A Database Publication

Computing with the AMSTRAD

No. 11
November 1985
£1

The independent magazine for Amstrad computer users

Spelling's fun with ALPHABET TRAIN

Plus complete listings for TWO fast-action arcade games

Expert advice on sound, graphics and the Z80—and all the latest software

ESP

ELECTRIC STUDIO PEN

THE PEN THAT LIKES TO SAY YES

FEATURES/ FUNCTIONS	ESP	dk'tronics	A.N.Other
SINGLE COMPLETE ON SCREEN MENU	YES	NO	
DRAG SCREEN OBJECTS	YES	NO	
FLIP SCREEN OBJECTS	YES	NO	
CURSOR REMOVAL	YES	NO	
ELASTIC BOXES	YES	YES	
ELASTIC LINE	YES	YES	
ELASTIC TRIANGLE	YES	NO	
ELASTIC ELLIPSE	YES	NO	
ELASTIC DIAMOND	YES	NO	
ELASTIC POLYGON	YES	NO	
ELASTIC HEXAGON	YES	NO	
ELASTIC OCTAGON	YES	NO	
ELASTIC CUBE	YES	NO	
ELASTIC PYRAMID	YES	NO	
CIRCLES	YES	YES	
SOLID CIRCLES	YES	NO	
SOLID BOXES	YES	NO	
SOLID ELLIPSES	YES	NO	
WEDGES	YES	NO	
BEZIER SIMULATIONS	YES	NO	
ZOOM EDIT	YES	YES	
REVERSE/MIRROR IMAGES	YES	NO	
REFERENCE BACKGROUND	YES	NO	
GRID BACKGROUND	YES	NO	
X,Y DISPLAY OPTION	YES	NO	
PAINT FILL	YES	YES	
COLOUR WASHING	YES	NO	
RESIDENT SCREEN DUMP	YES	NO	
3D EDGE PLOTTING	YES	NO	
TEXT	YES	YES	
9 BRUSH SIZES	YES	NO	
18 SPRAY NOZZLES	YES	NO	
4 BASIC TEXTURES	YES	NO	
TEXTURE VARIATION	YES	NO	
XOR TEXTURE SHADING	YES	NO	
RESIDENT SYMBOL/SHAPE FILE	YES	NO	
RESIDENT FLECK PRINT FILE	YES	NO	
26 PAPER COLOURS	YES	NO	
15 COLOUR/TONE PALETTE	YES	NO	
POINT SETTING	YES	YES	
FIXED POINT RAYS	YES	NO	
MIRROR DRAWING	YES	NO	
HOME FUNCTION	YES	NO	
KEY CONTROL NUDGE	YES	YES	
JOYSTICK NUDGE	YES	NO	
AVAILABLE FOR 464	YES	YES	
AVAILABLE FOR 664	YES	?	
MODE 1 & 2 AVAILABLE	YES	?	

Please compare any other pen package currently available

DUE TO LACK OF SPACE, WE ARE NOT ABLE TO LIST THE OTHER 20+ FUNCTIONS OUR PEN IS CAPABLE OF.

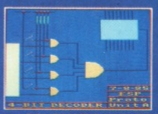
WHILST OUR COMPETITORS MOSTLY SAY NO!

JUST SOME OF THE THINGS YOU CAN DO WITH THIS COMPLETE LIGHT PEN PACKAGE

USER DEFINED CHARACTERS

FREEHAND DRAWING WITH THE GRAPHICS PROGRAM

GAME SCREEN DESIGN WITH COLOUR EDITING FACILITY



SCHEMATIC AND GEOMETRIC DESIGN DRAWINGS

AVAILABLE FOR: CPC464 ON TAPE **£19.95**

CPC464 ON DISC (incl. Interface) **£26.95**

CPC664 ON DISC **£26.95**

Also available: High Res Graphics Program for more serious applications (includes both Mode 1 and Mode 2)

TAPE **£9.95** OR DISC **£14.95**

AVAILABLE FROM ANY GOOD COMPUTER STORE

If you have any difficulty obtaining our products, please send cheque/P.O. to:

