,mark,dark,park,1
2030 DATA jet,pond,pet,let,2
2040 DATA ride,side,tide,land,4
2050 DATA pet,clay,say,play,1
2060 DATA can,pan,plug,ran,3
2070 DATA pad,mad,dad,cat,4
2080 DATA map,cat,sap,rap,2
2090 DATA late,hate,Kate,cat,4
2100 DATA mate,gate,cap,rate,3
2110 DATA car,far,bar,ball,4
2120 DATA cat,bike,like,Mike,1
2130 DATA late,plank,ate,plate,2
2140 DATA bank,pull,plank,rank,2
2150 DATA glad,shade,glade,blade,1
2160 DATA chop,cop,cat,shop,3
2170 DATA wait,wake,bake,take,1
2180 DATA lake,full,rake,make,2
2190 DATA shut,run,rut,nut,2
2200 REM ** to load a data file in fro
m tape or disc **
2210 CLS:LOCATE 5,2:PEN 1:PRINT"To loa
d a file in from tape."
2220 LOCATE 3,5:PEN 2:PRINT"Please ins
ert your data tape now."
2230 OPENIN "data"
2240 FOR z= 1 TO 60
2250 FOR y=1 TO 4
2260 INPUT #9,a$(z,y):NEXT y
2270 INPUT #9,aa%(z):NEXT z
2280 CLOSEIN
2290 CLS:RETURN
2300 REM ** create a data file and sav
e it onto tape **
2310 CLS:LOCATE 5,2:PEN 1:PRINT"To cre
ate a date file and save it."
2320 FOR z=1 TO 60
2330 LOCATE 10,4:PEN 3:PRINT"Question
number ";z
2340 PEN 2:LOCATE 1,6:PRINT"Please typ
e in word number 1 ";
2350 PEN 1:INPUT a$(z,1)
2360 PEN 2:LOCATE 1,8:PRINT "Please ty
pe in word number 2 ";
2370 PEN 1:INPUT a$(z,2)
2380 PEN 2:LOCATE 1,10:PRINT "Please t
ype in word number 3 ";
2390 PEN 1:INPUT a$(z,3)
2400 PEN 2:LOCATE 1,12:PRINT "Please t
ype in word number 4 ";
2410 PEN 1:INPUT a$(z,4)
2420 PEN 2:LOCATE 1,14:PRINT"Which of
these is the odd one out (1-4) ";
2430 PEN 1:INPUT aa%(z):IF aa%(z)<1 OR
aa%(z)>4 THEN PRINT CHR$(7):GOTO 2420
2440 PEN 2:LOCATE 4,20:PRINT"Is this c
orrect (Y)es / (N)o ?"
2450 a$=INKEY$:a$=UPPER$(a$):IF a$="N"
THEN CLS:GOTO 2330
2460 IF a$<>"Y" THEN 2450
2470 CLS:NEXT z
2480 CLS:LOCATE 2,2:PEN 1:PRINT"Please
insert a blank tape now.":PEN 2
2490 OPENOUT "data"
2500 FOR z= 1 TO 60
2510 FOR y=1 TO 4
2520 PRINT #9,a$(z,y):NEXT y
2530 PRINT #9,aa%(z):NEXT z
2540 CLOSEOUT
2550 CLS:RETURN
2560 REM ** instructions **
2570 PEN #2,0:CLS #2:LOCATE #2,1,1:PRI
NT#2,"In this game, you will be shown
four "
2580 PRINT#2,SPC(5);"words and you mus
t find the"
2590 PRINT#2,SPC(12);"'ODD MAN OUT'"
2600 PRINT#2,SPC(3);"before Jumbo eats
all his buns."
2610 PRINT#2,"To make life harder, yo
u have only 3";SPC(15);"lives.";
2620 PEN #2,3:PRINT#2:PRINT#2:PRINT#2,
SPC(3);"Press the <Space Bar> to start
."
2630 a$=INKEY$:IF a$<>" " THEN 2630
2640 PEN #2,0:GOSUB 1520
2650 RETURN
```

Give your fingers a rest . . .

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

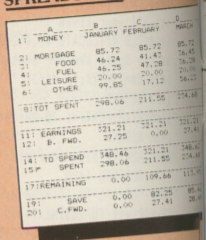
See our special offer on Page 77.

QUICK TO LEARN

THAT'S...

MINI OFFICE

SPREADSHEET



JUST LOOK WHAT THIS PACKAGE CAN DO!

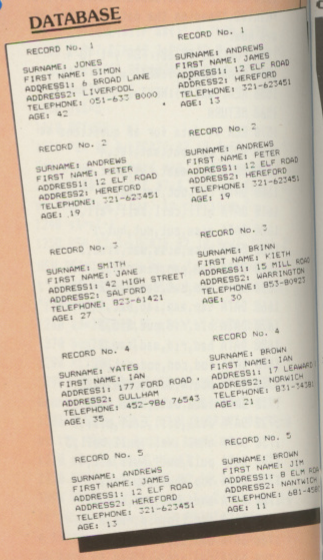
WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

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

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

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

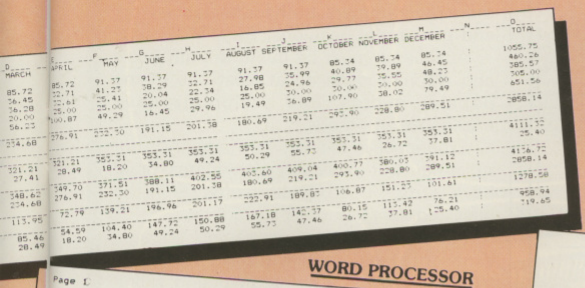
DATABASE



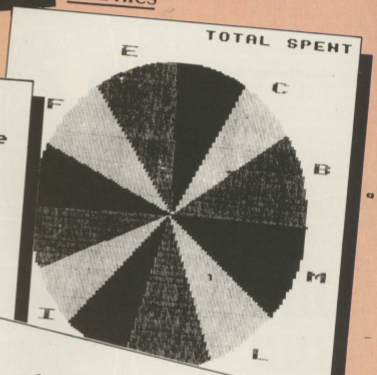
...and it's all at t price of just £

RN, EASY TO USE

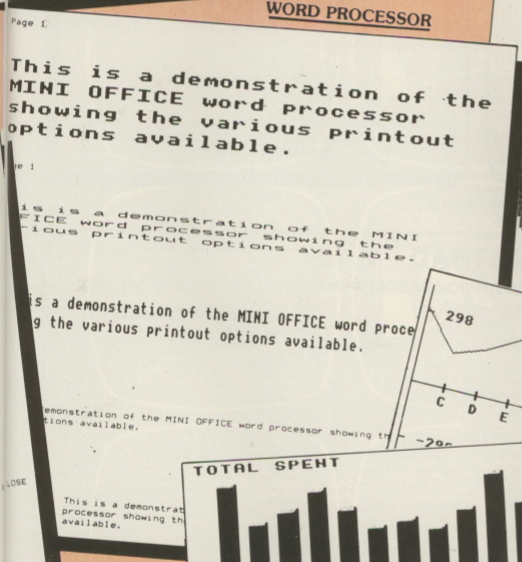
Specially written for your AMSTRAD CPC 464

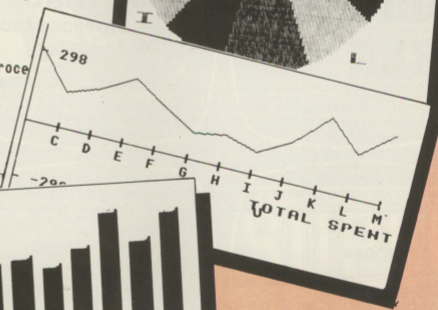


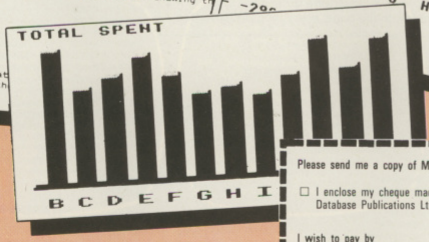
GRAPHICS



WORD PROCESSOR







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. CA7

at the unbelievable

£5.95 CASSETTE
3" DISC £9.95

DATABASE SOFTWARE

SUPERCHARGE YOUR AMSTRAD.

'SUPERPOWER' SIDEWAYS ROMCARD FOR THE AMSTRAD

This unit opens up a whole new field of personal computing, previously only available to owners of the BBC Micro and other top of the range computers.

Rom-based software is instantly available from the keyboard and being written in machine-code is very fast in operation.

The SUPERPOWER SIDEWAYS ROMCARD (Ref B101) has the following features:-

Matching Case, with easily detachable cover. Fits snugly to rear of computer.

Bus extension for fitting of Disk Interface or other units. Houses up to 7 foreground or background Roms – any mix of 8K and 16K.

Simple selection of programs from Keyboard – <BAR> Mail etc. No additional power supply necessary. Additional cards can be daisy chained for further capacity.

HIGH QUALITY BLANK EPROMS AVAILABLE. These are 200 nanosecond devices, the speed recommended by AMSOFT.

8K Eprom £7.95 (Ref B 104). 16K £11.95 (Ref B105).

LIMITED INTRODUCTORY OFFER

Until 30th June the SUPERPOWER ROMCARD will only be available by Mail Order. We are giving away a 16K Eprom (worth over £10) with all orders received by that date. Moreover, this will contain a FREE copy of Micro Power's highly acclaimed 'Ghouls' program to demonstrate the instant nature of Rom based Software.

SUPERPOWER SIDEWAYS ROMCARD (Ref B101) £39.95.

'SUPERPOWER' ROM BASED SOFTWARE

A complete range of Rom based programs is under development, releases being planned at the rate of two per month, starting in June. Available from June 1st are the Mailing List/Club Membership program plus the Disk User's Utilities Rom.

Later releases will include Word Processor, Spreadsheet, Database and Graphics/Printer routines packages – with data interchange facilities – PLUS an Enhanced Basic/Toolkit program and a Machine-Code Monitor.

SUPERPOWER MAILING LIST and club membership PROGRAM (Ref B102) £39.95.

'SUPERPOWER' MAILING LIST (AND CLUB MEMBERSHIP) PROGRAM (REF B102)

The program handles very large lists of names and addresses on a selective basis, acts as a simple Database, and is particularly suitable for Club Membership records. The main features are as follows:- Each Record can contain up to 19 fields, those to appear on labels being user-selected. Variable length fields are used to optimise memory and disk space.

In practice, approximately 2000 records containing name and address and two non-label fields can be held on one side of a disk. Multiple double-sided disks are catered for. Each record can have up to 20 classification indicators. When used as Membership List, 12 can be nominated for monthly subscription reminders.

A screen report gives breakdown by categories. Printing options include Label fields only, and Total Record including classifications. Label can be of two standard sizes or user-defined. Program works with any parallel printer.

Alphabetical Order is dealt with on Entry. User choice of Keyword e.g. 'J' ohn or 'S' mith.

The Name field can be searched for the first part or the whole of a keyword. The whole file can also be searched for any string. There is sophisticated line and character editing, including change of keyword. Function Keys can be defined to give single key entry of commonly used string e.g. 'Membership No.' In Entry mode an automatically incrementing number is available. Foreground and Background colours can be selected.

SUPERPOWER DISK USER'S UTILITIES ROM (Ref B103) £39.95.

'SUPERPOWER' DISK USER'S UTILITIES ROM (REF B103)

This program allows detailed inspection and modification of information held on disk. It is of particular use in the recovery of data from corrupted disks. Individual sectors can be read from and written to. All data can be output to the screen and/or printer. The program also contains a number of functions of use to the assembly language programmer.

Main Functions.

FILELOAD – loads first sector into buffer and remainder into memory for fast access later.

READ – reads a sector into the buffer and enters Edit Mode.

EDIT – displays the current buffer. Data displayed is Buffer Address, Hexadecimal representation of each byte and ASCII representation of each byte.

Depending on mode selected, display is of 12 or 24 lines of 8 or 16 bytes.

– modification of Hex numbers, changing the ASCII automatically and vice versa.

– comprehensive cursor controls for easy editing.

FIND – can search a sector or total file for an ASCII string.

WRITE – writes a sector previously read by READ or FILELOAD.

SUBSIDIARY FUNCTIONS –

CATALOGUE – similar to AMSDOS catalogue.

MODE – Select 40 col./12 line display or 80/12, 40/24 or 80/24.

INK – Select Background and Foreground Colours.

ROM CHECK – lists all sideways Roms, giving Position, Foreground or Background, Name, Version No. etc.

OTHER DISK COMMANDS – Access to other commands such as Format and Verify is provided directly from the Rom.

ASSEMBLY PROGRAMMER'S AIDS.

Disassembler. Relative Jump Calculator. Calculate the Sum and Difference of two hex numbers. Hex to Decimal Conversions – and vice versa. Intelligent Copy.

SUPER POWER

ROM-BASED SOFTWARE
FULFILS THE PROMISE OF YOUR AMSTRAD.

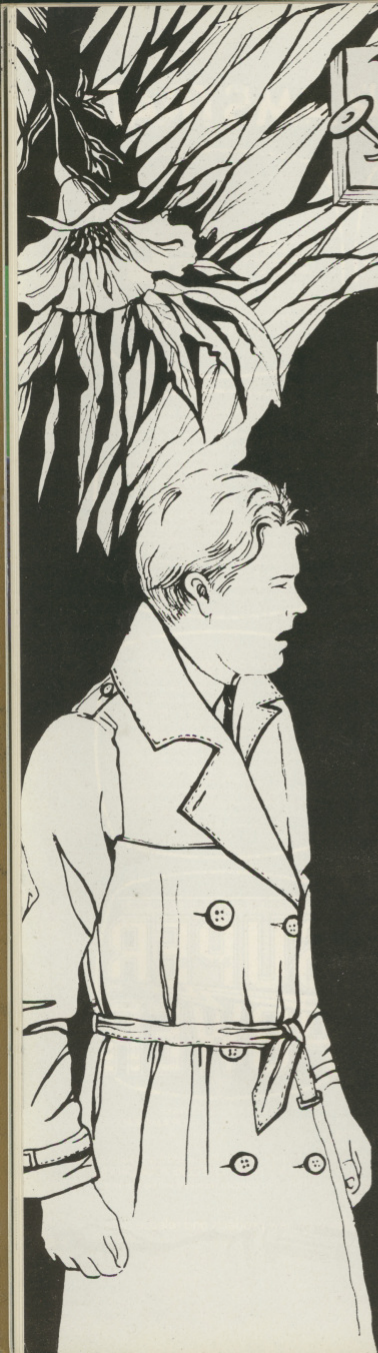
SuperPower – Serious software from the Micro Power organisation

DEALERS! SuperPower represents an exceptional profit opportunity. Ensure your part of the action by becoming a SuperPower Advice Centre and Stockist and benefiting from our dealer support package. Contact Eileen Garfield on 0532 434006.

HOW TO ORDER Phone or write to: The SuperPower Project Manager, Sheepscar House, Sheepscar Street South, LEEDS LS7 1AD. TELEPHONE (0532) 434006. State your NAME and ADDRESS and the REFERENCE NUMBERS of the products you wish to purchase. Buy on Access/Visa card by stating your card number or write enclosing a cheque made payable to: Micro Power Ltd. (Please add 95p to your full order amount to cover post and packing.) ACT NOW TO SECURE YOUR ORDER AND YOUR INTRODUCTORY OFFER. (No cheques received will be banked before your order is despatched.)

KEEP UP TO DATE Phone (0532) 434006 to join our mailing list and receive regular bulletins on new products and release dates.

Dare you go



ARE you fed up with shooting aliens, jumping barrels or hopping roads? You are? Then why not try an adventure?

"A what?" you cry. I'll explain. An adventure is a fantasy world, which you, the hero, have to explore, generally with the object of finding treasure or rescuing princesses.

"Kid's stuff!" I hear you shout, but that's just it, it isn't. You see, the crafty programmer who's written the game doesn't want you to win too easily, so he makes it as hard as possible, which is often very hard indeed.

Believe you me, when you've spent an hour trying to find a key to open a mysterious locked door, only to find that the door is locked from the other side, you'll be ready to strangle that programmer.

An adventure is like a detective novel, full of clues, puzzles and red herrings. Your job is to sift the clues, solve the puzzles and, hopefully, recognise the red herrings. What's more, because you're in a fantasy world with its own natural laws, you can also have goblins, magic or even aliens to cope with.

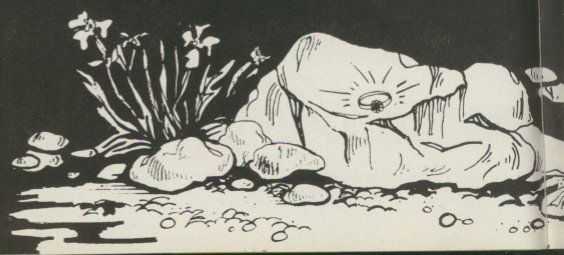
Not quite that easy after all, is it?

So, where do these adventure games come from? Well, they owe their origins to the Dungeons and Dragons (D&D) craze that swept America in the mid-'70s. Two mainframe programmers, Crowther and Woods, wrote a program called Colossal Cave, which simulated a D&D game, but had more emphasis on problem-solving and less on fighting monsters.

This game quickly achieved cult status among other programmers and might have remained on the mainframes but for an enterprising man called Scott Adams. He adapted one of these massive programs to a 16k TRS-80, published it, and the first adventure for a home micro, Adventureland, was released.

Since then many adventures have been written and they can be split into four basic types – the swords and sorcery type, text, graphical and arcade adventures.

The swords and sorcery type are attempts to emulate a



go Adventuring?

By Gandalfh

true D&D game in that you choose the character you wish to play, such as a warrior, cleric, barbarian, wizard, village idiot, and so on.

Each of these characters has its own inimical nature and your job is to minimise weaknesses and exploit strengths to complete the objectives set in the adventure, such as collecting treasure.

Since this is generally in the possession of some monster or other, you spend most of your time fighting them, and so your progress often seems to depend more on luck than skill.

Text adventures earn their name because they consist of text only and are based on the type of problems found in the original Crowther and Woods game. They require you to use your brain rather than your sword arm.

The third type of adventure, graphical, is an extension of either of the first two. They are only made possible by the massive memory available on computers such as the Amstrad.

Graphical adventures are, simply put, adventures with pictures. They range from those in which you move a man around on-screen, fighting monsters and bandits and so on, to those which are text adventures that "show" you your location.

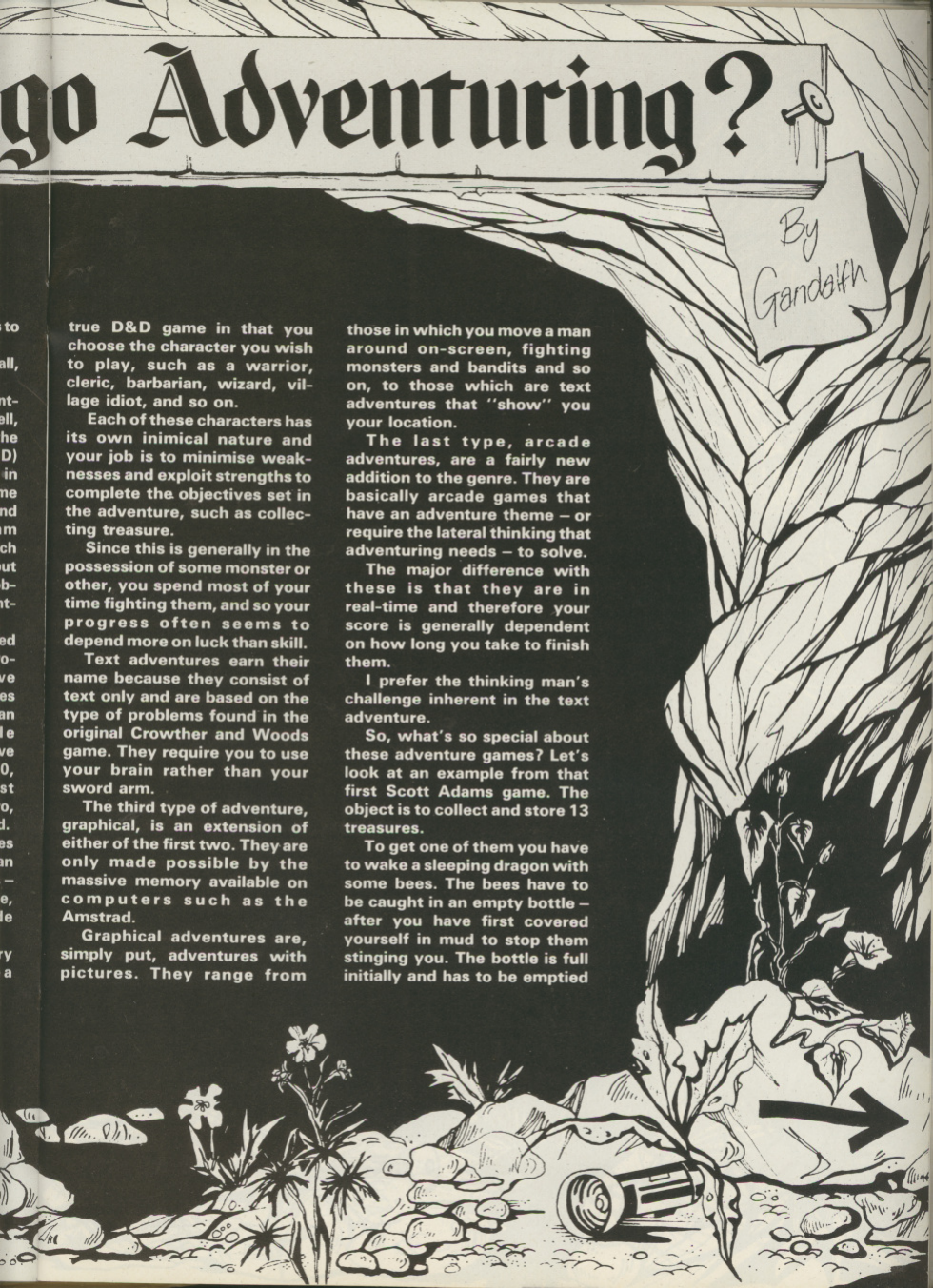
The last type, arcade adventures, are a fairly new addition to the genre. They are basically arcade games that have an adventure theme – or require the lateral thinking that adventuring needs – to solve.

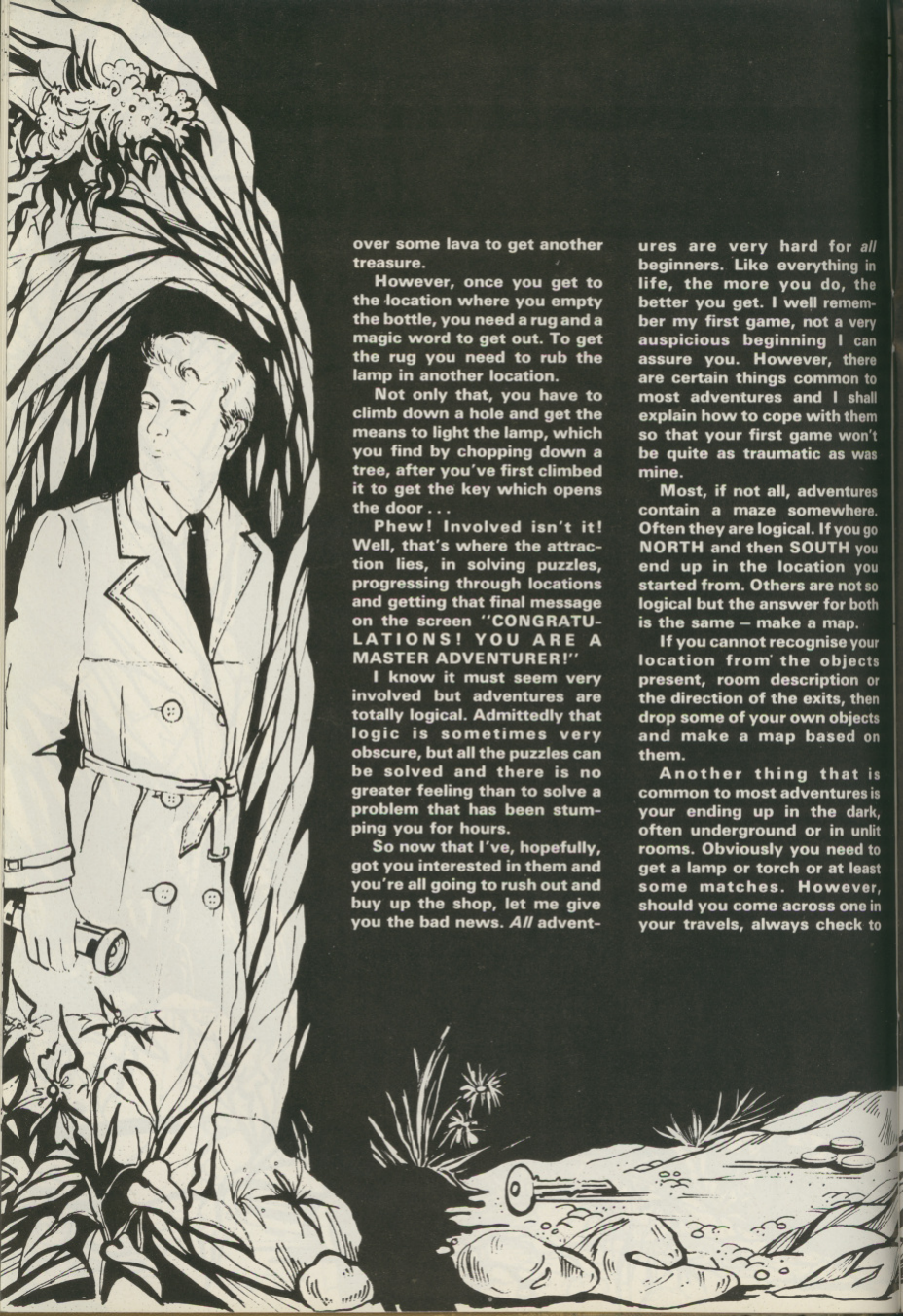
The major difference with these is that they are in real-time and therefore your score is generally dependent on how long you take to finish them.

I prefer the thinking man's challenge inherent in the text adventure.

So, what's so special about these adventure games? Let's look at an example from that first Scott Adams game. The object is to collect and store 13 treasures.

To get one of them you have to wake a sleeping dragon with some bees. The bees have to be caught in an empty bottle – after you have first covered yourself in mud to stop them stinging you. The bottle is full initially and has to be emptied





over some lava to get another treasure.

However, once you get to the location where you empty the bottle, you need a rug and a magic word to get out. To get the rug you need to rub the lamp in another location.

Not only that, you have to climb down a hole and get the means to light the lamp, which you find by chopping down a tree, after you've first climbed it to get the key which opens the door . . .

Phew! Involved isn't it! Well, that's where the attraction lies, in solving puzzles, progressing through locations and getting that final message on the screen "CONGRATULATIONS! YOU ARE A MASTER ADVENTURER!"

I know it must seem very involved but adventures are totally logical. Admittedly that logic is sometimes very obscure, but all the puzzles can be solved and there is no greater feeling than to solve a problem that has been stumping you for hours.

So now that I've, hopefully, got you interested in them and you're all going to rush out and buy up the shop, let me give you the bad news. *All* adventures are very hard for *all* beginners. Like everything in life, the more you do, the better you get. I well remember my first game, not a very auspicious beginning I can assure you. However, there are certain things common to most adventures and I shall explain how to cope with them so that your first game won't be quite as traumatic as was mine.

Most, if not all, adventures contain a maze somewhere. Often they are logical. If you go NORTH and then SOUTH you end up in the location you started from. Others are not so logical but the answer for both is the same – make a map.

If you cannot recognise your location from the objects present, room description or the direction of the exits, then drop some of your own objects and make a map based on them.

Another thing that is common to most adventures is your ending up in the dark, often underground or in unlit rooms. Obviously you need to get a lamp or torch or at least some matches. However, should you come across one in your travels, always check to

see if you can light it first.

Do you need matches, batteries or oil? If you do end up in "PITCH DARKNESS", try and reverse the move you have just made, or if that proves fatal, try to find the lamp and the means of lighting it before you re-visit that location.

If you've got the lamp, try LIGHT LAMP or ON or anything else you can think of before moving.

A few other tips that might help you involve shovels, scenery and ropes. If you find a shovel, it's a good bet that you will have to DIG somewhere, either to find a treasure or to get an object that will help you somewhere else in the adventure.

Examine your surroundings. If you are in a forest, can you climb a tree? Or if you've got an axe, can you chop that tree down?

Can you climb a wall, or a statue? If you find a rope it's likely to be needed somewhere to either climb something, or perhaps pull something. An object that's too heavy to lift might be pulled if you TIE ROPE and PULL object.

If the program allows you to save the game to return to the location you have reached should something you do prove fatal, then use it before you enter any suspicious places, or before trying something dangerous.

If something doesn't work immediately, such as taking a bucket stuck in the mud with TAKE BUCKET, try doing it a couple of times. Programmers are a crafty bunch, and sometimes make you do a thing a few times before you succeed.

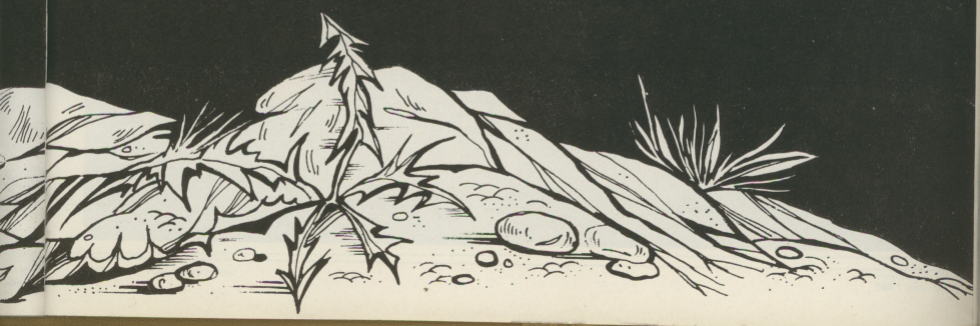
Always read the room descriptions very carefully. Sometimes clues are hidden here. And always EXAMINE everything!

Well I hope you now have an idea of what adventuring is all about and you never know, maybe we'll be seeing you staggering into the computer club with bleary eyes sometime.

If you're stuck with any adventure, why not drop me a line c/o *Computing with the Amstrad,* and I'll do my best to help. But be sure to enclose a sae for your reply.

Happy adventuring,

Gandalfh




MISSILE
COMMAND

ROLAND WADDILOVE hands over command of the missile battery to you – and it's now your job to save the cities from alien attack

IN this action-packed arcade classic you are commander of one of Earth's many missile batteries. Your duty is to protect the surrounding cities from attack by hostile aliens who send their missiles streaking toward you.

At your disposal are powerful thermonuclear warheads which can be fired at the aliens' missiles. By carefully positioning your sights the warhead can be directed so as to explode just in front of a missile. This is sufficient to destroy it.

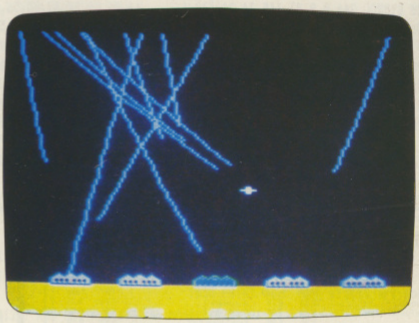
If you survive the first wave another swiftly follows – there's hardly time to catch your breath. There are 10 incoming missiles on each screen, and with each screen they get faster and faster.

The program has a high score table capable of holding the top five scores. There are nine levels of difficulty and any one can be selected at the start, the lowest being the easiest. Music accompanies both the high score table and title screen.

The program is structured and fairly easy to follow. Each subroutine has a title describing its function to help debugging.

VARIABLES

m(10)	Missile coordinates.
x(10)	X increment.
y(10)	Y increment.
t(10)	Target for each missile.
c(5)	Cities.
name$(5)	Names for high score table.
hiscore(5)	High scores.
score	Score.
screen	Screen.