THE ELECTRIC STUDIO

P.O. BOX 96, LUTON LU3 2JP Tel: (0582) 595222

AMSTRAD CPC 464 £7.95

Other Amstrad titles include:–

DARK STAR • TANK BUSTERS

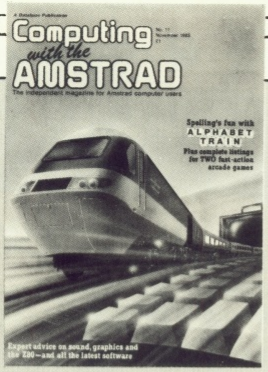
Selected Design Design titles are available from:–

Boots WOOLWORTH John Menzies
COMET WHSMITH

and all leading software stockists, or direct from:–

Design Design

Design Design Software,
125 Smedley Road, Cheetham Hill, Manchester M8 7RS
Trade enquiries:– 061 205 6603
Price includes p & p



Vol. 1 No. 11 November 1985

Managing Editor: **Derek Meakin**
Features Editor: **Mike Bibby**
The A Team: **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
Telecom Gold Mailbox: 72:MAG001
Telex: 256871 MONREF G
Quoting Ref. MAG001
Prestel Mailbox: 614568383

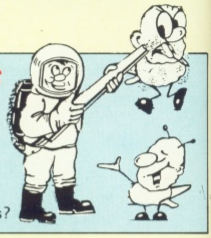
Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

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

ABC Member of Audit Bureau of Circulations

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

12 FRUTTIES
The fearsome Frutties are on the rampage in the factory. Can you put a stop to their evil onslaught before they gobble up all the workers?



17 MACHINE CODE
We take a detailed look at Relative Branching as our Z80 series for the absolute beginner delves further into representing numbers in machine code.

21 BEGINNERS
All you wanted to know about the Amstrad's READ and DATA commands and were afraid to ask. You'll find they're much easier to handle than you think and extremely efficient.

25 MAXAM
If you're keeping up with our machine code series you'll be looking for an assembler. Arnor's Maxam could be what you need – and it contains a monitor and text editor as well.

29 HARDWARE
Micro Power's Superpower ROM card and their Disc Power ROM could be the forerunners of a flood of add-ons for the Amstrad. They could also be setting a standard that will be difficult to beat.

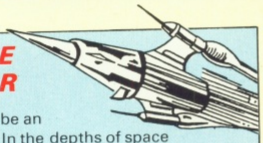
32 ADVENTURES
Gandalf was missing last month – he got stuck in a maze of twisty little passages and has only just appeared. Still he's all the better for it and ready to carry on with his expedition into Adventureland.

Contents

37 SOFTWARE SURVEY
Our team of frank and thorough reviewers look at some of the latest releases for the Amstrad.

41 READY REFERENCE
Have the Amstrad's CP/M elements at your fingertips with the eighth of our quick reference charts.

44 GRAPHICS
We bring this fascinating series to a close with some superb routines to demonstrate the Amstrad's EOR function and its use in screen animation.

52 SPACE TRUCKER

Have you the guts to be an intergalactic Yorkie? In the depths of space there's more to delivering freight than meets the eye.

58 MACHINE CODE GAMES
Keep a watch on those wandering sprites with this superb collision detection routine.

61 HOW BASIC WORKS
In the first of a two part series we take a close look at the inner workings of Amstrad Basic.

63 TRACE
De-bugging your programs will be so much easier with this easy to use alternative to the TRON command.

68 ALPHABET TRAIN
Learning spelling and shape recognition are child's play with the help of this superb educational game.

70 SOUND
We bring down the curtain on this series with a look at the SQ command which monitors the channels.

76 ALEATOIRE

Do you think most card games are just a matter of bluff? If so, you may just change your mind after playing against our puzzles expert's latest conumdrum.

82 AL'S BEAT
The enemy down below has worn out Al we fear. He's going to have a little rest, and start again next year.

87 DISASSEMBLER
To complement his assembler in the July issue, Roland's written this simple but effective disassembler to add to your machine programmer's toolkit.

91 BUSINESS
Continuing our regular series for the businessman, Jo Stork takes an in depth look at Entrepreneur, the complete business start up kit from Triptych.

93 ANALYSIS
Saving files to cassette or disc and reading them back again is simple if you follow the guidelines in Trevor Roberts' latest programming exercise.