```
10 REM Missile Command
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 GOSUB 1650:REM instructions
50 GOSUB 530:REM initialise
60 WHILE NOT bored.stiff
70 GOSUB 310:REM set variables
80 WHILE c(3)
90 GOSUB 980:REM screen
100 GOSUB 730:REM new missiles
110 WHILE c(3)=1 AND h<11
120 GOSUB 1160:IF h=10 THEN h=h+1
130 FOR i=1 TO 5
140 IF m(i,1)>60 THEN GOSUB 930 ELSE
IF c(t(i)) THEN GOSUB 840
150 NEXT
160 IF f<0 THEN GOSUB 1210 ELSE IF IN
KEY(47)>-1 THEN a=x%-16:b=y%+10:SOUND
 129,10,40,15,1,1,1:SOUND 1,1000,100,
12,1,1,10:GOSUB 1210
170 GOSUB 1160
180 FOR i=6 TO 10
190 IF m(i,1)>60 THEN GOSUB 930 ELSE
IF c(t(i)) THEN GOSUB 840
200 NEXT
210 WEND
220 IF f<0 THEN GOSUB 1210
230 IF c(3) THEN GOSUB 1290:REM bonus
240 WEND
250 SOUND 130,200,200,15,1,3,15:SOUND
 4,200,200,15,1,3,15:SOUND 4,200,200,
15,1,3,15:FOR j=0 TO 1000:NEXT:LOCATE
 5,10:PRINT " GAME  OVER "
260 FOR j=0 TO 6000:NEXT
270 IF score>hiscore(5) THEN GOSUB 14
00
280 WEND
290 END
300 REM ----------------------------
310 REM set variables
320 screen=0
330 MODE 1:BORDER 1:INK 1,24:INK 2,20
:INK 3,6
340 DRAW 0,398,1:DRAW 638,398:DRAW 63
8,0:DRAW 0,0
350 PAPER 3:PEN 1:LOCATE 11,1:PRINT "
   Top  Commanders   "
360 LOCATE 6,20:PRINT " 1-9 selects s
tart level => 1 "
370 PAPER 1:PEN 3:LOCATE 8,25:PRINT "
 Press space bar to start ":PAPER 0
380 WHILE INKEY$<>"":WEND
390 i=1:k$=""
400 WHILE k$<>CHR$(32):k$=INKEY$
410 IF k$>"0" AND k$<":" THEN PEN 1:P
APER 3:LOCATE 34,20:PRINT k$:screen=V
AL(k$)-1:PAPER 0
420 LOCATE 7,4+i*2:PEN 1:PRINT i;:PEN
 INT(RND*3+1):PRINT " ";name$(i);" ";
:PEN 3:WHILE POS(#0)<29:PRINT ".";:WE
ND:PEN 1:PRINT hiscore(i)
430 i=i+1:IF i=6 THEN i=1:INK 2,1+INT
(RND*26)
440 IF SQ(1)=4 THEN GOSUB 1600
450 WEND
```

```
460 MODE 0:BORDER 0:INK 2,13:INK 3,13
470 WINDOW #1,1,20,1,22:PAPER #1,0
480 WINDOW #2,1,20,23,25:PAPER #2,14
490 ORIGIN 0,0,1000,2000,3000,4000:CL
G 0:ORIGIN 0,0,0,640,0,400
500 score=0:c(3)=1:PRINT CHR$(7)
510 RETURN
520 REM ----------------------------
530 REM initialise
540 SYMBOL AFTER 240
550 SYMBOL 240,17,42,85,128,128,128,1
70,128:SYMBOL 241,136,84,170,1,1,1,85
,1
560 SYMBOL 242,0,17,42,127,85,85,85,1
27:SYMBOL 243,0,136,84,254,170,170,17
0,254
570 SYMBOL 244,0,0,0,0,32,32,17,80,127:
SYMBOL 245,0,0,0,0,16,2,128,74,254
580 SYMBOL 246,0,0,0,97,145,74,74,132,1
28:SYMBOL 247,0,16,16,169,217,42,181,
1
590 SYMBOL 248,63,127,255,255,255,255
,255,255:SYMBOL 249,252,254,255,255,2
55,255,255,255
600 SYMBOL 250,255,255,255,255,255,25
5,127,63:SYMBOL 251,255,255,255,255,2
55,255,254,252
610 SYMBOL 255,255,255,255,255,255,25
5,255,255
620 DEFINT a-f,h-l,n-w
630 DIM m(10,1),x(10),y(10),t(10),c(5
),name$(5),hiscore(5)
640 FOR i=1 TO 5:name$(i)="The A Team
":hiscore(i)=60-i*10:NEXT
650 FOR i=0 TO 15:READ j:INK i,j:NEXT
660 DATA 0,24,13,13,26,26,0,0,20,8,0,
0,18,2,15,26
670 INK 10,3,6:INK 11,6,3
680 RESTORE 690:FOR i=0 TO 54:READ a$
:POKE i+&8000,VAL("&"+a$):NEXT
690 DATA DD,6E,04,DD,66,05,DD,5E,06,D
D,56,07,CD,C0,BB,3E,04,CD,DE,BB,3E,01
,CD,59,BC,3E,28,CD,FC,BB,DD,6E,00,DD,
66,01,DD,5E,02,DD,56,03,CD,C0,BB,3E,2
8,CD,FC,BB,3E,00,C3,59,BC
700 ENT 1,100,4,1:ENV 1,8,-1,15:ENT 2
,200,-1,1:ENT -3,10,10,10,10,-10,10
710 RETURN
720 REM ----------------------------
730 REM new missiles
740 h=0:INK 1,17+INT(RND*9)
750 FOR i=1 TO 10
760 m(i,0)=-100+640*RND:m(i,1)=400+RN
D*100
770 t(i)=INT(5*RND+1)
780 grad=(m(i,1)-60)/((110*t(i))-m(i,
0))
790 y(i)=1+RND*screen*4
800 x(i)=y(i)/grad
810 NEXT
820 RETURN
830 REM ----------------------------
840 REM hit city
850 SOUND 130,200,200,15,1,3,15:c(t(i
))=0
860 TAG:PLOT -10,-10,0:MOVE 110*t(i)-
30,60:PRINT CHR$(255);CHR$(255);:TAGO
FF
870 PRINT CHR$(23);CHR$(1);:TAG
880 PLOT -10,-10,10:MOVE 110*t(i)-30,
60:PRINT CHR$(244);CHR$(245);
890 PLOT -10,-10,11:MOVE 110*t(i)-30,
60:PRINT CHR$(246);CHR$(247);
900 TAGOFF:PRINT CHR$(23);CHR$(0);
910 RETURN
```

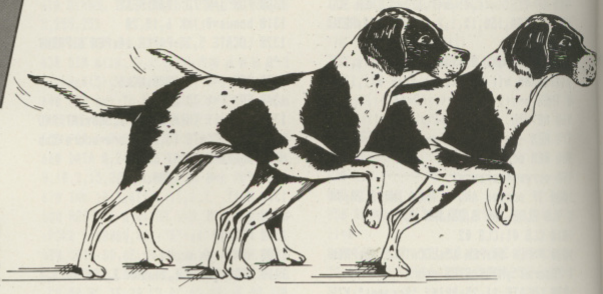
```
920 REM ------------------------------
930 REM missile
940 IF TEST(m(i,0),m(i,1))=3 THEN SOU
ND 132,500,150,15,1,2:GOSUB 1250:RETU
RN
950 m(i,0)=m(i,0)+x(i):m(i,1)=m(i,1)-
y(i):DRAW m(i,0),m(i,1),1:IF m(i,1)<6
0 THEN h=h+1
960 RETURN
970 REM ------------------------------
980 REM screen
990 screen=screen+1
1000 IF screen AND 1 THEN INK 8,20:IN
K 9,8 ELSE INK 8,23:INK 9,7
1010 CLS #1:CLS #2
1020 PAPER 14:PEN 15:LOCATE 1,25:PRIN
T "Score:";MID$(STR$(score),2)
1030 LOCATE 11,25:PRINT "Screen:";MID
$(STR$(screen),2)
1040 PRINT CHR$(23);CHR$(1);:TAG
1050 FOR i=1 TO 5
1060 c(i)=1
1070 IF i<>3 THEN PLOT -100,-100,8 EL
SE PLOT -10,-10,12
1080 MOVE 110+i-30,60:PRINT CHR$(240)
;CHR$(241);
1090 IF i<>3 THEN PLOT -100,-100,9 EL
SE PLOT -10,-10,13
1100 MOVE 110+i-30,60:PRINT CHR$(242)
;CHR$(243);
1110 NEXT
1120 f=1:x%=316:y%=100:PLOT -10,-10,4
:MOVE x%,y%:PRINT "+";
1130 TAGOFF:PRINT CHR$(23);CHR$(0);
1140 RETURN
1150 REM ------------------------------
1160 REM move sights
1170 SOUND 130,2000,100,2:xx%=x%+16*(
INKEY(7)>-1)-16*(INKEY(63)>-1):yy%=y
%+8*(INKEY(30)>-1)-8*(INKEY(28)>-1):I
F (xx%<10 OR xx%>620) OR (yy%<100 OR
yy%>390) THEN RETURN
1180 CALL &8000,x%,y%,xx%,yy%:x%=xx%:
y%=yy%
1190 RETURN
1200 REM ------------------------------
1210 REM explosion
1220 f%=-f%:PRINT CHR$(23);CHR$(1);:M
OVE 330,60:DRAW a+32,b-16,2:TAG:MOVE
 a,b:PRINT CHR$(248);CHR$(249);:MOVE
a,b-16:PRINT CHR$(250);CHR$(251);:TAG
OFF:PRINT CHR$(23);CHR$(0);
1230 RETURN
1240 REM ------------------------------
1250 REM hit missile
1260 m(i,1)=10000:score=score+screen:
h=h+1:LOCATE 7,25:PRINT MID$(STR$(sco
re),2)
1270 RETURN
1280 REM ------------------------------
1290 REM bonus
1300 FOR j=0 TO 1000:NEXT
1310 bonus=0:INK 6,10,20
1320 LOCATE 5,23:PAPER 14:PEN 6:PRINT
 "B o n u s "
1330 FOR j=0 TO 2000:NEXT
1340 FOR i=1 TO 5
1350 IF c(i) THEN SOUND 3,500+INT(RND
*500),50:LOCATE 15,23:score=score+5:b
onus=bonus+5:PRINT bonus
1360 FOR j=0 TO 1000:NEXT
1370 NEXT
1380 INK 6,0
1390 RETURN
1400 REM high score
1410 MODE 1:INK 1,26:INK 2,18:INK 3,1
9
1420 DRAW 0,398,1:DRAW 638,398:DRAW 6
38,0:DRAW 0,0
1430 PEN 1:PAPER 0:LOCATE 7,5:PRINT "
This is a new high score"
1440 LOCATE 10,10:PEN 3:PRINT "Please
 type in your name"
1450 LOCATE 14,20:PAPER 2:PEN 0:PRINT
 " ?..... "
1460 LOCATE 16,20:k$="":name$(5)="":f
=1
1470 WHILE INKEY$<>"":WEND
1480 WHILE k$<>CHR$(13)
1490 IF SQ(1)=4 THEN GOSUB 1600
1500 IF k$>CHR$(31) AND k$<"z" AND LE
N(name$(5))<10 THEN name$(5)=name$(5)
+k$:PRINT k$;
1510 IF k$=CHR$(127) AND LEN(name$(5)
) THEN name$(5)=LEFT$(name$(5),LEN(na
me$(5))-1):PRINT CHR$(8);CHR$(16);
1520 k$=INKEY$
1530 WEND
1540 hiscore(5)=score
1550 FOR i=5 TO 2 STEP -1
1560 IF hiscore(i)>hiscore(i-1) THEN
k$=name$(i):name$(i)=name$(i-1):name$
(i-1)=k$:score=hiscore(i):hiscore(i)=
hiscore(i-1):hiscore(i-1)=score
1570 NEXT
1580 RETURN
1590 REM ------------------------------
1600 REM tune
1610 f=(f+1)MOD 15
1620 IF f<5 THEN SOUND 1,119,10:SOUND
 1,95,10:SOUND 1,80,10:SOUND 130,239,
500:SOUND 132,190,500:RETURN
1630 IF f>9 THEN SOUND 1,89,10:SOUND
1,71,10:SOUND 1,60,10:SOUND 130,179,5
00:SOUND 132,142,500:RETURN
1640 SOUND 1,80,10:SOUND 1,63,10:SOUN
D 1,53,10:SOUND 130,159,500:SOUND 132
,127,500:RETURN
1650 REM instructions
1660 MODE 1:CALL &BB4E:CALL &BC02
1670 INK 2,0:INK 0,13:BORDER 13
1680 WINDOW #1,11,31,1,3:PAPER #1,3:C
LS #1
1690 WINDOW #1,10,30,2,4:PAPER #1,1:C
LS #1
1700 PEN #1,2:LOCATE #1,4,2:PRINT #1,
"Missile Command"
1710 WINDOW #1,11,31,22,24:PAPER #1,3
:CLS #1
1720 WINDOW #1,10,30,23,25:PAPER #1,1
:CLS #1
1730 LOCATE #1,6,2:PRINT #1,"Press SP
ACE"
1740 WINDOW #1,2,40,6,20:PAPER #1,3:C
LS #1
1750 WINDOW #1,1,39,7,21:PAPER #1,1:C
LS #1
1760 PRINT #1
1770 PRINT #1," As commander of one o
f Earth's many"
1780 PRINT #1," missile batteries you
r task is to"
1790 PRINT #1," protect the surroundi
ng cities from"
1800 PRINT #1," missile attack."
1810 PRINT #1
1820 PRINT #1," They can be stopped b
y exploding a"
1830 PRINT #1," nuclear warhead in th
eir path."
1840 PRINT #1
1850 PRINT #1," Your sights are contr
olled with..."
1860 PRINT #1
1870 PRINT #1,"  Z=left    X=right
 +=up    ?=down"
1880 PRINT #1
1890 PRINT #1,SPC(8)"SPACE bar fires
warhead"
1900 WHILE INKEY$<>"":WEND
1910 WHILE INKEY$<>CHR$(32)
1920 IF SQ(1)=4 THEN GOSUB 1600
1930 WEND
1940 CLS:INK 0,0
1950 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.

Part VII of MIKE BIBBY's introduction to machine code



LAST month we saw how our single registers could be combined to form register pairs BC, DE, HL, each capable of holding a 16 bit number. This means a register pair can hold any number in the range &0 – &FFFF (0 – 65535).

Loading the register pair with a constant couldn't be easier – just specify the relevant opcode for the pair in question, followed by two bytes containing the constant in our usual lo byte, hi byte fashion.

For example,

```
LD BC, &2FF8
```

would translate as:

```
01 F8 2F
```

We also met a similar looking type of instruction:

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

In this case, though, the &2FF8 isn't a constant – it's pointer to a memory location. As we've seen, it's a sort of two byte peek, with C taking the value stored in memory location &2FF8 and B taking the value stored in &2FF9.

It translates as:

```
ED 4B F8 2F
```

The brackets make a considerable difference!

The register pairs' ability to specify any number in the range &0 to &FFFF means that we can use them to point at, or index, any location in memory.

In fact we often use register pairs in this way. For instance, if HL contained &2FF8, the instruction:

```
LD A, (HL)
```

would load the A register with the *contents* of memory location &2FF8. In effect it's doing an 8 bit peek, with the address to be peeked stored in register pair HL.

And, just as you can "peek", so you can "poke":

```
LD (HL),A
```

would copy the contents of the A register into the memory location specified by HL.

Try Program I to see how this idea works:

```
address  hex code  mnemonics
3000     21 F8 2F  LD HL, &2FF8
3003     3E FF     LD A,&FF
3005     77        LD (HL),A
3006     C9        RET
```

Program I

Once it's executed, you should see that &2FF8 – the first location in Hexer's workspace – has the value &FF.

Line by line, the program:

- Sets the pointer to memory by loading HL with &2FF8.
- Loads the A register with &FF – our "marker byte".
- Loads the location pointed to by HL – &2FF8 – with the contents of A – &FF.
- Returns.

As you can see, all works as expected. You might be wondering why we didn't load A with &FF and "poke" directly, as in Program II:

```
address  hex code  mnemonics
3000     3E FF     LD A,&FF
3002     32 F8 2F  LD (&2FF8),A
3005     C9        RET
```

Program II

Actually, there's nothing wrong with doing it this way, and it does save you a couple of bytes. However, if you point at memory with HL rather than directly with a constant, you can take advantage of instructions such as:

```
INC HL   ( opcode &23 )
```

and:

```
DEC HL   ( opcode &2B )
```

to alter the memory location you're pointing at.

Program III shows the sort of thing I mean:

```
address  hex code  mnemonics
3000     21 F8 2F  LD HL, &2FF8
3003     3E FF     LD A, &FF
3005     77        LD (HL),A
3006     23        INC HL
3007     77        LD (HL),A
3008     23        INC HL
3009     77        LD (HL),A
300A     C9        RET
```

Program III

This puts &FF into three consecutive bytes of workspace. In practice, as you'll know from your Basic programming, you'd normally place the repeated:

```
LD (HL),A
INC HL
```

inside a loop, but more of that later.

To see if you've understood what we've done so far, try writing a program that has identical results to Program III, but using DEC HL instead of INC HL.

Then write another version that puts the values 0,1,2 into consecutive memory locations from &2FF8 onwards.

You'll probably need INC A –

Use register pairs as pointers to specify a memory location

opcode &3C).

One of the nice things about using HL to point to memory is that, in effect, the memory specified becomes an "extra register". That is, the instructions you've learned using single registers can often be used with (HL) substituted for one of those registers, the contents of HL pointing to the memory location you want to use.

For example, since:

```
INC A
```

and:

```
DEC A
```

exist, you won't be surprised to learn that there are instructions:

```
INC (HL)    opcode &34
```

and:

```
DEC (HL)    opcode &35
```

These work in exactly the same way as their single register counterparts except that, instead of directly specifying the register you want, you give the address in memory you want to effect via the HL register. Using HL in this way fills the "gaps" in our tables of opcodes. You may have noticed that there's a pattern to the opcodes. If not, have a look at last month's Ready Reference.

For instance, if you look at our old table for LD r,n you'll see that the opcode for the B register is &06, for C it's &0E, D is &16, E is &1E, H is &26 and L is &2E. See the pattern? Suddenly, though, there's a jump to &3E for A. What happened to the missing &36?

Well, &36 is the opcode for:

```
LD (HL),n
```

as you've probably already guessed. Table I shows the full range of opcodes for LD r,n including (HL).

LD B,n	06	n
LD C,n	0E	n
LD D,n	16	n
LD E,n	1E	n
LD H,n	26	n
LD L,n	2E	n
LD (HL),n	36	n
LD A,n	3E	n

Table I: LD r,n opcodes

	r'	B	C	D	E	H	L	(HL)	A
	B	40	41	42	43	44	45	46	47
	C	48	49	4A	4B	4C	4D	4E	4F
	D	50	51	52	53	54	55	56	57
r	E	58	59	5A	5B	5C	5D	5E	5F
	H	60	61	62	63	64	65	66	67
	L	68	69	6A	6B	6C	6D	6E	6F
	(HL)	70	71	72	73	74	75	*	77
	A	78	79	7A	7B	7C	7D	7E	7F

Table II: Opcodes for LD r,r'

(HL) also finds its place in the LD r,r' instructions, as you'll see in Table II. LD A, (HL) exists, as does LD (HL),A. Notice there's no opcode for LD (HL), (HL) – a meaningless instruction. I've placed an asterisk there.

To sound off our coverage, you won't be surprised to learn that you can:

```
ADD A,(HL)          &86
```

and:

```
SUB (HL)    &96
```

Using a register pair to point at memory this way is known as indirect addressing – you don't directly tell the Z80 the memory location you want, you tell it to go look at HL to find out!

As you'll see, indirect addressing can be very powerful, but the ideas take a bit of getting used to, so let's have some practice.

Program IV uses indirect addressing to store &FF in memory location &2FF8. Not as we did in Program I, by using the A register, but using LD (HL),&FF.

```
address  hex code   mnemonics
3000     21 F8 2F   LD HL,&2FF8
3003     36 FF      LD (HL),&FF
3005     C9         RET
```

Program IV

Now take a careful look at Programs V and VI. They illustrate the difference between INC HL and INC (HL).

```
address  hex code   mnemonics
3000     21 F8 2F   LD HL,&2FF8
3003     36 FF      LD (HL),&FF
3005     23         INC HL
3006     C9         RET
```

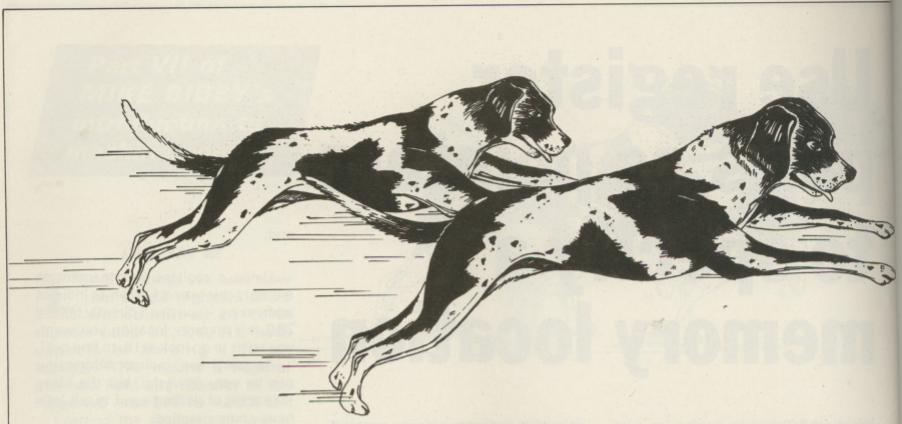
Program V

```
address  hex code   mnemonics
3000     21 F8 2F   LD HL,&2FF8
3003     36 FF      LD (HL),&FF
3005     34         INC (HL)
3006     C9         RET
```

Program VI

Program V is much the same as Program IV, except for an ineffectual INC HL at the end. The program simply puts &FF in &2FF8, as you'll see if you examine memory with Hexer. The INC HL increases the value in HL by one, to &2FF9.

In Program VI, however, after loading &FF into &2FF8, we then INC



(HL). This, as you'll see, doesn't change the number in HL (which is still &2FF8), but increases the contents of the memory pointed to by HL by one. That is, the contents of &2FF8 are increased by one.

And, since we've already loaded &2FF8 with &FF, increasing it by one wraps the value round to zero. Take a look with Hexer if you don't believe me.

So, INC HL increases the number in the register pair by one. INC (HL) increases the value of the memory location specified by HL by one.

Right, before you run Program VII, can you predict what it does?

```
address  hex code  mnemonic
  3000   21 F8 2F  LD HL,&2FF8
  3003   34         INC (HL)
  3004   23         INC HL
  3005   34         INC (HL)
  3006   23         INC HL
  3007   34         INC (HL)
  3008   C9         RET
```

Program VII

If you use Hexer to examine the workspace before and after you run the program you'll see that the first three bytes are increased by one.

So far, we've used the HL register as our pointer to memory. In fact, just as the A register has the most versatile range of instructions of the 8 bit registers, so HL is the "leader" of the 16 bit registers.

However, we can point into memory with BC, DE as well, but only in conjunction with the A register, as Table III shows.

```
mnemonic   opcode
LD (BC),A    02
LD (DE),A    12
LD A,(BC)    0A
LD A,(DE)    1A
```

Table III: Indirect addressing with BC,DE

Program VIII uses this idea to copy the first three bytes of the program (from &3000) down into the workspace. This may seem rather futile, but, as I said in the introduction to this series, much of machine code involves moving data from one location to another – and often in blocks.

In fact, the Z80 has special instructions for moving blocks of data about in memory, which we'll be covering.

In Program VIII we use two pointers to memory. HL is loaded with &3000, the start of the program, where we are copying bytes from, while DE is loaded with &2FF8, the start of the workspace we are copying to.

Next, A is loaded with the byte pointed at by HL with LD A, (HL).

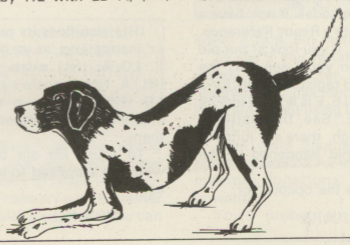
```
address  hex code  mnemonic
  3000   21 00 30  LD HL,&3000
  3003   11 F8 2F  LD DE,&2FF8
  3006   7E        LD A,(HL)
  3007   12        LD (DE),A
  3008   13        INC DE
  3009   23        INC HL
  300A   7E        LD A,(HL)
  300B   12        LD (DE),A
  300C   13        INC DE
  300D   23        INC HL
  300E   7E        LD A,(HL)
  300F   12        LD (DE),A
  3010   C9        RET
```

Program VIII

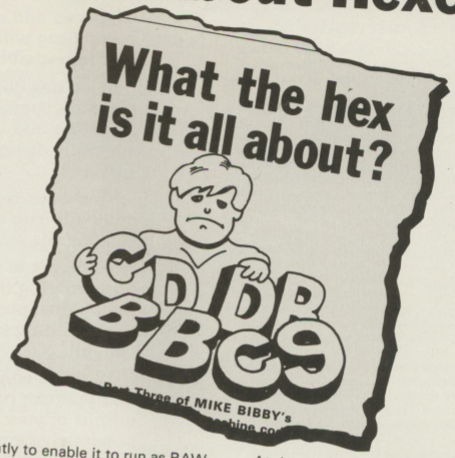
Then that value is copied into the location pointed at by DE with LD (DE),A.

Both DE and HL are increased by one so as to point to the next bytes along and the copying process repeated. DE and HL are then increased again and the copying done a final time.

● *Well, that's all for this month. Next month we'll continue our exploration of machine code with a look at loops.*



More about Hexer



WE'RE now at the stage where you might want to start saving and loading back the machine code programs you've created with Hexer. (Hexer was first published in the March 1985 issue of *Computing with the Amstrad.)*

Program I shows the lines you have to either add or alter to add these options.

Also, we've featured RAW, a full Z80 assembler, in this issue.

This new version of Hexer and RAW can quite easily be combined to produce a useful toolkit for developing machine code programs. Both can reside in the memory at the same time, provided they have different line numbers, and a simple menu can be added to allow one or the other to be selected.

First load Hexer, renumber it so that it starts at 10000 and save it as an Ascii file. Load RAW and merge Hexer, as follows:

```
LOAD "Hexer"
RENUM 10000
SAVE "Hexer",A
LOAD "RAW"
MERGE "Hexer"
```

Next, type Program II. Hexer needs altering very slightly to enable it to run as RAW takes up a fair chunk of memory. The machine code must be located higher in the memory so change any reference to &2FF8 to &7FF8, and change any &3000's or 3000's to &8000 or 8000.

At the start of Hexer MEMORY is set to &2FF8. This would need to be changed to &7FF8 and the start address is now &8000.

When you've made these changes the program can be saved and then run.

Program I

```
150 PRINT "5. End program"
155 PRINT "6. Save code"
156 PRINT "7. Load code"
170 IF INSTR("1234567",b$)=0 GOTO 160
180 b = VAL(b$) : ON b GOSUB 210,400,
550,350,200,700,800
700 INPUT "Start Address"; start$
710 IF start$ = "" THEN start$ = "300
0"
720 start = VAL( "&" + start$ )
730 INPUT "Name";name$
740 IF name$ = "" THEN GOTO 730
750 INPUT "Length"; length$
760 IF length$ = "" THEN length$ =
"400"
770 length = VAL( "&" + length$ )
780 SAVE name$,b,start,length
790 RETURN
800 INPUT "Name";name$
810 LOAD name$
820 RETURN
```

Program II

```
4000 REM PROGRAM II
4010 MODE 1:INK 0,0:BORDER 0
4020 LOCATE 10,5:PEN 3:PAPER 1:PRINT
" Programmer's Toolkit "
4030 LOCATE 15,10:PAPER 0
4040 PEN 1:PRINT "1. ";:PEN 2:PRINT "
Hexer."
4050 LOCATE 15,12
4060 PEN 1:PRINT "2. ";:PEN 2:PRINT "
Assembler."
4070 LOCATE 15,16
4080 PEN 3:PRINT "Press 1 or 2"
4090 IF INKEY(64)>-1 THEN CLS:GOTO 10
000
4100 IF INKEY(65)=-1 GOTO 4090
4110 PEN 1
```

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 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.

Computing with the Amstrad reader offer!

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

CA7

Discover some real characters!

In Part VI of our series on Amstrad graphics GEOFF TURNER and MICHAEL NOELS show you how to design your own character set

So far we've explored two ways of displaying information on the screen. We've used the text cursor for printing alphanumeric characters and the graphics cursor for drawing and plotting shapes.

Whichever method is used to write to the screen, the end result as far as the computer hardware is concerned is to change the colour of a number of points or pixels within the screen area.

Depending upon the particular combination of points selected, we hope to see a recognisable shape, whether a printed character such as the letter A or a shape drawn with the graphics cursor, like a triangle or circle.

In this article we are returning to the use of the text cursor, and we'll examine some of the many characters which can be printed on the screen. We'll also find out how these characters are produced.

Let's start off by resetting the computer. Now it's in command mode, and if we press any key then we expect that particular character to appear on the screen, positioned at the text cursor.

Press the letter A, and of course A will be displayed. Now if you look closely at the character it may be apparent that it is made up of a combination of dots. We could actually reproduce the letter A by using the PLOT command, as long as we can work out the correct combination of points to plot.

Program I demonstrates how the character may be plotted. Notice that we have printed A first at line 40, and then plotted a number of points to also produce the letter A.

Line 90 reads the X and Y coordinates from the DATA statements of lines 120 to 180. Notice that relative plotting is achieved by using the PLOTR command. The two characters produced are identical, but it was much easier to print the character rather than plot it.

Imagine having to plot every letter, number and punctuation mark! Even producing a short sentence would involve a considerable amount of work, and programming would be a tedious business.

Fortunately we don't need to plot any of these characters, as they can all be produced with the PRINT command. Buried somewhere deep in the Amstrad's firmware is all the information needed to generate the pattern of dots to represent all of our commonly-used characters, plus quite a few more. The work has been done for us.

The set of points required for each character is known as its bit pattern. The computer stores the bit pattern for each character and when it is asked to print a particular character it looks up the appropriate pattern and sends the data to the screen.

All this is performed by fast-working machine code routines, resulting in the character appearing even before we have had chance to release the key.

Each character is laid out or drawn on an 8 by 8 grid. Figure I illustrates how the letter A is produced. We used this particular bit pattern to plot the character in Program I.

The Amstrad has a capacity to store the information relating to 256 characters. If you wish to examine the bit pattern for any of them, they are all to be found in Appendix III of the CPC464 User Manual, and Chapter 7 of the CPC664 manual.

Each is given a number in the range 0 to 255 so that the computer may identify and print the correct character. For example, the letter A is referred to by the number 65 and we can use this code number to print the letter A.

So instead of entering:

```
PRINT "A"
```

we can use this alternative:

```
PRINT CHR$(65)
```

Both lines will produce a letter A on the screen. The second says, in effect: "Print the character whose code number is 65".

You might like to try using the above statement substituting some other numbers instead of 65. For the moment avoid using numbers below 32 and above 126. Any number within this range will produce a recognisable letter, number or punctuation mark.

The 256 built-in characters can be split into three distinct groups. Those in the number range 0 to 31 are a

```
10 REM PROGRAM I
20 MODE 1
30 LOCATE 1,5
40 PRINT"This is printed ";"A"
50 LOCATE 1,12
60 PRINT"This is plotted "
70 FOR pixel=1 TO 28
80 MOVE 256,208
90 READ x,y
100 PLOTR x,y
110 NEXT
120 DATA 2,2,4,2,10,2,12,2
130 DATA 2,4,4,4,10,4,12,4
140 DATA 2,6,4,6,6,6,8,6,10,6,12,6
150 DATA 2,8,4,8,10,8,12,8
160 DATA 2,10,4,10,10,10,12,10
170 DATA 4,12,6,12,8,12,10,12
180 DATA 6,14,8,14
```

Program I

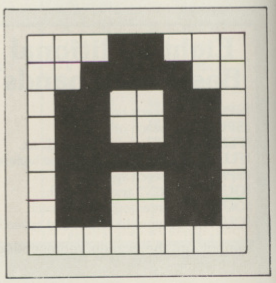


Figure I: Bit pattern for A

special set of control characters. They are often referred to as non-printable characters, as they don't usually cause any character to appear on the screen.

A useful example of one of these control characters is the one with code number 7. Try entering:

```
PRINT CHR$(7)
```

You won't see the character, but you will hear it! This character causes a short beep to be generated via the computer's internal loudspeaker. Hence the term control characters – they control what the micro does.