94 SIDEWRITER
Add that extra dimension to your text screens with this clever utility to turn characters on their side.

97 POSTBAG
The part of the magazine you write yourselves. Just a small selection from the many interesting and informative letters you've been sending us.

100 ORDER FORM
Take out a subscription, order a back issue, cassette tape, disc, dust cover or binder – and you can do it all on one simple form.

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.

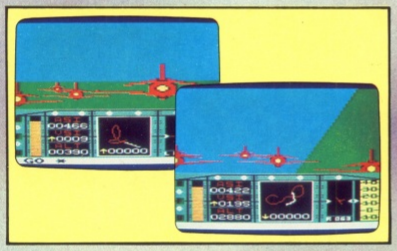
Every ounce of power contained in the micro, and its enhanced sound and graphics capabilities, is used to give the utmost realism to re-creating the most spectacular aeronautical displays ever seen in the skies of Britain.

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!



FREE RED ARROW CONTEST

Everyone buying the program can enter an exciting competition. Winners will be given a VIP visit to the Red Arrows base at RAF Scrampton, the wartime home of the Dambusters, including two nights' accommodation at a luxury hotel. While at Scampton you will be invited to sit at the controls of a Hawk.

ORDER FORM

Amstrad 464, 664, 6128

Tape	£8.95	*6050* ☐
3" Disc	£12.95	*6051* ☐

I wish to pay by:

☐ Access/Mastercard/Eurocard No.

☐ Barclaycard/Visa No.

☐ Cheque/PO made payable to Database Publications Ltd.

Expiry date: /

Name _____

Address _____

_____

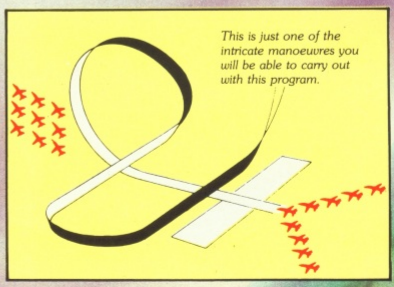
Signed _____

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 0171 *Don't forget to quote your credit card number and full address.*

CA11



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
"If you want to have some fun and say that you were flying in formation with the 'Big Nine' then this is the program for you."
– Amstrad User

NOW. TWO VORTEX GIANTS TURN ON AMSTRAD.

Vortex SOFTWARE

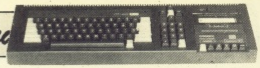
- PLEASE SEND ME -

☐ T.L.L.	AMSTRAD CPC 464	£7.95
☐ ANDROID TWO	AMSTRAD CPC 464	£7.95
☐ ANDROID ONE	AMSTRAD CPC 464	£7.95
TOTAL VALUE		£

NAME _____ BLOCK LETTERS PLEASE

ADDRESS _____

ENCLOSE P/O CHEQUE PAYABLE TO VORTEX SOFTWARE (SALES) LTD.
SEND TO VORTEX SOFTWARE (SALES) LIMITED, VORTEX HOUSE,
24 KANSAS AVENUE, OFF SOUTH LANGWORTHY ROAD, SALFORD, M5 2GL.



Tandy joins in

HIGH Street giant Tandy has jumped aboard the Amstrad bandwagon with the decision to market the 464 and 6128 models through their outlets.

A range of popular software for the machines is also to be introduced in time for the Christmas demand.

Commenting on the decision to market computer brand names other than Tandy, Tandy's computer general manager, Ted Russell, said: "This new initiative reinforces the seriousness of our High Street presence in both business and home computing markets".

BESA PICKS AMSTRAD

THE Amstrad success story has led the newly formed British Educational Software Association to make the machines its primary target for expansion.

"It is a logical step for us to embrace Amstrad computers as soon as possible, particularly now they are starting to sell in quantity to schools and colleges", BESA spokesman Craig Thatcher told *Computing with the Amstrad.*

CPC6128 leading in Christmas computer stakes

AMSTRAD has emerged as the hot favourite to win the Christmas computer stakes thanks to the latest addition to its stable, the CPC6128.

It is now confidently predicted that the company will overtake traditional front runners Sinclair and Commodore in terms of cash sales during the festive season.

The forecast comes from John Rowland, merchandise controller of High Street retail giant W.H. Smith's computer division.