There are quite a few other useful control characters. Numbers 8, 9, 10 and 11 control movement of the text cursor so that:

```
PRINT CHR$(10)
```

will move the text cursor down one line, while:

```
PRINT CHR$(11)
```

moves the cursor upwards one line.

Many of the control characters have equivalent Basic statements. For instance, character number 29 will set the border colour, although it will need to be followed by two other numbers to select the particular colour. The statement:

```
PRINT CHR$(29);CHR$(3);CHR$(3)
```

is exactly the same as the command:

```
BORDER 3
```

and will set the border to red. Notice that we need to enter two colour numbers even though they are the same. Different numbers would result in a flashing border.

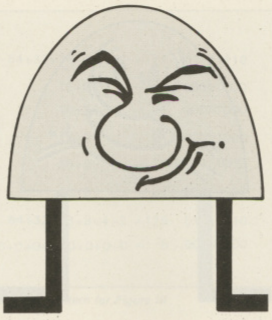
Similarly character number 4 is equivalent to the MODE command, and the following statement will select Mode 2:

```
PRINT CHR$(4);CHR$(2)
```

We don't often need to use the control characters in Basic programming, but they can be useful when they are used as part of longer character strings, as we will see next month.

Although I said that characters 0 to 31 were non-printing ones, it is possible to print a symbol representing the character. To do this we use the Ctrl key in conjunction with one of the other keys. For example pressing Ctrl and G (the seventh letter of the alphabet) at the same time will print a symbol on the screen representing the bleeper.

You can examine all of these



characters by holding down the Ctrl key and pressing each key A to Z in turn. Some of the symbols are quite clear in what they represent. The cursor movement characters 8 to 11 (Ctrl plus H to K), are represented by arrows showing the direction of cursor movement. However many of the symbols are a little bit obscure in relation to their meanings.

Notice that if we enter a command:

```
PRINT CHR$(7)
```

the control function is actually executed. In this case we hear the beep. If, however, we press Ctrl and G simultaneously the symbol is displayed but the function is not executed and no sound is heard.

We can use the Ctrl key to enter the control functions in program listings and get it to work, provided we surround it with quotes. Therefore instead of using a line like this:

```
10 PRINT CHR$(7)
```

we could achieve the same result by

```
10 REM PROGRAM II
20 MODE 1
30 x=1:y=1
40 FOR character =32 TO 126
50 LOCATE x,y
60 PEN 1
70 PRINT character;
80 PEN 2
90 PRINT CHR$(character)
100 y=y+1
110 IF y=25 THEN y=1 : x=x+6
120 NEXT
130 WHILE INKEY$="":WEND
```

Program II

using this command:

```
10 PRINT "G"
```

where G is the symbol displayed when Ctrl and G are pressed together. Now when we run the program the control function will actually be performed when line 10 is executed, but the symbol will not appear on the screen.

The second group of characters are those numbered from 32 to 126. This group represents all the letters, numbers and punctuation marks which are found on the keyboard. Program II displays all these characters together with their respective code numbers.

You can also find the characters in the User Manual.

The interesting thing about these characters is that they are a standard set which can be found on most computers.

As we have already seen, if you instruct the computer to print character number 65 it will display the letter A. If you have access to any other type of computer you may like to try printing character number 65 on that one.

You should find it will also display the letter A, but the way it is built up may not be exactly identical. However it will be recognisable as the letter A. From this you might guess that there is a standard list of numbers that correspond to letters.

This set of characters is known as the Ascii set. Ascii is short for American Standard Code for Information Interchange. As its title implies, it is a code that originated in America, but it has been generally accepted as the standard code for all alphanumeric characters on most types of computer.

You might find slight variations with some of the lesser-used symbols and punctuation marks. Being an American code, there is no representation of the British pound sign within the standard set.

To overcome this problem some British manufacturers have inserted the pound sign within the standard character set in place of one of the lesser-used characters. However Amstrad have tackled this problem differently and allocated a number outside the standard range to the pound sign.

There is a Basic keyword to represent the Ascii code, so that if we wish to find the code number given to

a particular character we can use the command:

```
PRINT ASC("A")
```

This will return the value 65, which we already know to be the code number of the letter A.

Program III is a variation on Program II, but this time it works the other way round. It prints out the Ascii value of each of the letters A to J. The letters are contained in the DATA statement in line 120.

If you wish the program to print out all of the Ascii codes, you would have to extend the DATA statement to include all the alphanumeric characters.

Alternatively you can use Program IV, which asks you to press a key and then displays the Ascii code number for that particular key. When you use this program, remember to take into account whether you have the Caps Lock key on or off. You can also make use of the Shift key to obtain the upper range of characters.

Finally we come to our third group of characters, those numbered from 127 to 255. These are a special set of symbols which are definitely non-standard. It's very unlikely that you would find an identical set of characters on any other computer.

You see, the designers of the

```
10 REM PROGRAM III
20 MODE 1
30 FOR letter=1 TO 10
40 READ letter$
50 PEN 1
60 PRINT "The ASCII code for ";
70 PRINT letter$;
80 PRINT " is ";
90 PEN 2
100 PRINT ASC(letter$)
110 NEXT
120 DATA A,B,C,D,E,F,G,H,I,J
```

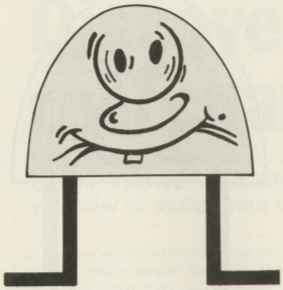
Program III

Amstrad have taken full advantage of the fact that the micro can store a total of 256 characters, and have designed an interesting range of non-standard symbols for use in our programs.

Once again you may like to examine the User Manual to see all of these characters, or alternatively you can change line 40 of Program II to read:

```
40 FOR character=127 TO 255
```

You'll see all sorts of characters



ranging from mathematical symbols to music symbols, playing-card symbols to little men. If you look carefully you may even spot the occasional Space Invader. (What

```
10 REM PROGRAM IV
20 MODE 1
30 PRINT "Press a key - followed by E
nter"
40 PRINT
50 PEN 1
60 INPUT "key ";K$
70 PEN 3
80 PRINT "The ASCII code is ";ASC(K$)
90 GOTO 50
```

Program IV

micro would be complete without one?)

Remember earlier we talked about the pound sign? This, too, is to be found within the range of special symbols as character number 163.

Many of these characters are not directly accessible from the keyboard, so to make them appear on screen it's usually necessary to use:

```
PRINT CHR$(n)
```

where n is the number of the character. Notice however that the pound sign can be accessed from the keyboard just to the left of the Clr key.

With all these characters to choose from, you would think that there would be every symbol we even need to use in our programs. However, sooner or later you'll find a use for one or more symbols that don't already exist in the built-in character set.

To overcome this problem, the programmers at Locomotive have given us the ability to design our own set of characters in addition to the internal set. In fact, any of the built-in characters may be re-designed to something completely different.

Before we start to design our own characters we need to examine in detail how any character is built up.

Remember that we said that each character is laid out on an 8 by 8 grid? Let's examine closely how the letter A is designed. We saw the layout in Figure I, and this is repeated in Figure II, but this time we have added some numbers to the drawing.

Across the top of the figure, each column is allocated a number. If you know anything about binary numbers then you should recognise this sequence of numbers. Starting from the right-hand column with number 1, each column to the left has a number double the previous one. These numbers represent the eight bits of a binary number, and any combination of these numbers can

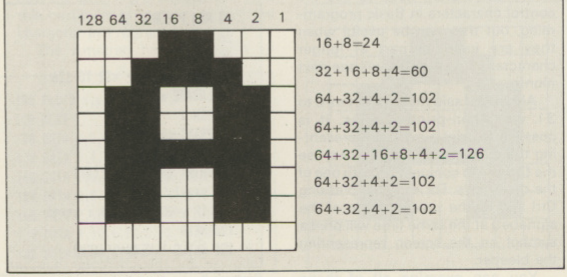


Figure II: Row values for A

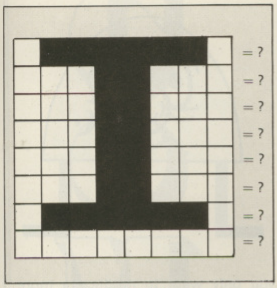


Figure III: A simple character

64+32+16+8+4+2	=126	01111110
16+8	= 24	00011000
16+8	= 24	00011000
16+8	= 24	00011000
16+8	= 24	00011000
16+8	= 24	00011000
64+32+16+8+4+2	=126	01111110
0+0+0+0+0+0+0+0	= 0	00000000

Table I: Bit pattern for Figure III

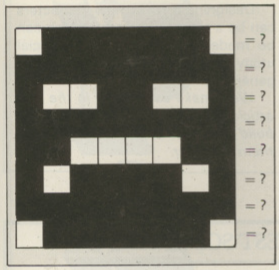


Figure IV: Alien Invader

0+64+32+16+8+4+2+0=126	01111110
128+64+32+16+8+4+2+1=255	11111111
128+ 0+ 0+16+8+0+0+1=153	10011001
128+64+32+16+8+4+2+1=255	11111111
128+64+ 0+ 0+0+0+2+1=195	11000011
128+ 0+32+16+8+4+0+1=189	10111101
128+64+32+16+8+4+2+1=255	11111111
0+64+32+16+8+4+2+0=126	01111110

Table II: Bit pattern for Figure IV

represent a number from 0 to 255.

In the top row of Figure II, the squares in columns 16 and 8 are plotted. Now if we add these two column numbers together, we get 24. The number 24 represented in binary is:

00011000

You can see that the numbers 0 and 1 are in exactly the same positions as the white and black squares in the top row of Figure II. In other words the 0s and 1s of the binary number mirror the structure of the rows. A 1 means that that column is plotted, a 0 that it is left untouched.

The same rule applies to each of the eight rows of Figure II. We can add up the column values of all the black squares in each row, and this gives us the number shown at the right hand side of the figure.

To see if you have understood this, you might like to try calculating the values of some other characters. Figure III (the letter I) is an easy one to start off with, but Figure IV (our Space Invader friend) is a little more complex.

You'll need to get the hang of calculating these values if you want to design your own characters. Tables I and II give the answers.

The Amstrad has already got the maximum 256 characters built in, so how can we accommodate any new characters? Well, the computer initially allows up to 16 characters to be redefined and these are numbered from 240 to 255. If we redesign character number 240 then the new symbol simply overwrites the existing symbol stored at number 240.

Let's try designing a new character now. First of all enter the command:

```
PRINT CHR$(240)
```

The micro will now display the original character numbered 240, which is in fact an upward-pointing arrow. To define a new character, we use the SYMBOL statement which is followed by the character number, which in turn is followed by eight numbers each representing the value of each row of the character.

Try entering:

```
SYMBOL 240,255,255,255,
255,255,255,255,255
```

followed by:

```
PRINT CHR$(240)
```

Now instead of the upward pointing arrow we should have a solid square displayed. You see we have used the maximum value of 255 in each row of the character, which results in all of the 64 squares in the 8 by 8 grid being plotted.

You might like to try designing a character of your own now. Remember you may use any character number from 240 to 255.

Try following the SYMBOL command with eight completely random numbers after the character number, between 0 and 255. Alternatively try to work out a sequence of numbers that will produce a recognisable symbol.

You may find a sheet of squared graph paper useful in helping you to design your characters.

Program V allows you to enter eight variable numbers which are then used to design a new character. In this program you will actually see each line of the symbol displayed as each number is entered.

The SYMBOL command works equally well whichever screen mode you are using. Although Mode 0 actually displays the characters over a wider area than Mode 1 and therefore uses more pixels, the symbol is still designed using the same numbering scheme.

Try changing line 20 in Program V to Mode 0 and you will see that the

```
10 REM PROGRAM V
20 MODE 1
30 SYMBOL 240,0,0,0,0,0,0,0,0
40 FOR number=1 TO 8
50 LOCATE 1,number
60 INPUT"Number ";n(number)
70 SYMBOL 240,n(1),n(2),n(3),n(4),n(5
),n(6),n(7),n(8)
80 NEXT
90 LOCATE 20,5
100 PRINT CHR$(240)
110 LOCATE 1,20
120 INPUT "Another one ";y$
130 IF UPPER$(y$)="Y" THEN GOTO 20
```

Program V

program works just as well, but of course the characters appear wider than before.

As we mentioned earlier, the Amstrad will allow up to 16 new characters to be defined, but this is only the default setting. If you find that 16 are not enough for your needs then the default setting can be changed.

To do this we use the statement:

```
SYMBOL AFTER n
```

where *n* is the number from where you wish to start redefining your characters. We can, in fact, redefine all the printable characters from 32 up to 255.

It is unlikely that we would want to redefine all of them, but let's say we needed 64 new characters. We would need to redefine character numbers 192 to 255. To do this we would use the command:

```
SYMBOL AFTER 192
```

Now we can start redefining any character from number 192 in the usual way like this:

```
SYMBOL 192,1,2,3,4,5,6,7,8
```

Just to prove that it can be done, we'll redefine the whole character set with Program VI. Try running it and then try to use the keyboard. Can you work out what's happened?

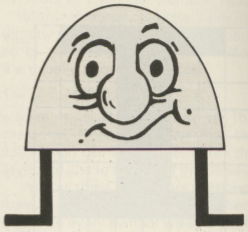
That's it for this month. As you see then, redefining some of your characters can be very useful even if you only want to produce a more complex set of friendly aliens from space.

Designing a lot of new characters can be a tedious business which can be made easier by using a special utility program called a character generator.

We'll make use of this in the next article, and we'll also see how we can build up larger characters by joining them together.

```
10 REM PROGRAM VI
20 MODE 1
25 PRINT "Wait for the beep !"
30 SYMBOL AFTER 32
40 FOR character =32 TO 255
50 SYMBOL character,255,255,255,255,2
55,255,255,255
60 NEXT
70 PRINT CHR$(7)
```

Program VI



COMPACTER

ROLAND WADDILOVE

offers a short machine code routine to speed loading a screen from tape

LOADING programs from tape can take quite a long time, running into several minutes with a lengthy listing. To relieve the boredom arising from this, commercial software often employs the use of an interesting title screen.

However, have you ever tried saving the screen to tape? Takes a long time doesn't it? Plus when you load it back again it seems to take even longer.

To solve the problem, here's a very short machine code routine that will reduce the time, and frustration, by half.

The reason it normally takes so long to load or save is that the screen display requires 16k of RAM. The whole of &C000 to &FFFF is used to store the data.

If we could reduce this in some way, then the process could be speeded up considerably.

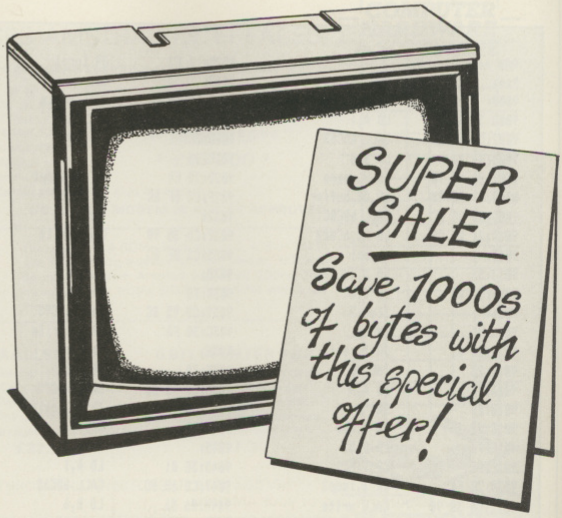
Often there are quite large plain areas of colour that have little detail. If you looked at the screen memory, then you would find a lot of similar bytes corresponding to the large plain areas.

What we could do is to get a byte from the screen memory and count how many similar ones there are. Then the byte and the count could be saved. In this way we could save two bytes instead of many.

Then, after saving the first byte and count, the process is repeated with the second, different byte, which may be 100 bytes further on, say.

No doubt some of you will have spotted that if every adjacent byte is different then the screen compact routine will actually expand it to double its normal size.

This is because one byte is required for the data and one for the



count.

However the savings with a simple screen are fantastic. It's not unusual to save 8k or more. Obviously it's going to be much quicker to save and load 7 or 8k than 16k.

Program I pokes the machine code to &9000, above HIMEM which is lowered to &9000–1. After storing the code the loader is no longer needed and can be deleted using NEW.

Listing II can be entered if you have an assembler.

To use the code, first draw your picture and set up the screen display, taking care not to scroll the screen.

Scrolling the screen will not do any harm, but when it loads back, unless the screen has been scrolled by exactly the same amount, then it will be a bit skew-whiff.

Call &9000 to save a compacted version of the screen, and Call &9045 to load it back, making sure you're in the same mode. Again it will not do any harm if you're not in the same mode. In fact, it's actually quite interesting.

Incidentally, the routine won't work for disc, unfortunately. It appears to find the end of file earlier than it should. It particularly doesn't like &1A as a screen byte. Can anyone explain why?

Still, with the speed of disc drives you won't need to worry about the size of screen you're saving, will you?

DISASSEMBLED LISTING

&9000-&900D	Open file for writing.
&9013	Get byte.
&9016-&9025	Count bytes.
&9038-&9044	Write data.
&9045-&9052	Open file for reading.
&9058-&9065	Get data and poke screen.
&906F	File name.
&9079-&9879	2k input output buffer.

```
10 REM Screen Compacter
20 REM By R.A.Waddilove
30 REM ****************************
40 REM There aren't any cassette
50 REM messages so position tape
60 REM and press keys
70 REM CALL &9000 to save screen
80 REM CALL &9045 loads it back
90 REM ****************************
100 REM(c)Computing With The Amstrad
110 MEMORY &8FFF-1
120 sum=0
130 FOR i=0 TO 116
140 READ byte$
150 POKE &9000+i,VAL("&"+byte$)
160 sum=sum+VAL("&"+byte$)
170 NEXT
180 IF sum<>14081 THEN PRINT"Error"
190 DATA 3E,01,CD,6B,BC,06,06,21,6F
200 DATA 90,11,75,90,CD,BC,BC,21,00
210 DATA C0,56,06,00,04,28,0E,23,7C
220 DATA B5,28,14,7A,BE,28,F4,CD,38
230 DATA 90,18,EC,CD,38,90,23,7C,B5
240 DATA 28,E4,C3,8F,BC,CD,38,90,C3
250 DATA 8F,BC,78,CD,95,BC,30,FA,7A
260 DATA CD,95,BC,30,FA,C9,3E,01,CD
270 DATA 6B,BC,06,06,21,6F,90,11,75
280 DATA 90,CD,77,BC,21,00,C0,CD,00
290 DATA BC,30,FB,47,CD,00,BC,30,FB
300 DATA 77,23,10,FC,CD,89,BC,38,EC
310 DATA C3,7A,BC,53,43,52,45,45,4E
```

Program I

```
RAW Assembler V.3
Pass... 2
9000:                   ORG &9000
9000:                   .save
9000:3E 01              LD A,1
9002:CD 6B BC           CALL &BC6B
9005:06 06              LD B,6
9007:21 6F 90           LD HL,name
900A:11 75 90           LD DE,buffer
900D:CD 8C BC           CALL &BC8C
9010:21 00 C0           LD HL,&C000
9013:                   .loop1
9013:56                 LD D,(HL)
9014:06 00              LD B,0
9016:                   .loop2
9016:04                 INC B
9017:28 0E              JR Z,zero
9019:23                 INC HL
901A:7C                 LD A,H
901B:B5                 OR L
901C:28 14              JR Z,end
901E:7A                 LD A,D
901F:BE                 CP (HL)
9020:28 F4              JR Z,loop2
9022:CD 38 90           CALL write
9025:18 EC              JR loop1
9027:                   .zero
9027:CD 38 90           CALL write
902A:23                 INC HL
902B:7C                 LD A,H
902C:B5                 OR L
902D:20 E4              JR NZ,loop1
902F:C3 8F BC           JP &BC8F
9032:                   .end
9032:CD 38 90           CALL write
9035:C3 8F BC           JP &BC8F
9038:                   .write
9038:78                 LD A,B
9039:CD 95 BC           CALL &BC95
903C:30 FA              JR NC,write
903E:                   .again
903E:7A                 LD A,D
903F:CD 95 BC           CALL &BC95
9042:30 FA              JR NC,again
9044:C9                 RET
9045:                   .load
9045:3E 01              LD A,1
9047:CD 6B BC           CALL &BC6B
904A:06 06              LD B,6
904C:21 6F 90           LD HL,name
904F:11 75 90           LD DE,buffer
9052:CD 77 BC           CALL &BC77
9055:21 00 C0           LD HL,&C000
9058:                   .loop4
9058:CD 80 BC           CALL &BC80
905B:30 FB              JR NC,loop4
905D:47                 LD B,A
905E:                   .tryagain
905E:CD 80 BC           CALL &BC80
9061:30 FB              JR NC,tryagain
9063:                   .loop3
9063:77                 LD (HL),A
9064:23                 INC HL
9065:10 FC              DJNZ loop3
9067:CD 89 BC           CALL &BC89
906A:38 EC              JR C,loop4
906C:C3 7A BC           JP &BC7A
906F:                   .name
906F:                   DEF$ 'SCREEN'
9075:                   .buffer
9075:                   DEFS &800
9875:                   END
```

Listing I

HERE is a simple program that will help you to keep track of your biorhythms.

As you probably know your life is ruled by your intellectual, physical and emotional states, and according to biorhythm theory they follow a cyclic pattern from the day you were born.

The intellectual cycle repeats itself every 33 days while the periods of the other two are shorter – 28 for the emotional and only 23 for the physical cycle.

When you run the program it will ask you for your name, date of birth and a month for which you want your biorhythms.

What you see on the screen is a calendar displayed in perspective with a tower on each of the days.

The tower is split into three parts. The bottom section (blue) tells you your intellectual status. The next one up (green) is your physical condition and at the top in brown you will see your emotional status. Thin is bad, fat is good.

If you are wise you will use the biorhythm calendar to plan all your activities. When you've got to do something strenuous such as watering the plants or running a marathon then try to find a day where your physical cycle is at its highest level. Otherwise you will find yourself working below your peak.

Playing chess as well as other mental activities like watching Mastermind or reading this magazine make heavy demands on your brain. So look up your intellectual level and see that it is up to the task.

As far as sex is concerned I'm not sure whether this comes under physical or emotional. Better play safe and check them both!

Occasionally you will see the towers on your calendar shrink down to just a red square.

These are the days when all your levels are at rock bottom. Anything you do will probably end in disaster.

Stay in bed and do nothing.

The program is liberally sprinkled with REMs and the structured design should be easy to follow.

The main loop starting at line 100 calls the data entry routine to ask for your name, birth date and month to display. Then comes a call to the calendar routine.

When you have finished looking at the screen you can ask for the next month (N), the previous month (P) or go back to the menu (M).

Dates are entered as 28nov44 or 1 DEC 1850 with or without spaces. They are decoded by lines 910 to 1030. *m* is set to 0 if the three lette. month name is not recognised. Years less than 100 have 1900 added.

The INSTR function in line 130 takes the key pressed converted to upper case and compares it with Space, N, P and M to produce a number 1, 2, 3, 4 or 0 if no match. The result, plus one, is used by the ON GOTO instruction.

PRINT CHR$(23);CHR$(0) in lines 400 and 1310 sets the graphics mode to opaque while CHR$(23); CHR$(3) at line 1230 sets it to OR to avoid overwriting the grid.

However CHR$(23) in line 1190 is just a trick to suppress an unwanted space.

Text printed between lines 1230 and 1310 is tagged to the graphics cursor.

VARIABLES

The DEFINT a-y instruction at line 310 makes all number variables integers except those that begin with z.

bd,bm,by,zb	Date of birth: day, month, year and number of days since 31 Dec 1BC.
dm,dy	Month to display.
name$	Your name.
mth$	An array to hold month names.
wkdy$	Weekday names.
do	Weekday of start of month displayed.
dn	Number of days in month.
wn	Number of weeks to span month.
xo,yo	Bottom left corner of calendar grid.
xg,yg	Top left corner.
xt,yt	Top left text/tower location.
t	Tower height.
d,m,y,z	Work variables for dates.
a,a$,b,b$	General work variables.

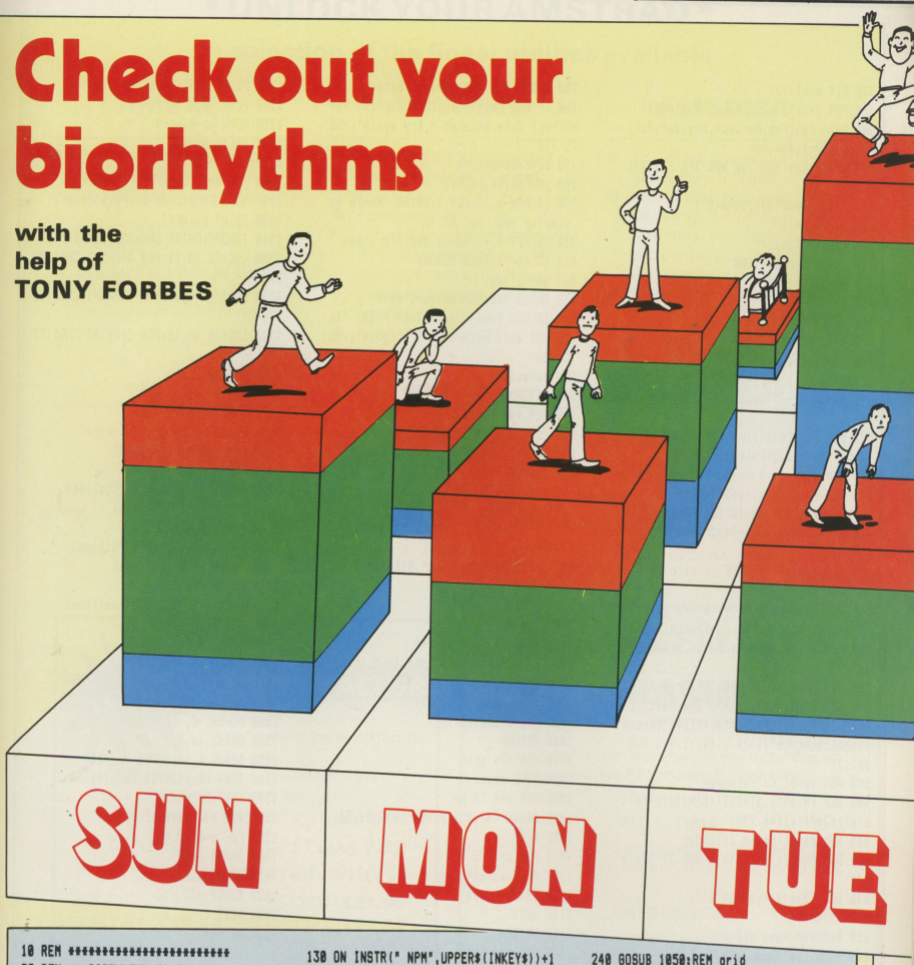
FUNCTIONS

Lines 460 to 640 define all the functions used by the program. They include some useful algorithms for messing around with dates. Include them in your own programs if you want to.

FNleap(y)	Returns –1 if *y* is a leap year, 0 otherwise. Correctly recognises ends of centuries.
FNdmth(m,y)	The trigonometry version of the Thirty days hath September rhyme. The number of days in a month is 31 plus a bit that waves about like the COSine function. February is the exception.
FNz(d,m,y)	Returns the number of days since 31 Dec 1 BC. For simplicity the Pope Gregory calendar change has been ignored. So dates before the 16th century may not give the correct answer.
FNoK(d,m,y)	Returns –1 if the date is valid.
FNwkday(z)	Returns the weekday of day number *z* (result of FNz function). 0 is Sunday, 1 Monday and so on up to 6 Saturday.
FNth$(a)	Returns the correct pair of letters to stick on the end of the number *a*.
FNlev(z,p)	Level of the period *p* cycle for day *z*.
FNxp,FNyp	Where to build a tower.

Check out your biorhythms

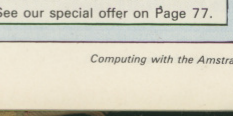
with the help of TONY FORBES




```
10 REM ************************
20 REM *  BIORHYTHM CALENDAR  *
30 REM *  ------------------  *
40 REM *    Tony Forbes       *
50 REM ************************
60 REM
70 REM(c)Computing With The Amstrad
80 REM
90 GOSUB 310:REM initialisation
100 REM *** MAIN LOOP
110 GOSUB 660:REM data entry
120 GOSUB 190:REM calendar
130 ON INSTR(" NPM",UPPER$(INKEY$))+1
 GOTO 130,130,140,160,110
140 dm=dm MOD 12+1:dy=dy-(dm=1)
150 GOTO 120
160 dm=(dm+10) MOD 12+1:dy=dy+(dm=12)
170 GOTO 120
180 REM *** calendar
190 MODE 1
200 do=FNwkday(FNz(1,dm,dy))
210 dn=FNdmth(dm,dy)
220 wn=INT((dn+do+6)/7)
230 xg=32*wn:yg=56*wn
240 GOSUB 1050:REM grid
250 xt=xo-22+xg:yt=yo-38+yg
260 GOSUB 1150:REM text
270 xt=xt+22:yt=yt+4
280 GOSUB 1370:REM towers
290 RETURN
300 REM *** initialisation
310 DEFINT a-y
320 GOSUB 470:REM functions
330 BORDER 16
340 INK 0,16:INK 1,9:INK 2,1:INK 3,3
350 RESTORE 380
```

Biorhythms

```
360 DIM mth$(12)
370 FOR a=1 TO 12:READ mth$(a):NEXT
380 DATA Jan,Feb,Mar,Apr,May,Jun,Jul,
Aug,Sep,Oct,Nov,Dec
390 wkd$="Sun Mon Tue Wed Thu Fri Sat
"
400 PRINT CHR$(23);CHR$(0);
410 xo=0:yo=16
420 name$="Tamsin"
430 bd=3:bm=5:by=1980
440 dm=4:dy=1985
450 RETURN
460 REM *** FUNCTIONS
470 REM y --> TRUE if y is leap
480 DEF FNleap(y)=y MOD 4=0 AND (y MO
D 100<>0 OR y MOD 400=0)
490 REM m,y --> number of days in mon
th
500 DEF FNdmth(m,y)=INT(31+COS(2.7*(m
-7.5))+(m>2)*(2+FNleap(y)))
510 REM d,m,y --> number of days AD
520 DEF FNz(d,m,y)=y*365+(y+3)\4-(y+9
9)\100+(y+399)\400+INT(30.401*(m-1))-
(m=2)+(m>2 AND m<8)-(m>2 AND FNleap(y
))+d
530 REM d,m,y --> TRUE if date is val
id
540 DEF FNok(d,m,y)=(y>0 AND y<=9999
 AND m>0 AND d>0 AND d<=FNdmth(m,y))
550 REM z --> weekday 0=Sun, 1=Mon, e
tc
560 DEF FNwkday(z)=z+5-INT((z+5)/7)*7
570 REM 1 --> 1st, 2 --> 2nd, etc
580 DEF FNth$(a)=STR$(a)+MID$("thstnd
rdthththththth",1-2*(a\10<>1)*(a MOD
10),2)
590 REM level in calendar
600 DEF FNlev(z,p)=INT((SIN((z/p-INT(
z/p))*2*PI)+1)*6.999)
610 REM position in calendar
620 DEF FNxp(a)=xt+64*(a MOD 7)-32*(a
\7)
630 DEF FNyp(a)=yt-56*(a\7)
640 RETURN
650 REM *** data entry
660 MODE 1:WINDOW 3,38,4,25
670 LOCATE 1,1:PEN 1:PRINT "B I O R H
 Y T H M   C A L E N D A R":PRINT STR
ING$(35,154)
680 LOCATE 1,5:PEN 2:PRINT "Enter you
r name";
690 LOCATE 4,7:GOSUB 880:REM input
700 IF a$="" THEN 780
710 name$=LEFT$(a$,20)
720 LOCATE 1,9:PEN 2:PRINT "Your date
 of birth (eg  28 nov 44)";
730 LOCATE 4,11:GOSUB 880:REM input
740 IF a$="" THEN 780
750 GOSUB 920:REM convert date
760 IF NOT FNok(d,m,y) THEN PRINT "In
correct date entered - try again";:GO
TO 730
770 bd=d:bm=m:by=y
780 zb=FNz(bd,bm,by)
790 LOCATE 1,13:PEN 2:PRINT "Month to
 display (eg  apr 85)";
800 LOCATE 4,15:GOSUB 880:REM input
810 IF a$="" THEN RETURN
820 a$="1"+a$
830 GOSUB 920:REM convert date
840 IF NOT FNok(d,m,y) THEN PRINT "In
correct date entered - try again";:GO
TO 800
850 dm=m:dy=y
860 RETURN
870 REM *** input
880 PEN 1:INPUT "====> ",a$
890 LOCATE 1,18:PEN 3:PRINT CHR$(20);
900 RETURN
910 REM *** convert date a$ --> d m y
920 d=VAL(a$)
930 a$=UPPER$(a$)
940 FOR a=2 TO LEN(a$)
950 IF INSTR("JFMASOND",MID$(a$,a,1))
>0 THEN 970
960 NEXT
970 b$=MID$(a$,a,3)
980 FOR m=12 TO 0 STEP-1
990 IF UPPER$(mth$(m))=b$ THEN 1010
1000 NEXT
1010 y=VAL(MID$(a$,a+3))
1020 y=y-1900*(y<=99)
1030 RETURN
1040 REM *** grid
1050 MOVE xo,yo
1060 FOR a=0 TO wn
1070 DRAWR 448,0,2:MOVER -448+32,56
1080 NEXT
1090 FOR a=0 TO 7
1100 MOVE xo+64*a,yo
1110 DRAWR xg,yg
1120 NEXT
1130 RETURN
1140 REM *** text
1150 PEN 1:LOCATE 1,1:PRINT "BIORHYTH
M CALENDAR"
1160 PEN 2:PRINT:PRINT name$
1170 PRINT:PRINT MID$(wkd$,4*FNwkday(
zb)+1,3)
1180 PRINT mth$(bm);FNth$(bd)
1190 PRINT CHR$(23);by
1200 WINDOW #1,33,40,21,25
1210 PEN #1,1:PRINT #1,mth$(dm);STR$(
dy)
1220 PEN #1,3:PRINT #1," N Next  P Pr
ev  M Menu";
1230 PRINT CHR$(23);CHR$(3);:TAG
1240 PLOT -8,0,1
1250 MOVE xo+8,yo-4
1260 PRINT wkd$;
1270 PLOT -8,0,3
1280 FOR d=1 TO dn
1290 MOVE FNxp(d+do-1),FNyp(d+do-1)
1300 PRINT d;:NEXT
1310 TAGOFF:PRINT CHR$(23);CHR$(0);
1320 LOCATE 40,12:PEN 3:PRINT CHR$(22
8);
1330 LOCATE 40,14:PEN 1:PRINT CHR$(22
9);
1340 LOCATE 40,16:PEN 2:PRINT CHR$(22
7);
1350 RETURN
1360 REM *** towers
1370 z=FNz(0,dm,dy)-zb
1380 FOR d=do TO do+dn-1:z=z+1
1390 MOVE FNxp(d),FNyp(d)
1400 PLOTR 0,8,2
1410 t=FNlev(z,33):REM Intellect
1420 GOSUB 1590:REM level
1430 PLOTR 0,8,1
1440 t=FNlev(z,23):REM Phisical
1450 GOSUB 1590:REM level
1460 PLOTR 0,8,3
1470 t=FNlev(z,28):REM Emotional
1480 GOSUB 1590:REM level
1490 FOR a=0 TO 3
1500 DRAWR 16,0:MOVER 0,2
1510 DRAWR -16,0:MOVER 2,2
1520 NEXT
1530 MOVER -8,-16
1540 DRAWR 16,0,0
1550 DRAWR 6,14:MOVER -6,-14
1560 DRAW FNxp(d)+16,FNyp(d)
1570 NEXT:RETURN
1580 REM *** level
1590 FOR a=1 TO t
1600 DRAWR 16,0
1610 DRAWR 6,14
1620 MOVER -22,-12
1630 NEXT
1640 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.

Machine code programming - in the RAW

ROLAND WADDILOVE takes the strain out of machine code programming with this Z80 assembler for the Amstrad

IF you have been following Mike Bibby's excellent introduction to machine code programming, then by now you are probably looking round for some sort of assembler to take all the hard work out of writing machine code programs.

As you probably know, a machine code program consists of a series of binary numbers in the range 0 to 255 (which we usually enter in hexadecimal).

This makes a program very difficult to read. What does &26, &2A mean? Very little, I should imagine, unless you know all the opcodes off by heart.

Assembly language is far easier to digest. A mnemonic is used to represent each machine code instruction. For example, the code above can be written as:

```
LD H,&2A
```

which is far more meaningful. It's not perfect, but it's a big improvement.

```
.score=&2A
LD H,score
```

is even better.

What an assembler does is to convert these assembly language mnemonics into machine code for you. There is no need to look up the individual codes and type them in as hex numbers.

You will find programs are far easier to write and far simpler to debug if they don't work first time (and they rarely do) and by using labels to represent constants and addresses machine code can become quite readable.

RAW, the assembler presented here, will allow you to write in assembly language.

When it is run, the mnemonics and labels will be converted into Z80 machine code which will be stored starting at any address not occupied by the assembler itself.

As the assembler is in Basic, it resides at the bottom of the memory, so it's convenient to place the code near the top.

HIMEM can be moved down if necessary to create space for the object (machine) code.

The assembly language program (source code) or the machine code generated by it can be saved to disc or tape, with or without the assembler program itself.

The code can afterwards be CALLed (there is no need for an assembler once the object code has been produced).

RAW will make such a difference to your programming that once you have used this assembler I can guarantee you will never do it by hand again.

All instructions and labels are placed in DATA statements before the assembler. Instructions can be 1, 2 or 3 part.

Only one instruction per line is allowed and it must be typed in upper case with one space between the first and second part (if any), and a comma between the second and third parts (if any). *No extra spaces are allowed.*

Labels must be in lower case only. For example:

RET	1 part instruction.
DJNZ loop	2 part instruction with label, 1 space.
LD A,(IX+4)	3 part instruction, 1 space, 1 comma.

Himem will usually be set to reserve space for the code. Use MEMORY as normal.

```
10 MEMORY &7FFF:REM &8000
on is available for code
```

The variable *printer* used by the assembler can be set to direct the listing to the printer on the second pass. *printer=0* turns printer off, *printer=1* turns printer on.

```
20 printer=0:REM printer
off
```

Note that there are no spaces between the label, equals, and the number.

The first instruction must be ORG:

the code is assembled at this address.

```
30 DATA  ORG &8000
```

A label must be preceded by a full-stop when it is defined.

Labels must be used for relative jumps.

A label can be used instead of a number, but must be positive.

They can be set to the current address by simply defining a label, as in line 70 of Program I.

```
10 REM PROGRAM I
20 DATA  .print=&BB5A
30 DATA  .offset=5
60 REM
70 DATA  .loop
80 DATA  LD A,(IX+offset)
90 DATA  CALL print
95 DATA  JR Z,loop
```

Program I

Alternatively they can be given a positive value (address or constant), by use of =, as in lines 40 and 50 of the same program.

Numbers can be decimal, hexadecimal binary or labels as in Program II. They must, however, be positive.

```
10 REM PROGRAM II
100 DATA  LD A,27
110 DATA  LD B,&1B
120 DATA  LD C,%11011
130 DATA  .number=27
140 DATA  LD D,number
```

Program II

Single bytes (DEFB), double bytes (DEFW) and strings (DEF$) can be placed at the current address. Space can be reserved for data (DEFS followed by the number of bytes required). See Program III for an example:

```
10 REM PROGRAM III
150 DATA  DEFB &FF
160 DATA  DEFW number
170 DATA  DEF$ 'Assembler'
190 DATA  DEFS 150
200 DATA  .number=1000
```

Program III

Comments can be included in the listing by placing them on a separate line and starting them with a ', (on the same key as 7).

Run the assembler. It will stop with an "Origin?" message.

Type AUTO and press the small Enter key. Throughout the program use the small Enter key instead of the large one.

The full-stop next to the small Enter key can be used instead if a comment is needed on the next line.

The source listing can be saved with the assembler or the assembler can be deleted and the source code saved.

The assembler can be merged with the source code when loading.

```
load "assembler"
aerge "code"
```

There is a bug in the disc filing system which sometimes prevents two programs merging successfully.

If you have any problems, try resaving one.

Alternative, one could be saved as an Ascii file.

Machine code can be saved or loaded as a binary file, for example:

```
save "code",b,&8000,&100
```

The program is quite well structured with hardly a GOTO to be seen.

Subroutines are separated by lines with a single colon for clarity.

Line numbers start at 5000 to allow room for the assembler text.

Some of the lines are very long. If they will not fit in, then you've put in some additional spaces somewhere. The program was written in Mode 2, so it makes more sense if read in that mode.

The classes of instruction are read into an array and the first part of the instruction, *text!$*, is checked against this.

If a match is found, then ON GOSUB is used to select the correct subroutine.

The names of all the labels are stored in the array *label$(50)*, and their values in *value(50)*.

As you can see from the listing, several instructions are assembled by the same subroutine. This is possible because of the similarity in the bit pattern between instructions.

BIT, RES and SET are a good example of this similarity.

The left two bits are 01, 10 and 11, and the rest are the same.

So by setting a variable the same subroutine can be used for all three.

When you have finished entering the program, test it on all the groups of instructions to make sure it is OK.

All instructions are assembled correctly and the program has been used to assemble many routines.

If it stops with an error report first check the DATA statements to see if they are correct, then the appropriate subroutine.

If you find the prospect of typing in such a long complex program too daunting, remember it is available on the monthly tape of listings.

Any errors that occur when using this are caused by an instruction being entered incorrectly.

Please note the rules above about spaces and so on. They are quite rigid!

Apart from that, you'll find RAW to be friendly, functional and extremely flexible. Happy assembling!

```
5000 REM ******* assembler ********
5010 REM By R.A.Waddilove
5020 REM(c)Computing With The Amstrad
5030 MODE 1:INK 0,0:BORDER 0
5040 GOSUB 5330:'initialise
5050 FOR pass=1 TO 2
5060 IF pass=2 AND printer=1 THEN lis
to=8
5070 CLS:RESTORE:READ text1$
5080 IF LEFT$(text1$,3)<>"ORG" THEN P
RINT "Origin ?":END
5090 x$=MID$(text1$,INSTR(text1$," ")
+1):GOSUB 7570:pc=x
5100 PRINT #listo,"RAW Assembler V.3"
:PRINT #listo:PRINT #listo,"Pass...";
pass;TAB(20);text1$:PRINT #listo
5110 WHILE text1$<>"END"
5120 PEN 1:READ text1$:textr$="":IF L
EFT$(text1$,1)="'" THEN PRINT #listo,
TAB(20)MID$(text1$,2):GOTO 5120
5130 PRINT #listo,HEX$(pc);":";:PEN 3
5140 byte1=-1:byte2=-1:byte3=-1:byte4
=-1:index=-1
5150 i%=-1
5160 FOR j%=0 TO 69
5170 IF INSTR(text1$,type$(j%))=1 THE
N i%=j%
5180 NEXT
5190 IF i%=-1 THEN PRINT:PRINT text1$
;"??":END
5200 IF i%<36 THEN ON i% GOSUB 6860,7
010,7230,6460,7110,6160,6380,6500,642
0,6420,6420,6420,6380,6380,6760,6380,
5980,6380,5980,6380,6380,6810,6060,67
```

```
60,6420,6420,6420,6420,6260,6210,5590
,6420,6420,6420,6420
5210 IF i%)35 THEN i%=i%-35:ON i% GOS
UB 6420,6380,6510,6420,6420,6110,6420
,6420,5520,5520,7130,6340,6420,6420,6
650,6380,6380,6640,6420,6670,6380,666
0,6380,6420,5490,6990,6380,7140,6600,
6620,6630,6480,6490,5420
5220 IF index)-1 THEN x=index:GOSUB 7
350
5230 IF byte1)-1 THEN x=byte1:GOSUB 7
350
5240 IF byte2)-1 THEN x=byte2:GOSUB 7
350
5250 IF byte3)-1 THEN x=byte3:GOSUB 7
350
5260 IF byte4)-1 THEN x=byte4:GOSUB 7
350
5270 PEN 2:PRINT #listo,TAB(20);textl
$;:IF textr$="" THEN PRINT #listo ELS
E PRINT #listo,",";textr$
5280 WEND
5290 NEXT pass
5300 PEN 1
5310 END
5320 :
5330 REM === initialise ===
5340 DIM type$(69),label$(50),value(5
0),cond$(7)
5350 RESTORE 7810:FOR i%=0 TO 69:READ
 type$(i%):NEXT
5360 RESTORE 7830:FOR i%=0 TO 7:READ
cond$(i%):NEXT
5370 KEY 139,CHR$(13)+"DATA  "
5380 KEY 138,CHR$(13)+"REM   "
5390 num1=0
5400 RETURN
5410 :
5420 REM === def ===
5430 IF MID$(textl$,4,1)="S" THEN x$=
MID$(textl$,6):GOSUB 7570:pc=pc+x:RET
URN
5440 IF MID$(textl$,4,1)="$" THEN FOR
 j%=INSTR(textl$,"'")+1 TO LEN(textl$
)-1:POKE pc,ASC(MID$(textl$,j%)):pc=p
c+1:NEXT:RETURN
5450 x$=MID$(textl$,INSTR(textl$," ")
+1):GOSUB 7570
5460 IF MID$(textl$,4,1)="B" THEN byt
e1=x ELSE GOSUB 7480:byte1=lb:byte2=h
b
5470 RETURN
5480 :
5490 REM === rst ===
5500 x$=MID$(textl$,5):x=VAL("&"+x$)\
8:byte1=VAL("&X11"+BIN$(x,3)+"111"):R
ETURN
5510 :
5520 REM === push/pop ===
5530 x$=RIGHT$(textl$,2)
5540 IF x$="IX" OR x$="IY" THEN index
=-&DD*(x$="IX")-&FD*(x$="IY"):x$="HL"
5550 IF x$="AF" THEN byte1=-&F5*(INST
R(textl$,"U"))0)-&F1*(INSTR(textl$,"O
")>0):RETURN
5560 GOSUB 7730:IF INSTR(textl$,"POP"
) THEN byte1=VAL("&X11"+x$+"0001") EL
SE byte1=VAL("&X11"+x$+"0101")
5570 RETURN
5580 :
5590 REM === ld ===
5600 READ textr$
5610 IF INSTR(textl$,"IX")+INSTR(text
r$,"IX") THEN index=&DD
5620 IF INSTR(textl$,"IY")+INSTR(text
```

```
r$,"IY") THEN index=&FD
5630 IF INSTR(textl$,"(")+INSTR(textr
$,"(")=0 GOTO 5810:'no brackets
5640 IF INSTR(textl$,"(") GOTO 5720:'
left bracket
5650 IF textr$="(HL)" THEN x$=RIGHT$(
textl$,1):GOSUB 7690:byte1=VAL("&X01"
+x$+"110"):RETURN
5660 x$=textr$:IF x$="(BC)" OR x$="(D
E)" THEN byte1=-&A#(x$="(BC)")-&1A#(x
$="(DE)"):RETURN
5670 IF INSTR(textl$,"HL")+INSTR(text
l$,"I") THEN byte1=&2A:x$=MID$(textr$
,2,LEN(textr$)-2):GOSUB 7570:GOSUB 74
80:byte2=lb:byte3=hb:RETURN
5680 IF INSTR(textr$,"I") THEN x$=RIG
HT$(textl$,1):GOSUB 7690:byte1=VAL("&
X01"+x$+"110"):x$=MID$(textr$,5,LEN(t
extr$)-5):GOSUB 7570:byte2=x:RETURN
5690 IF LEN(textl$)=4 THEN byte1=&3A:
x$=MID$(textr$,2,LEN(textr$)-2):GOSUB
 7570:GOSUB 7480:byte2=lb:byte3=hb:RE
TURN
5700 byte1=&ED:x$=RIGHT$(textl$,2):GO
SUB 7730:byte2=VAL("&X01"+x$+"011"):x
$=MID$(textr$,2,LEN(textr$)-2):GOSUB
 7570:GOSUB 7480:byte3=lb:byte4=hb:RE
TURN
5710 :
5720 IF LEFT$(textr$,1)="I" OR textr$
="HL" THEN byte1=&22:x$=MID$(textl$,5
,LEN(textl$)-5):GOSUB 7570:GOSUB 7480
:byte2=lb:byte3=hb:RETURN
5730 IF INSTR(textl$,"(I") THEN GOTO
5790
5740 x$=MID$(textl$,5,1):IF x$="B" OR
 x$="D" THEN byte1=-&2*(x$="B")-&12*(
x$="D"):RETURN
5750 x$=textr$:IF x$="BC" OR x$="DE"
OR x$="SP" THEN byte1=&ED:GOSUB 7730:
byte2=VAL("&X01"+x$+"0011"):x$=MID$(t
extl$,5,LEN(textl$)-5):GOSUB 7570:GOS
UB 7480:byte3=lb:byte4=hb:RETURN
5760 IF INSTR(textl$,"H")=0 THEN byte
1=&32:x$=MID$(textl$,5,LEN(textl$)-5)
:GOSUB 7570:GOSUB 7480:byte2=lb:byte3
=hb:RETURN
5770 x$=textr$:IF x$)="A" AND x$(="L"
 THEN GOSUB 7690:byte1=VAL("&X0111O"+
x$):RETURN
5780 byte1=&36:GOSUB 7570:byte2=x:RET
URN
5790 IF textr$)="A" AND textr$(="L" T
HEN x$=textr$:GOSUB 7690:byte1=VAL("&
X01110"+x$):x$=MID$(textl$,8,LEN(text
l$)-8):GOSUB 7570:byte2=x:RETURN
5800 byte1=&36:x$=MID$(textl$,8,LEN(t
extl$)-8):GOSUB 7570:byte2=x:x$=textr
$:GOSUB 7570:byte3=x:RETURN
5810 IF RIGHT$(textl$,1)="I" OR textr
$="I" THEN byte1=&ED:byte2=-&57*(textr
$="I")-&47*(textr$="A"):RETURN
5820 IF RIGHT$(textl$,1)="R" OR textr
$="R" THEN byte1=&ED:byte2=-&5F*(textr
$="R")-&4F*(textr$="A"):RETURN
5830 IF INSTR(textl$,"I") THEN byte1=
&21:x$=textr$:GOSUB 7570:GOSUB 7480:b
yte2=lb:byte3=hb:RETURN
5840 IF textr$="HL" OR LEFT$(textr$,1
)="I" THEN byte1=&F9:RETURN
5850 IF MID$(textl$,LEN(textl$)-1,1)
" " THEN x$=RIGHT$(textl$,2):GOSUB 77
30:byte1=VAL("&X00"+x$+"0001"):x$=tex
tr$:GOSUB 7570:GOSUB 7480:byte2=lb:by
te3=hb:RETURN
5860 x$=RIGHT$(textl$,1):GOSUB 7690
5870 IF textr$)="A" AND textr$(="L" T
HEN byte1=VAL("&X01"+x$):x$=textr$:GO
SUB 7690:byte1=VAL("&X"+BIN$(byte1,5)
+x$):RETURN
5880 byte1=VAL("&X00"+x$+"110"):x$=te
xtr$:GOSUB 7570:byte2=x:RETURN
5890 :
5900 REM === ex ===
5910 READ textr$
5920 IF textr$="IX" THEN index=&DD
5930 IF textr$="IY" THEN index=&FD
5940 IF INSTR(textl$,"S") THEN byte1=
&E3:RETURN
5950 IF textr$="HL" THEN byte1=&EB EL
SE byte1=&8
5960 RETURN
5970 :
5980 REM === djnz ===
5990 byte1=&10:x$=MID$(textl$,INSTR(t
extl$," ")+1):GOSUB 7570:GOSUB 7520:b
yte2=x:RETURN
6000 :
6010 REM === im ===
6020 byte1=&ED:x=VAL(RIGHT$(textl$,1)
)
6030 byte2=-&46*(x=0)-&56*(x=1)-&5E*(
x=2)
6040 RETURN
6050 :
6060 REM === in ===
6070 READ textr$
6080 IF textr$="(C)" THEN byte1=&ED:x
$=RIGHT$(textl$,1):GOSUB 7690:byte2=V
AL("&X01"+x$+"000"):RETURN
6090 byte1=&DB:x$=MID$(textr$,2,INSTR
(textr$,")")-2):GOSUB 7570:byte2=x:RE
TURN
6100 :
6110 REM === out ===
6120 READ textr$
6130 IF INSTR(textl$,"C") THEN byte1=
&ED:x$=textr$:GOSUB 7690:byte2=VAL("&
X01"+x$+"001"):RETURN
6140 byte1=&D3:x$=MID$(textl$,6,INSTR
(textl$,")")-6):GOSUB 7570:byte2=x:RE
TURN
6150 :
6160 REM === call ===
6170 x$=RIGHT$(textl$,2):GOSUB 7410
6180 IF x=-1 THEN x$=MID$(textl$,6):G
OSUB 7570:GOSUB 7480:byte1=&CD:byte2=
lb:byte3=hb:RETURN
6190 byte1=VAL("&X11"+BIN$(x,3)+"100"
):READ textr$:x$=textr$:GOSUB 7570:GO
SUB 7480:byte2=lb:byte3=hb:RETURN
6200 :
6210 REM === jr ===
6220 x$=RIGHT$(textl$,2):GOSUB 7410
6230 IF x=-1 THEN x$=MID$(textl$,4):G
OSUB 7570:GOSUB 7520:byte1=&18:byte2=
x:RETURN
6240 byte1=VAL("&X001"+BIN$(x,2)+"000
"):READ textr$:x$=textr$:GOSUB 7570:G
OSUB 7520:byte2=x:RETURN
6250 :
6260 REM === jp ===
6270 IF INSTR(textl$,"X") THEN index=
&DD
6280 IF INSTR(textl$,"Y") THEN index=
&FD
6290 IF INSTR(textl$,"(") THEN byte1=
&E9:RETURN
6300 x$=RIGHT$(textl$,2):GOSUB 7410
6310 IF x=-1 THEN x$=MID$(textl$,4):G
OSUB 7570:GOSUB 7480:byte1=&C3:byte2=
lb:byte3=hb:RETURN
6320 byte1=VAL("&X11"+BIN$(x,3)+"010"
):READ textr$:x$=textr$:GOSUB 7570:GO
SUB 7480:byte2=lb:byte3=hb:RETURN
6330 :
6340 REM === ret ===
6350 IF textl$="RET" THEN byte1=&C9 E
LSE x$=RIGHT$(textl$,2):GOSUB 7410:by
te1=VAL("&X11"+BIN$(x,3)+"000")
6360 RETURN
6370 :
6380 REM === odds & ends ===
6390 x$=textl$:byte1=-&1F*(x$="RRA")-
&F*(x$="RRCA")-&37*(x$="SCF")-&3F*(x$
="CCF")-&2F*(x$="CPL")-&27*(x$="DAA")
-&F3*(x$="DI")-&FB*(x$="EI")-&D9*(x$=
"EXX")-&76*(x$="HALT")-&17*(x$="RLA")
-&7*(x$="RLCA")
6400 RETURN
6410 :
6420 x$=textl$:x=-&A9*(x$="CPD")-&67*
(x$="RRD")-&B9*(x$="CPDR")-&A1*(x$="C
PI")-&B1*(x$="CPIR")-&AA*(x$="IND")-&
```

```
BA*(x$="INDR")-&A2*(x$="INI")-&B2*(x$
="INIR")-&A8*(x$="LDD")-&B8*(x$="LDDR
")-&A0*(x$="LDI")-&B0*(x$="LDIR")
6430 byte2=x-&44*(x$="NEG")-&B8*(x$="
OTDR")-&B3*(x$="OTIR")-&AB*(x$="OUTD"
)-&A3*(x$="OUTI")-&4D*(x$="RETI")-&45
*(x$="RETN")-&6F*(x$="RLD")
6440 byte1=&ED:RETURN
6450 :
6460 REM === and/sub/xor ===
6470 byte=&A6:GOTO 6530:'and
6480 byte=&96:GOTO 6530:'sub
6490 byte=&AE:GOTO 6530:'xor
6500 byte=&BE:textl$="CP "+MID$(textl
$,3):GOTO 6530:'cp
6510 byte=&B6:textl$="OR "+MID$(textl
$,3):GOTO 6530:'or
6520 :
6530 IF INSTR(textl$,"I") GOTO 6570
6540 IF INSTR(textl$,"HL") THEN byte1
=byte:RETURN
6550 x$=RIGHT$(textl$,1):IF (x$>="A"
AND x$<="L") AND INSTR(textl$,"(")=0
THEN GOSUB 7690:byte1=(byte AND &X111
11000)+VAL("&X"+x$):RETURN
6560 byte1=byte OR &X1000000:x$=MID$(
textl$,5):GOSUB 7570:byte2=x:RETURN
6570 IF INSTR(textl$,"IX") THEN index
=&DD ELSE index=&FD
6580 byte1=byte:x$=MID$(textl$,9,INST
R(textl$,")")-9):GOSUB 7570:byte2=x:R
ETURN
6590 :
6600 REM === sla/sra/srl/rlc/rl/rrc/r
r ===
6610 byte=&26:GOTO 6690:'sla
6620 byte=&2E:GOTO 6690:'sra
6630 byte=&3E:GOTO 6690:'srl
6640 byte=&6:GOTO 6690:'rlc
6650 byte=&16:textl$="RL "+MID$(textl
$,3):GOTO 6690:'rl
6660 byte=&E:GOTO 6690:'rrc
6670 byte=&1E:textl$="RR "+MID$(textl
$,3):GOTO 6690:'rr
6680 :
6690 IF INSTR(textl$,"I") GOTO 6730
6700 byte1=&CB
6710 IF INSTR(textl$,"HL") THEN byte2
=byte:RETURN
6720 x$=RIGHT$(textl$,1):IF x$>="A" A
ND x$<="L" THEN GOSUB 7690:byte2=(byt
e AND &X11111000)+VAL("&X"+x$):RETURN
6730 IF INSTR(textl$,"IX") THEN index
=&DD ELSE index=&FD
6740 byte1=&CB:x$=MID$(textl$,9,INSTR
(textl$,")")-9):GOSUB 7570:byte2=x:by
te3=byte:RETURN
6750 :
6760 REM === inc/dec ===
6770 IF INSTR(3,textl$,"I") THEN GOTO
 6820
6780 byte=-1*(textl$<"I")
6790 IF INSTR(textl$,"(H") THEN byte1
=&34 OR byte:RETURN
6800 IF RIGHT$(textl$,2)<"!" THEN x$=
RIGHT$(textl$,1):GOSUB 7690:byte1=byt
e OR VAL("&X00"+x$+"100"):RETURN
6810 byte=-8*(textl$<"I"):x$=RIGHT$(t
extl$,2):GOSUB 7730:byte1=byte OR VAL
("&X00"+x$+"011"):RETURN
6820 IF INSTR(textl$,"IX") THEN index
=&DD ELSE index=&FD
6830 IF INSTR(textl$,"(") THEN byte=-
1*(textl$<"I"):byte1=&34 OR byte:x$=M
ID$(textl$,9,INSTR(textl$,")")-9):GOS
UB 7570:byte2=x:RETURN
6840 byte=-8*(textl$<"I"):byte1=&23 O
R byte:RETURN
6850 :
6860 REM === label ===
6870 PEN 1:kX=-1:x$=MID$(textl$,2)
6880 IF INSTR(x$,"=")>0 THEN x$=LEFT$
(x$,INSTR(x$,"=")-1)
6890 FOR jX=0 TO num1
6900 IF x$=label$(jX) THEN kX=jX
6910 NEXT
6920 IF kX=-1 THEN kX=num1:num1=num1+
1
6930 IF num1>50 THEN PRINT:PRINT "Too
 many labels":STOP
6940 label$(kX)=x$
6950 IF INSTR(textl$,"=")=0 THEN valu
e(kX)=pc:RETURN
6960 x$=MID$(textl$,INSTR(textl$,"=")
+1):GOSUB 7570:value(kX)=x
6970 RETURN
6980 :
6990 REM === sbc/adc ===
7000 byte=&X10011110:GOTO 7020:'sbc
7010 byte=&X10001110:GOTO 7020:'adc
7020 READ textr$
7030 IF INSTR(textl$,"HL") THEN x$=te
xtr$:GOSUB 7730:byte1=&ED:byte2=VAL("
&X01"+x$+"0010")+(8 AND textl$<"S"):R
ETURN
7040 IF textr$>="A" AND textr$<="L" T
HEN byte=byte AND &X11111000:x$=textr
$:GOSUB 7690:byte1=byte+VAL("&X"+x$):
RETURN
7050 IF textr$="(HL)" THEN byte1=byte
:RETURN
7060 IF INSTR(textr$,"I") GOTO 7080
7070 byte1=byte OR &X1000000:x$=textr
$:GOSUB 7570:byte2=x:RETURN
7080 IF INSTR(textr$,"IX") THEN index
=&DD ELSE index=&FD
7090 byte1=byte:x$=MID$(textr$,5,INST
R(textr$,")")-5):GOSUB 7570:byte2=x:R
ETURN
7100 :
7110 REM === bit/set/res ===
7120 byte=&X1000000:GOTO 7150:'bit
7130 byte=&X10000000:GOTO 7150:'res
7140 byte=&X11000000:GOTO 7150:'set
7150 READ textr$:x=VAL(RIGHT$(textl$,
1)):byte2=byte+VAL("&X"+BIN$(x,3)+"00
0")
7160 IF INSTR(textr$,"I") GOTO 7200:'
ix/iy
7170 byte1=&CB
7180 IF textr$=(HL)" THEN byte2=byte
2+VAL("&X110"):RETURN
7190 x$=textr$:GOSUB 7690:byte2=byte2
+VAL("&X"+x$):RETURN
7200 IF INSTR(textr$,"IX") THEN index
=&DD ELSE index=&FD
7210 byte3=byte2+VAL("&X110"):byte1=&
CB:x$=MID$(textr$,5,INSTR(textr$,")")
-5):GOSUB 7570:byte2=x:RETURN
7220 :
7230 REM === add ===
7240 READ textr$
7250 IF INSTR(textl$,"Y") THEN index=
&FD
7260 IF INSTR(textl$,"X") THEN index=
&DD
7270 IF INSTR(textl$,"L") OR INSTR(te
xtl$,"I") THEN x$=textr$:GOSUB 7730:b
yte1=VAL("&X00"+x$+"1001"):RETURN
7280 IF textr$>="A" AND textr$<="L" T
HEN x$=textr$:GOSUB 7690:byte1=VAL("&
X10000"+x$):RETURN
7290 IF textr$="(HL)" THEN byte1=&86:
RETURN
7300 IF INSTR(textr$,"I") GOTO 7320:'
adc a,(ix/iy+n)
7310 byte1=&CE:x$=textr$:GOSUB 7570:b
yte2=x:RETURN
7320 IF INSTR(textr$,"Y") THEN index=
&FD ELSE index=&DD
7330 byte1=&86:x$=MID$(textr$,5,INSTR
(textr$,")")-5):GOSUB 7570:byte2=x:RE
TURN
7340 :
7350 REM === print hex$(x) ===
7360 POKE pc,x:pc=pc+1
7370 IF x<16 THEN PRINT #listo,"0";
7380 PRINT #listo,HEX$(x);" ";
7390 RETURN
7400 :
7410 REM === get code for condition x
$ -> x ===
7420 x=-1
7430 FOR jX=0 TO 7
```

```
7440 IF x$=cond$(j%) THEN x=j%
7450 NEXT
7460 RETURN
7470 :
7480 REM === x -> lowbyte,hibyte ===
7490 lb=x-256*INT(x/256):hb=INT(x/256)
7500 RETURN
7510 :
7520 REM === calculate jump ===
7530 IF x>255 THEN x=x-pc-2:IF x>255
THEN PRINT text1$;" Too far":END
7540 IF x<0 THEN x=x+256
7550 RETURN
7560 :
7570 REM === x$,number/label -> x,num
ber ===
7580 WHILE x$<"!":x$=MID$(x$,2):WEND
7590 IF LEFT$(x$,1)="&" THEN x=VAL(x$
)-65536*(VAL(x$)<0):RETURN
7600 IF LEFT$(x$,1)>="0" AND LEFT$(x$
,1)<="9" THEN x=VAL(x$):RETURN
7610 IF LEFT$(x$,1)="%" THEN x=VAL("&
X"+MID$(x$,2)):RETURN
7620 IF pass=2 THEN x=-1 ELSE x=0:RET
URN
7630 FOR j%=0 TO num
7640 IF label$(j%)=x$ THEN x=value(j%
)
7650 NEXT
7660 IF x<0 THEN PRINT text1$;" Label
??":END
7670 RETURN
7680 :
7690 REM === x$,register -> x$,binary
 code ===
7700 x$=BIN$(ASC(x$)-66-8*(x$="A")+2*
(x$="H")+5*(x$="L"),3)
7710 RETURN
7720 :
7730 REM === x$,register pair -> x$,b
inary code ===
7740 IF x$="BC" THEN x$="00"
7750 IF x$="DE" THEN x$="01"
7760 IF x$="HL" OR x$="IX" OR x$="IY"
 THEN x$="10"
7770 IF x$="SP" THEN x$="11"
7780 RETURN
7790 :
7800 REM === classes ===
7810 DATA END,,,ADC,ADD,AND,BIT,CALL,
CCF,CP,CPD,CPDR,CPI,CPIR,CPL,DAA,DEC,
DI,DJNZ,EI,EX,EXX,HALT,IM,IN,INC,IND,
INDR,INI,INIR,JP,JR,LD,LDD,LDDR,LDI,L
DIR,NEG,NOP,OR,OTDR,OTIR
7820 DATA OUT,OUTD,OUTI,POP,PUSH,RES,
RET,RETI,RETN,RL,RLA,RLCA,RLC,RLD,RR,
RRA,RRC,RRCA,RRD,RST,SBC,SCF,SET,SLA,
SRA,SRL,SUB,XOR,DEF
7830 DATA "NZ","Z","NC","C","PO","P
E","P","M"
7840 REM ****************************
*
7850 MODE 1
```

Give your fingers a rest . . .

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

See our special offer on Page 77.

Lateral thinking might help to get round those Basic bugs

ROLAND WADDILOVE looks at ways of coping with Amstrad Basic's peculiarities

THE manual supplied with the Amstrad is a fair introduction for the novice, apart from a few minor errors. But the more advanced programmer will require a little bit more information than is provided within its pages.

While programming I have come across several tricky problems which caused quite a few headaches. From the letters received it is clear that many other programmers are also having difficulty, so I would like to share my experience and save a lot of frustration.

I hope to cover some of the problems encountered and their solutions, without delving into complicated machine code. It often requires a bit of lateral thinking, as it is usually possible to do what you want to do, but you have to do it in a slightly different way.

There are several short listings to demonstrate each problem and its solution. Each program assumes that you are in Mode 1 and that the Amstrad has just been switched on or reset with Ctrl+Shift+Esc.

The first problem I encountered was how to set the internal clock. Every micro has an internal clock which can be accessed in one way or another. Reading it is easy:

```
PRINT TIME
```

or:

```
t=TIME:PRINT t
```

will tell you the time elapsed in 300ths of a second since the machine was switched on, or reset with Ctrl+Shift+Esc.

Setting it is not quite so simple. Try:

```
TIME=0
```

or:

```
t=0:TIME=t
```

and the result is "Syntax error". The problem is that TIME is not a variable, but a function, and can only return a value. It's no use trying to set it, as it does not make sense.

The clock must be situated somewhere in the 65536 bytes of RAM, so if we knew where it was we could POKE the address with whatever value we wanted.

As the clock ticks pretty fast – 300 times a second – then 1 byte (which can store a number between 0 and 255), obviously would be of little use to keep track of the time. Even two bytes (0-65535) would only be sufficient to last just over three minutes, so the minimum is three bytes which would allow the clock to run for about 15 hours before it got round to zero again.

So how can it be found? Program I will look for it and print a list of the most likely addresses. It takes a long time to scan through 64k, so be prepared for a long wait.

```
10 REM PROGRAM I
20 FOR i=0 TO 65533
30 IF ABS(TIME-(PEEK(i)+256*PEEK(i+1)
+256*256*PEEK(i+2))<2 THEN PRINT "Ti
me is at";i
40 NEXT
```

Program I

It works by looking at three consecutive bytes of memory and comparing their value with TIME. If they are very close then the address is printed out.

It is quite likely that more than one address will be printed out – the correct one plus a couple of others that are the same by pure coincidence. Run the program again and the true address will be printed again. The others will now be different.

You should find the address at which TIME is stored to be 45447, or &B187 in hexadecimal. These three bytes can be poked with any value you choose to set TIME.

In actual fact TIME is a four byte integer, but unless you leave your micro running for more than 15 hours without resetting it the fourth byte will not be used.

Program II will set the clock to any value input between 0 and about 16

```
10 REM PROGRAM II
20 CLS
30 INPUT "Time=";number
40 byte3=INT(number/(256*256))
50 byte2=INT((number-256*256*byte3)/2
56)
60 byte1=number-256*byte2-256*256*byt
e3
70 POKE 45447,byte1
80 POKE 45448,byte2
90 POKE 45449,byte3
100 WHILE NOT bored
110 LOCATE 10,10
120 PRINT TIME
130 WEND
```

Program II

million. It does this by POKEing the time to 45447-45449. As each byte can only store a number up to 255, byte 1=1s, byte 2=256s and byte 3=256*256s.

Another feature/bug is that a space is printed before and after every

number. This can be a nuisance if you want to print a number in the middle of something or in a tight space as the character immediately to the left and right are erased.

Run Program III and you will see the problem. It attempts to print a

```
10 REM PROGRAM III
20 INPUT "Number";n
30 PRINT
40 PRINT "*****";n;"*****"
```

Program III

number in the middle of a line of asterisks, and fails.

By converting the number to a string the spaces can be removed. Curiously STR$(n) still has the leading space if it is positive, but not if it is negative. This must be checked for.

Program IV will print the number correctly by calling a subroutine at 9000. This could be added to any program. Just set *n* equal to the number and GOSUB 9000 to print it.

```
10 REM PROGRAM IV
20 INPUT "Number";n
30 PRINT
40 PRINT "*****";
50 GOSUB 9000
60 PRINT "*****"
70 END
9000 REM Print number (n)
9010 IF n<0 THEN PRINT STR$(n); ELSE
PRINT MID$(STR$(n),2);
9020 RETURN
```

Program IV

Another bug/feature that can cause problems is word wrap when printing strings. If you try to print a string half way along a line and it is too long to fit then it will be printed at the start of the next line.

What should happen is that the string is printed at the correct position and any part that will not fit goes on the next line. Program V illustrates the problem.

The solution is to print each character of the string separately. Program VI shows how this may be done. A subroutine at 9000 will print the string correctly.

Before any input is requested by a program it is always best to clear the

```
10 REM PROGRAM V
20 CLS
30 a$="This is a very long string"
40 FOR i=10 TO 20
50 LOCATE i,i
60 PRINT a$
70 NEXT
80 END
9000 REM Print string (a$)
9010 FOR n=1 TO LEN(a$)
9020 PRINT MID$(a$,n,1);
9030 NEXT
9040 RETURN
```

Program V

```
10 REM PROGRAM VI
20 CLS
30 a$="This is a very long string"
40 FOR i=10 TO 20
50 LOCATE i,i
60 GOSUB 9000
70 NEXT
80 END
9000 REM Print string (a$)
9010 FOR n=1 TO LEN(a$)
9020 PRINT MID$(a$,n,1);
9030 NEXT
9040 RETURN
```

Program VI

input buffer. This is an area of memory set aside by the CPC464 to make a note of any keys pressed since the last input. It constantly checks the keyboard and places any keys pressed in the buffer.

During a game many keys may be pressed so the buffer fills up with

"There are no commands for setting the graphics pen and paper"

garbage. This must be removed or when an input is asked – say for the high score table – the garbage will be accepted instead.

A simple:

```
WHILE INKEY$<>"":WEND
```

will remove all characters from the buffer allowing you to ask for an input or a key to be pressed.

Printing at the graphics cursor using TAG allows characters to be moved very smoothly around the screen.

Program VII demonstrates this:

```
10 REM PROGRAM VII
20 MODE 0
30 PEN 4:PAPER 5:CLS
40 TAG
50 FOR x=0 TO 640 STEP 8
60 MOVE x,200:PRINT "O";
70 FOR t=0 TO 100:NEXT
80 MOVE x,200:PRINT " ";
90 NEXT
100 TAGOFF
```

Program VII

Line 30 sets the pen and paper, 40 directs all further printing to the graphics cursor and lines 60-100 move the letter O across the screen. Run it and watch what happens.

Why is the letter O yellow when we set the pen to 4 which is white, and why is a blue stripe printed across the screen?

Try altering line 30 and you will quickly discover that the colour of the O is unaffected by altering the pen or paper colour. What is wrong?

What is happening is that the O is being printed in the graphics foreground pen on top of the graphics background paper – a yellow O on a blue background.

We need some way of setting the graphics colours, but the problem is that there are no commands for setting the graphics pen and paper. What we can do, though, is PLOT a point in any colour we choose or clear the graphics window to any pen.

Add these four lines to Program VII to make Program VIII:

```
23 PLOT -100,-100,4
24 ORIGIN 0,0,1000,2000,3000,4000
25 CLG 5
26 ORIGIN 0,0,0,640,0,400
```

Now the O is printed in the correct colours, white on a black background.

What we have done is to set the graphics foreground colour by plotting a point with pen 4 in line 23. You can't actually see this point as it is way off the screen, –100,–100. Line 24 sets up a graphics window which is also off the screen which is then ▶

cleared to pen 5 and restored to its original position.

The important point is that the

```
10 REM PROGRAM VIII
20 MODE 0
23 PLOT -100,-100,4
24 ORIGIN 0,0,1000,2000,3000,4000
25 CLG 5
26 ORIGIN 0,0,0,640,0,400
30 PEN 4:PAPER 5:CLS
40 TAG
50 FOR x=0 TO 640 STEP 8
60 MOVE x,200:PRINT "O";
70 FOR t=0 TO 100:NEXT
80 MOVE x,200:PRINT " ";
90 NEXT
100 TAGOFF
```

Program VIII

graphics colours are set but the screen remains unchanged by these additional lines.

One problem that can't be overcome is the lack of a "No such variable" error. This is a bug/feature of the Basic. If Amstrad Basic comes across a variable that does not exist then a new variable with that name is created and set to zero.

"One problem that can't be overcome is the lack of a "No such variable" error"

Typing errors are inevitable when entering long programs, but most Basics can be relied on to point out the spelling mistakes.

Not Amstrad Basic though – it will sail through creating new variables all over the place, which makes debugging quite difficult.

Suppose you typed:

```
area=length*with
```

instead of:

```
area=length*width
```

Amstrad Basic would create a new variable *with* and set it to zero. It would not cause an error to be reported.

This would make *area* equal to zero no matter what the length or width was. It can be quite hard to spot this type of mistake, so extra care must be taken over spelling.

These are by no means the only problems likely to be encountered, but by working through the solutions I hope I have given you a few ideas about how to solve them.

If you find that something is not working as expected, or cannot be done straightforwardly then try to do it in a slightly different way.

Look at it from a different angle and use a bit of lateral thinking!

Bewitched by all this Sorcery

I THINK I am a victim of Sorcery – the game, that is. The saga runs as follows:

Tape No. 1 was bought from Boots at Worcester. Side 1 loaded OK and I played the game four times – and a superb game it is too.

Side 2 wouldn't load – it crashed shortly after the "filling in" lines started to appear, though not always at the same stage.

At the fifth session side 1 also gave up loading – persistently crashing at any arbitrary stage of loading, sometimes immediately and once right at the end.

I went back to Boots and the tape was replaced without question. The delightful lady "Computer Consultant" suggested that I try the replacement on their demo computer before taking it away. So I did.

Neither side would load. She gave me another to try – same again. I tried all their stock of about 10 tapes and none would load.

Crashing occurred always near the start of loading. They nor I could understand the problem. Boots has sold lots and had none back so far. Anyway, I took another home, just in case, but no luck. That was No. 2.

Boots having refunded my money, I went next door to Greens at Debenhams and bought tape No. 3. It loaded perfectly on their Amstrad – both sides.

Got it home, and guess what? Yes, crashing all over the place. Getting monotonous, isn't it?

Greens gave me a replacement. There seemed no point in a shop test, so I tried it at home. Side 1 went all the way through the loading until the tape came to an end – but the screen stayed on "Sorcery is Flashloading" and stuck there.

Tried it again – crashed at the beginning! Tried side 2 – same again.

Nothing I do will get it to load, though the point at which the crashing occurs varies – although always near the start.

I have the tape here. There is no address on it, or the inlay to which I can complain, hence bending your ear.

My Amstrad loads without difficulty everything else I put in, so have I hit on a bad batch of tapes, or has the necromancer finally triumphed?

Who can I turn to? – **T.J. Knowles, Worcester.**

■ Sorcery uses a very high baud rate to speed up loading, which may cause problems with some cassette decks.

Interceptor's azimuth setting kit may help.

Egg Blitz scores

CAN you tell me if there was a bug in the Egg Blitz listing (April issue) to do with the score?

The problem is that when the score reaches 100 it prints 00 instead. Even the high score will not show more than 99.

The "bug" may be due to my typing, but if so I am unable to find it.

Due to my being an absolute beginner in computing and the program being machine code I can't follow the listing and understand more than a small percentage of it.

Secondly, can you recommend a printer that will produce a reasonably good copy and can cope with a few graphics? As money is in very short supply I am looking for a (budget) cheap one until I have saved enough for the likes of the Texas Kagan or Epson RX80.

Thirdly – a tip I read in a magazine and most of your readers appear to have missed. It may help someone out there.

To get the Amstrad to erase a tape for re-use, it is necessary to fool the computer into running the cassette.

Type in CAT, place tape to be erased in the cassette deck, rewind to point you wish to start at, press ENTER then RECORD and PLAY (instead of just PLAY).

This will cause the computer to run the cassette and, as the record head is in contact with the tape and no signal is being recorded, the result is a blank tape.

I've used this method many times with no problems.

Lastly, may I thank you all for an excellent magazine. – **R. Holden, Pontefract, Yorks.**

■ How on earth have you scored more than 99? We can't get past the second screen and can't get anywhere near this. However, we can assure you that there is no bug in the program.

In the long run it would be cheaper to go for a good printer first rather than a cheap one, then an expensive one.

Short of pounds

THANK you for the February Computing with the Amstrad monthly tape. I particularly like the word processor program I'm using here.

Can you please let me know whether I have a faulty tape or whether it is a shortcoming of the program that the pound sterling sign will not print?

I'm not complaining about value for money, frankly I'm amazed at what one gets for so little. – **Ian Yates, Lochinver, Sutherland.**

■ The £ is difficult to print as some printers use one Ascii code and others a different one.

The Ascii code for £ on the CPC464 is 163, which is non-standard and can't be sent to the printer.

This being the case, it's very difficult to give a general answer.

Access code

LAST October I bought a CPC464 Amstrad colour computer – because I have a lot of catalogues and price lists which I'm in the habit of misplacing (and had stolen).

So I now have the entire stock on tape.

What I am trying to do is get an access code so that only I can see what is on the tapes.

I have succeeded in so much as when the Input-Run the computer asks for the access code, and on inputting this the contents are revealed.

However, should anyone simply input "LIST", then they have the entire contents of my tapes at their disposal. Can you help? – **E.J. Cannasso, Merthyr Tydfil.**

■ The simple answer is to incorporate the following lines into the start of your program:

```
1 REM***
```

then in direct mode type in POKE &176,255. This prevents your program being listed by poking the middle * with a control character.

After line 1, you can use any INPUT "code" routine you like and no one will be able to see it.

Bargain?

CAN I get a printer around about £200 that will work on my Commodore 64 and Amstrad CPC464?

Also is there a disc drive

that will work on both machines? – **Mr Kay, London.**

■ Commodore and Amstrad peripherals are not generally compatible, so would need additional hardware and software – which probably isn't available.

Thanks!

MANY thanks, Al, for the return of my Texted cassette and your remarks.

I'm pleased to say the program is now working perfectly. But is my face red!

I really should have noticed the option$/options $ lash-up after four or five times of checking the listing, particularly as these were the lines I looked at intently. And the missing space in line 710. There is no excuse.

Thanks again for the lesson. – **Anon.**

THANK you, Al, for replying so quickly to my letter with problem over Digger.

However, as you have stated that there are no errors in the printing, as I had suspected, the fault is obviously one of my own and I would like to persevere to see if I can find it.

Your hints in magazine No. 4 will be useful and if we had bought that beforehand I would not have troubled you at all.

I will start listing carefully – and breaking long lines down as you suggest – in the hope that I can sort it out for myself!

I know it is only a program for "fun", but I just don't want to be defeated! – **Anon.**

□ We don't provide a debugging service as a matter of course, but occasionally Al has tried to help out. This not only assists our readers, but has also helped Al with his debugging hints.

We are always willing to help anyone with programming queries, but you must send a SAE if you want a reply.

Working Adder

IN the April edition of Computing with the Amstrad there was a letter from Cyril Bellis which said that he had typed in the first three programs and they didn't work.

But I did Mad Adder and it worked perfectly, so he must have made some errors in the program. – **Leigh Nolan, Poynton, Cheshire.**

Disabled Break

LET me first say how much I enjoy reading your interesting articles.

In the June edition, Mr Rogers asked for a simple way to disable the Break key. This is easily done by using the KEY DEF command.

```
KEY DEF 66,1,&00 - turns
break off
KEY DEF 66,1,&FC - turns
break on
```

However, if you wish to use the ON BREAK GOSUB command, then you must indeed use INKEY$ instead of INPUT. To aid with this I enclose a small routine.

Before calling, 1n% is set to the number of characters to be input and on exit the variable res$ contains the given string.

Line 60400 is used to select which Ascii characters the routine will except and can be altered to suit.

The call in line 61100 resets the event queue, otherwise repeated Breaks disable the ON BREAK GOSUB statement – I don't know why, perhaps someone can tell me.

Finally, if you can publish my full address I would be glad to hear from any Amstrad users in my area. – **T.K. Jones, 4, Marbury Close, Moorside, Sunderland SR3 2QG.**

```
10 MODE 2:ON BREAK GOSUB 61
100
20 ln%=5:LOCATE 10,10:PRINT
"Please enter five characte
r string ";:GOSUB 60200:IF
LEN(res$)<>ln% THEN LOCATE
48,10:PRINT SPC(5);CHR$(7):
GOTO 20
30 LOCATE 10,12:PRINT "The
string you gave was : ";res
$
40 END
60000 REM*****************
*****
60100 REM*******T.K.Jones**
****
60200 PRINT "? ";:ky$="":re
s$=""
60300 PRINT CHR$(143);:ky$=
INKEY$:PRINT CHR$(8);:IF ky
$="" THEN 60300 ELSE IF ASC
(ky$)=13 THEN 60800
60400 IF ASC(ky$)<32 OR ASC
(ky$)>127 THEN 60300
60500 IF ASC(ky$)<>127 THEN
 60700 ELSE IF LEN(rse$)=0
THEN 60300
60600 IF LEN(res$)>1 THEN r
es$=LEFT$(res$,(LEN(res$)-1
));PRINT SPC(1);CHR$(8);CHR
$(8);SPC(1);CHR$(8);:ky$=""
:res$=""
60700 PRINT ky$;:res$=res$+
ky$:IF LEN(res$)=ln% THEN 6
0800 ELSE GOTO 60300
60800 PRINT SPC(1):RETURN
60900 REM*****************
*******
61000 REM*****************
*******
61100 ENT 1,200,100,10:SOUN
D 1,800,40,7,0,1,0:CALL &BC
C8:RETURN
```

Educational software

AS a mature student attempting to master A Level Computer Science one evening a week, I would like to add my comments to your remarks regarding the Research Machines 380/480Z as a very important source of educational software.

The A Level syllabus requires that a project is completed in a structured language and the one chosen by my college is SBAS for the 380/480Z by Software Production Associates.

They inform me that this is not available for the CPC464 and that they see no possibility of producing a version to run on it.

So although the resident Basic is excellent, it is not suitable for my A Level project and when one is only attending college one evening a week the project really needs to be worked on at home.

I have bought Hisoft's Pascal but I am struggling to learn this on my own.

A further area where an implementation of a good point on the 380/480Z computers is the front panel mode which shows the operation of the Z80's registers as well as the memory locations when running machine code problems. I have a machine code monitor but this doesn't show the registers, which are important in seeing what goes on.

So in the field of higher education there are many programs designed for the 380/480Z computers which would be very useful for students to advance their knowledge at home.

I consider the magazine an excellent one and hope that your educational column will expand to include larger articles and contain programs for our use.

You may also be able to influence companies to change their minds about converting their educational software to run on the excellent CPC464.

This letter, by the way, is written with the aid of the super Mini Office word processor, which is educational in its own right. Keep up the good work. – **Cliff Wright, Sale, Cheshire.**

Restore ASN, ACS

I HAVE had my CPC464 for three months now and I am delighted at its quality and

speed (compared to, dare I say it, my ZX81).

I am, however, amazed at the lack of two simple functions on the Amstrad and by the fact that nobody either on your staff or, apparently, among the users has discovered these.

The functions missing are: ASN (arcsine) and ACS (arccos). Only ATN (arctan) is present.

For the majority of non-technical users this is OK, but in geometrical and scientific use – especially if Mr Sugar wants to break into the schools market – they are essential.

To overcome this lack you will have to use the "t formulae", relating ASN and ACS to ATN.

The following subroutines use these to find ASN & ACS:

```
999 REM *** phi=ASN(S) ***
1000 IF S=0 THEN phi=0:RETU
RN
1010 t=(1-SQR(1-S*S))/S:phi
=2*ATN(t):RETURN
1099 REM *** phi=ACS(C) ***
1100 IF C=-1 THEN phi=180:R
ETURN
1110 t=SQR((1-C)/(1+C)):phi
=2*ATN(t):RETURN
```

These return ASN in the range –90 to +90 degrees and ACS from 0 to 180 degrees.

I can only assume that the Locomotive authors had something like this in mind and I hope these are of use to your readers. – **Ian Townsend, Newhaven, Sussex.**

Bug tag

THANK you for printing my last letter (Tag Colour, April). After putting my letter in the post I realised that there is a bug in the routine.

Instead of making space above HIMEM and storing the routine in that space, the routine made space above HIMEM and then stored the routine above the old value of it, overwriting whatever was there before.

The solution to this problem is to change line 10 to:

```
10 h = HIMEM-&1MEMORY h
```

Incidentally, why doesn't Amstrad deliver hardware in Bristol? The two Comet branches nearest me have been waiting since Christmas for deliveries and other shops haven't got a clue about when they will get deliveries. – **Philip Marson, Bristol.**

□ Thanks for the quick response. We didn't notice any bug as the program worked fine when it was checked.

However, if anyone is having any problems you now have the answer.

Printer plotters

THANK you so much for your magazine – it is quite excellent.

However, if I may, I would like to suggest two small points which you may like to consider.

It would be rather nice if you could include in the Software Survey section, photographs of the actual computer graphics – rather than that of the cassette case.

Similarly it would also be useful to have a score table(s) in this section illustrating various aspects of each software and giving ratings of, for example, graphics, playability, etc.

The game listings I have typed in have been quite good (graphically) – but somewhat slow since most are written in Basic.

It would be much appreciated if you could include a few more – perhaps in fast action machine code? This would also add more volume to the magazine.

Incidentally would you offer me some advice on printer plotters? What are their disadvantages – apart from their relatively slow speed – and are there any (A4 size) low-cost type available for the Amstrad?

Incidentally I typed in the "mode" program on Page 69 (issue 5), but it caused my computer to scroll frantically until I changed the number "18" in line 30 to "17". It's excellent. – **George Usch, Beeston, Nottingham.**

■ As you can see in this issue, we've started printing screen shots of our games listings, but commercial software can be difficult to freeze to take a photo.

Machine code listings tend to cause huge problems in debugging. And, if we provided assembly listings they'd take up the whole magazine.

Has any Amstrad user had any experience of printer plotters?

Off-line woes

AAARGH! The frustration of it all!

I'm writing in the context of a Micronet 800 subscriber who has been off-line for almost five months when the Spectrum had to be sold to balance an Amstrad disc drive.

In these five months I have been tracking down manufacturers of serial interfaces and modems, inquiring about communication packages, etc. and was just about to commit myself to a hotch-potch system from assorted sources...

Also in the five months I realised just how much I missed the communications side of my hobby.

Now I see you have the answer to my isolation (June edition, Page 13) – but I must suffer yet one more agonisingly-long month at least to find the details! GROAN!

Please spare me the waiting, the queueing, the rush of applications and orders, and delivery delays!!!... I need no Prestel application forms, no jack socket fitting ... it is sitting there on the wall looking forlorn and lonely, pining for a friendly modem to be fitted..., and Prestel keep on billing me for services I can no longer access!

Just send the works – hardware and software! – **Paul A. Bootes, Sheffield.**

■ If you take a look at our special offer on Page 9 you'll see that we've solved the communications gap.

Flush the buff

TO put readers out of their agony, they can flush the keyboard buffer simply by using the following command:

```
10 CALL &BFF9,&C9,&1CED,&CF
00
```

This will execute the keyboard flush routine at address &1CED in the lower ROM.

Any routine can be called this way from Basic in any ROM by making the second parameter the address and making the third &CF00 for the lower ROM or &D700 for the upper ROM.

I will leave you to work out how it does it and to consider

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

Grow a better tree

I WOULD like to say how much I enjoy your magazine. However I must say that Trevor Roberts' article on trees (actually the program) was little more than a beginner's attempt.

As he points out in his diagram, he is trying to produce a binary tree diagram. This is not the case when he gets down to the actual program.

He points out that it is tricky. But his branches merge and overlap, giving people, perhaps, the wrong idea of what a binary tree really looks like.

I have enclosed a program that will draw a true representation of a binary tree using a recursive method.

Due to the intricacies of integer divide, that is, just taking the left side of the decimal point, when the dimensions are multiplied they do not match with previous values.

This means that to get a tree with more levels you would have to extend the initial values, but the tree would go off the screen.

Mr Roberts also said his was inefficient because of the store used by his line 40.

```
40 DIM x(maxlevel,2^maxlevel)
```

I agree and have cut this down to the number of nodes with children.

Finally, if you put in a delay at 115 then you can see how the tree is built up. – **J. Diamond, Cheadle, Cheshire.**

□ We asked if anyone could make Trevor's program more efficient and this is one of many replies.

```
10 MODE 2:maxlevel=6:treewi
dth=-100:length=-100:factor
=2
20 DIM right(2^(maxlevel-1)
):treeend=2^(maxlevel-1)-1
30 ORIGIN 320,400
40 node=1
50 GOSUB 110
100 END
110 REM*recursive subroutin
e*
120 IF node>treeend THEN 24
0
130 REM * left branch *
140 DRAWR treewidth,length
150 node = node*2
160 treewidth = treewidth\f
actor:length=length\factor
170 GOSUB 110
180 REM right branch
190 DRAWR -treewidth,length
:right(node)=1
200 node=node*2+1
210 treewidth = treewidth\f
actor:length=length\factor
220 GOSUB 110
230 REM reset variables & f
all out
240 node=node\2:treewidth=t
reewidth*factor:length=leng
th*factor
250 IF right(node)>0 THEN M
OVER treewidth,-length:RETU
RN
260 MOVER -treewidth,-lengt
h
270 RETURN
```

In the dumps

ROLAND Waddilove's Screen Dump in your March issue does not appear to work correctly when using the ORIGIN command.

The offset as set by ORIGIN does not seem to have any effect on the screen dump. I would guess that this is because the machine code does not take into account offsets defined by ORIGIN.

In order to make the screen dump work correctly with an offset, the offset must be done numerically in the plotting rather than by using the ORIGIN command.

For instance:

```
10 ORIGIN 320,200
20 PLOT x,y
```

should be changed to:

```
10 offsetx=320:offsety=200
20 PLOT x+offsetx,y+offsety
```

to yield the same result on screen but allow the screen dump to work correctly.

Entering:

```
ORIGIN 0,0
```

as a direct command or placing it at the start of a program should ensure that the origin is set to its default position.

I have also discovered that the "Dump ANYTHING off the screen" statement at the head of the article is not entirely true, because when the screen dump program is loaded and run it will not allow the definition of new characters via the SYMBOL AFTER and SYMBOL commands.

I assume that this is because the machine code resides in the same area in memory as the Symbol data.

If this is so, then would it be possible to publish the program again but with the machine code routine residing elsewhere in memory (possibly avoiding the DDI-1 interface as well) for those people who do not have the neccesary software or knowledge to relocate it themselves? – **J.W. Constable, Acton, London.**

□ It's not the machine code that's at fault, it is a consequence of the way SYMBOL AFTER works. Save the machine code as a binary file, then after the SYMBOL AFTER command, move HIMEM down and load the machine code.

Enhanced ROM

I FEEL the time has come that my pen should be pulled from its scabbard. Not necessarily to draw blood, but rather to prick a certain body to awareness.

I am one of the growing number of people who took a leap of faith, rather than quantum, and bought a CPC464. Thankfully, up to now I have not regretted this decision.

However, with the dawn of the 664 and its enhanced Basic ROM, I was very surprised to read that Amstrad are not to offer this ROM as an upgrade path for existing 464 owners.

I find the thinking behind this decision would suggest that Amstrad is ignoring the needs of the very people that helped make the 464 the success that it is today.

Even Sinclair has the decency to offer upgrades where possible.

Why should it be so much more difficult for Amstrad to render the same service – at least if only to recognise and reward the loyalty of its patrons?

It is a shame that without the new ROM the 464 already has a smell of obsolescence about it.

I had hoped for something better from Amstrad. They started so well . . . – **Malcolm W. Taylor-Hay, Colchester, Essex.**

MICROWARE DISCOUNTS (Cleckheaton)

Adventure Quest	7.99	Ghostbusters	9.75	Return to Eden	8.20
Airtraffic Control	6.75	Ghouls	5.75	Roland Ahoy	7.75
American Football	8.20	Grnd Prix Driver	6.95	Roland Goes Digging	7.75
Bridge Player	9.95	Harrier Attack	7.50	Roland on the Run	7.75
Centre Court	7.75	Heroes of Khan	5.95	Software Star	6.75
Chess (Amsoft)	8.95	Hunchback	7.75	Sorcery	7.75
Console Adventure	7.95	Hunter Killer	6.75	Spanerman	6.25
Combat Lynx	7.75	Jet Set Willy	7.45	Splat	8.95
Cubit	7.50	Killer Gorilla/Gauntlet	8.75	Steve Davis Snooker	6.25
Dungeon Adventure	7.99	Magic Sword	8.95	Tank Busters	6.75
Eletro Teddy	7.75	Manic Miner	7.25	Technician Ted	6.75
Emerald Isle	6.75	Millionaire	5.75	The Hobbitt	12.75
Erbert	5.20	Mini Office	5.95	The Quill	14.99
Eric the Viking	8.50	Osprey	9.95	Tripods	11.25
Football Manager	6.75	Pjamarama	7.75	World Cup Football	6.75
Forest at Worlds End	5.95	Quackerjack	7.75		

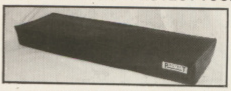
Low Cost – Fast Delivery – Original Software

Cheque/POs with order to:

MICROWARE DISCOUNTS (Cleckheaton)
P.O. Box 22 Cleckheaton BD19 4UB. West Yorks.

IMPORTANT

PROTEKT YOUR AMSTRAD – PROTECT YOUR POCKET



Dust/Spillage covers, tailored in luxury soft fabric backed vinyl. All seams stitched in strong cotton. Offered in a choice of three colours: BLACK, NAVY, CREAM.

Special price of only **£4.75** inc. P&P (UK only)

Please send Cheques/POs made payable to:
PROTEKT
91 Longue Drive, Calverton, Nottingham NE14 6QE.
Allow 28 days for delivery.
"PROTEKT" FOR DUST COVERS THAT WILL ATTRACTIVELY PROTECT YOUR AMSTRAD.

AN ADVANTAGE AT LAST!

JOIN THE INDEPENDENT USER GROUP FOR THE AMSTRAD CPC464

For the same price as one games cassette, you get:
- 12 monthly newsletters with ideas, reviews, letters and more.
- Access to a database of information, programs and advice.
- Opportunities to communicate with other computer users.
- Offers of reasonably priced software and accessories.

Send £8.95 to:
ADVANTAGE, 33 Malyns Close,
Chinnor, Oxfordshire
or send large stamped addressed envelope for more information

DISCOUNT SOFTWARE

GAMES		BUSINESS & TUTORIAL	
Ghostbusters	9.25	Guide To Basic 1	15.95
Daley T's Decathlon	7.25	Guide To Basic 2	15.95
Sorcery	7.50	Easy Amsword	7.95
Steve Davis Snooker	5.95	Amsword (adv)	15.95
Football Manager	6.25	Concise Basic Spec	14.95
Jet Set Willy	6.75	Concise Firmware Spec	15.95
Fighter Pilot	7.25	Home Budget	15.95
Centre Court	6.95	Devpac ass/diss	19.95
Master Chess	6.95	Masterfile (cass)	19.95
American Football	7.75	Screen Designer	9.25
Flight Path 737	5.75	Pitman Typing Tutor	7.95
Jewels Of Babylon	4.65	Kuma Database	12.25
Heroes Of Karn	4.65	Kuma Easy VAT	29.95
Forest At Worlds End	4.65	Tasword 464	15.95
Air Traffic Control	6.25	Tasprint	7.75
The Hobbit	11.25	Mini Office	4.95

ALL PRICES INCLUDE POSTAGE & VAT
All original titles. Write for FREE price list. Please send Cheques/POs to:
M.J.C. SUPPLIES,
'Scoja', London Road, Hitchin, Herts SG4 9EN.

BUSINESS SOFTWARE
for the
AMSTRAD CPC464

SIMPLE ACCOUNTS **£29.95 incl VAT**

Easy to use, comprehensive book-keeping system complete with step by step instruction manual.
- Full Analysis of Income & Expenditure.
- Powerful Entry Search Routines.
- Monthly and Year to Date Reports.
- Quarterly VAT reports.

(All VAT routines are optional and may be omitted by non-VAT Traders)

MANAGEMENT ANALYSIS **£19.95 incl VAT**

Extracts Management Information From Your Accounts
- Superb Graphic Representation.
- Pie Charts Analyse Sales and Purchases.
- Bar Charts Compare your Monthly Figures
- PLUS Detailed Statistical Reports.
- Compatible With SIMPLE ACCOUNTS.

Specification Manual and Sample Output Available on Request.

CORNIX SOFTWARE LTD.
16 Kneesworth Street, Royston, Herts SG8 5AA.
Tel: Royston (0763) 46065

Specialists in Business & Financial Software

For advance notice of future programs
Make sure you are on our mailing list

EASIDATA Fast machine code database. Up to 32 fields per record. Search works on any combination of field and can include wild cards. Complete with report generator program and full sort facilities. ***Disc £29.99.***

DEAN DATABASE/ MAILING LIST Easy to use and flexible electronic card index. Up to 50 items per record. Fast sort and search facilities. Simple label printing with multiple copy facility. ***Cassette £12.99. Disc £19.99.***

THE "MASTERCLASS" MUSIC COMPOSER Superb fun for all the family. Suitable for the beginner or experienced musician. Clear graphics display the music as it's composed. Manuscript prints for Epson RX80 or Amstrad DMP1 printers. Envelope design utility. ***Cassette £12.99. Disc £19.99.***

EASIVAT Full VAT book-keeping system. Send S.A.E. for full details.

All prices include VAT
postage and manual.

DEAN SOFTWARE LIMITED
The Mews, Mitcheldean, Glos. GL17 0HH.

Another great reader offer!

This is the book that shows you how easy it is to create adventures on your Amstrad! Just type in the program, then follow the simple instructions to create as many exciting, hard-to-solve adventures as you like.

All you have to do is enter the data of your choice and the Pentacle program does ALL the hard work, building up an adventure you can save – and even sell!

As a bonus there's also a full set of data showing Pentacle in use with an adventure set in bygone days – Castle of Doom. Then a change of pace is provided with a full graphical adventure, The Search for Speck.

Now adventuring will never be the same again!

FREE!

The book that will really set you adventuring with your Amstrad will be sent to every subscription ordered on the form on the right.

Worth £2.99!

GIVE YOUR FINGERS A REST!

JANUARY: ***Smiley:*** *Guide Smiley round the labyrinth avoiding Grumpies.* ***Code:*** *Be a mastermind.* ***Binary:*** *Baffled by binary bits? Let our utility help you out.* ***Dancer:*** *Simple but fun. This is a lovely mover.* ***Trapper:*** *Pen the monsters of the maze.* ***Scroller:*** *A slick sideways scrolling routine.* ***Letter Litter:*** *Learn the keyboard.*

FEBRUARY: ***Digger:*** *Search for crystals and avoid the aliens.* ***Simon:*** *Our version of this classic game.* ***Kingdom of Craal:*** *A devious fantasy.* ***Text Editor:*** *A powerful word processor.* ***Reaction Timer:*** *Test out your reactions on this simple little program.* ***Origins:*** *Changing the graphics origin produces dramatic effects.*

MARCH: ***Starfleet:*** *Can you qualify as a starfleet pilot?* ***Swamp:*** *Join the battle against giant frogs.* ***Parachute:*** *Parachute to earth in this educational program.* ***Dump:*** *Screen dump utility for Modes 0 and 1.* ***Hexer:*** *A hexadecimal loader.* ***PLUS . . . 3D Four in a Row.***

APRIL: ***Mad Adder:*** *Appease our aggravated adder.* ***Egg Blitz:*** *Eggs, boy scouts and a stunt pilot make up the fun.* ***Simple Sprites:*** *Create sprite-like effects.* ***Caesar Cipher:*** *Crack the code.* ***Pilot:*** *An easy to learn CAL language.* ***Spelling:*** *Learn to spell with this educational game.* ***Aleatoire:*** *The odd ways mariners make decisions.* ***Palindrome Tester:*** *Slick string handling techniques.*

MAY: ***Tronn Cycles:*** *Enter the Maze and pit your wits in this two player arcade classic.* ***Castle of Fear:*** *Fantasy is the order of the day*

Computing with the AMSTRAD

MAIL ORDER OFFERS

ORDER FORM

All prices include postage, packing and VAT and are valid to July 31

Please enter your requirements by ticking boxes

		£	p

Annual subscription

UK £12	6001 ☐
Eire £13 (IR £16)	6002 ☐
Overseas (Surface) £20	6003 ☐
Overseas (Airmail) £40	6004 ☐

Please send me my free Amstrad Adventure Book 6052 ☐

Commence with _____ issue TOTAL

Tapes £3.75 (UK & Overseas)

January (Smiley)	6020 ☐
February (Digger)	6021 ☐
March (Starfleet)	6022 ☐
April (Mad Adder)	6023 ☐
May (Tronn Cycles)	6024 ☐
June (Da Bells)	6025 ☐
July (Missile Command)	6026 ☐

TOTAL

3" Discs £6.75 (UK & Overseas)

January (Smiley)	6060 ☐
February (Digger)	6061 ☐
March (Starfleet)	6062 ☐
April (Mad Adder)	6063 ☐
May (Tronn Cycles)	6064 ☐
June (Da Bells)	6065 ☐
July (Missile Command)	6066 ☐

TOTAL

Back issues £1.25 UK, £1.50 Overseas (Surface)

January	6006 ☐
February	6007 ☐
March	6008 ☐
April	6009 ☐
May	6010 ☐
June	6011 ☐

TOTAL

Mini Office

Tape £5.95	6035 ☐
3" Disc £9.95	6036 ☐

TOTAL

Modems

Modem 2000 package £89.95	6053 ☐
Telemod 3 package £164.90	6054 ☐

TOTAL

Cassette tape and disc annual subscription

Tape £40 (UK & Overseas)	6005 ☐
3" Disc £70 (UK & Overseas)	6059 ☐

Commence with _____ (state month)

TOTAL

>>>

TOTAL

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 keyboard 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

Binder £3.95 UK, £5.00 Overseas 6038 ☐

TOTAL

Dust Cover £3.95 (UK & Overseas) 6037 ☐

TOTAL

dk'tronics speech synthesiser £34.95 (UK) 6040 ☐

TOTAL

Amstrad Pentacle Adventure Creator £2.99 6052 ☐

TOTAL

Red Arrows

Tape £8.95	6050 ☐
Disc £12.95	6051 ☐

TOTAL

Classic Games on the Amstrad Vol. 1

Tape £5.95	6048 ☐
Disc £9.95	6049 ☐

TOTAL

Overseas postage: add 50p per item unless otherwise indicated.

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:
061-480 0171
(24 hours)
Don't forget to quote your credit card number and full address

All the program listings from each month's issue are available for only ***£3.75*** *on cassette and* ***£6.75*** *on disc.*

in this baffling concoction. ***Mouse:*** *Can you trap the furry fellow in this simple but addictive game?* ***Cedric:*** *Fun with shape recognition in this novel interpretation of pelmanism.* ***Character Maker:*** *Define your own characters with our superb utility.* ***RSX:*** *New commands for your micro.* ***Cards:*** *Baffle your friends with an apparently simple computer card trick.* ***Trees:*** *A clever routine to create 'trees'.*

JUNE: ***Easydraw:*** *Hours of fun and entertainment. This versatile graphics utility allows you to create and manipulate your own pictures, saving them to tape or disc.* ***Da Bells:*** *Hilarious arcade action for the whole family with the obligatory Hunchback on the ramparts. Can you rescue Esmerelda?* ***AMmon:*** *A superb machine code monitor to complement our Z80 series.* ***Menu:*** *Create colourful menus with this intelligent disc utility.* ***Nim:*** *Baffle your friends with Alleatoire's simple routine to play the famous mathematical game.* ***Quads:*** *A clever routine to display expanding quadrilaterals.*

JULY: ***Missile Command:*** *Take command of your laser base and save earth from the invading hordes.* ***Jumbo's Wordgame:*** *Have hours of family fun with our Amstrad version of Odd One Out.* ***Biorhythms:*** *Find out if you're going to have "one of those days" with this superb utility.* ***Raw:*** *Add power to your programming with this superb machine code assembler.* ***Compacter:*** *Reduce the memory used while saving a screen with this handy utility. Load it back in half the time.*

ADVERTISERS INDEX

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 theAmstrad 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

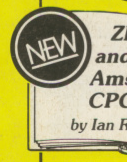
North Sea Bullion | Galaxia | Fruity Frank | Star Avenger | Shadow of the Bear

Home Budget | Database | Zen Assembler | Artwork | Bridge

BOOKS

The Amstrad CPC 464 Explored
by John Braga

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.



ZEN and the Amstrad CPC 464
by Ian R. Sinclair

This book covers the CPC 464 itself, a detailed guide to Zen, an introduction to the Z-80, ROM, and RAM, subroutines, screen and keyboard, cassette I/O and how to plan a program. The text is littered with illustrations, diagrams and helpful program examples.

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC 464 Micro-computer.

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 ☐

CWA7

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.