"Although Amstrad is only likely to come third in units sold, it will beat the big two when it comes to revenue", he told *Computing with the Amstrad.*

"That result will owe much to the CPC6128, for which there has been an excellent initial response since the launch".

It seems that the machine is attracting a different consumer age group than normal in the W.H. Smith in-store computer departments.

"The people who are buying them are slightly older than usual", observed John Rowland. "Customers for the CPC 6128 tend to fall into the 18 to 35 age group, as compared to being predominantly in the 12 to 17-year-old category for other machines.

Applications

"And they are also much more interested in applications software, not just games. This in itself would suggest that sales will not be entirely seasonal".

Meanwhile over at Dixons, group financial director Egon von Greayez was similarly enthusiastic about Amstrad's future.

"The company will be very successful", he insisted. "It understands the marketplace and can be flexible, which can only be to the good".

Whereas Dixons agree with W.H. Smith in that the CPC6128 will sell very significant numbers, von Greayez sees the new PCW8256 as having the most impact on the market in the long term.

"This is going to be a major performer all year round", he said, "and we are going to make sure of that".

MOUSE FOR CPC RANGE

ONE of the most successful peripherals in the history of home computing – the AMX Mouse – has been launched for the Amstrad range of micros.

Shortlisted for the prestigious Peripheral of the Year award and with unit sales of its original BBC Micro version topping 10,000 in the first nine months following its launch, the opto mechanical device from Advanced Memory Systems has been the subject of much critical acclaim.

The new version offers Amstrad users an entirely new approach to computing that makes the keyboard seem almost old-fashioned.

The package includes AMX Art, a computer-aided drawing program utilising on-screen windows, icons, pull-down menus and pointers for producing professional standard drawings or mere doodles that can be saved and printed.

Pattern Designer enables the user to create an unlimited number of designs that can be stored and used to fill in the work produced with AMX Art.

Users can create a Mouse environment in their own programs with AMX Control which extends the Basic software commands. Icon Designer is a program for creating and storing icons for the user's own programs.

"We could have converted the mouse for any micro", says Nick Pearson of AMS. "But we judged the Amstrad machines as being the ones to set the pace not only up to Christmas but next year as well".

Top selling Amstrad programs

Compiled every month for *Computing with the Amstrad* by Terry Blood Distribution Ltd.

1	***Way of the Exploding First***	*Melbourne House*
2	***Starion***	*Melbourne House*
3	***Alien 8***	*Ultimate*
4	***Rocky Horror Show***	*C.R.L.*
5	***Dundarach***	*Gargoyle Games*
6	***Arnhem***	*C.C.S.*
7	***Beach Head***	*U.S. Gold*
8	***Knight Lore***	*Ultimate*
9	***Decathlon***	*Activision*
10	***Confuzion***	*Incentive*

MAJOR ADVERTISING CAMPAIGN WILL BOOST SALES

AMSTRAD is spending £5 million on its first major TV and press advertising campaign since it entered the home computer market last year.

Sixty five per cent of the money will go on TV commercials, the rest on coverage in the dailies, quality Sundays and specialist press.

Three 40 second commercials which started in October feature the CPC464 and the CPC6128.

The first commercial was made six months ago by Delaney Fletcher Delaney, the advertising agency which handles the account. Featuring the CPC464 and emphasising the total package concept of the product, it has only now been aired to coincide with pre-Christmas demand.

Promote

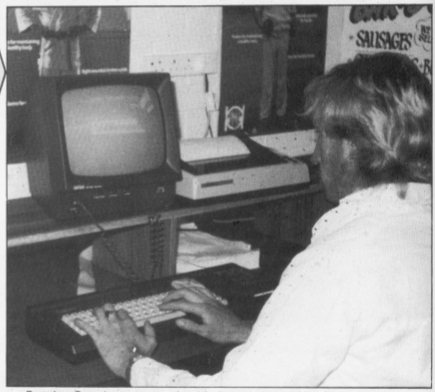
An office situation is used in the CPC6128 advertisement to foster the catch phrase: "Who says business and pleasure can't mix?" It is designed to promote the machine to the home and business micro user.

Greg Delaney, joint MD of the agency, describes the third commercial as a "sports pastiche". It depicts a boy commentating on a number of sports and aims to emphasise the fun and fantasy aspect of Amstrad machines to the youth market.

Amstrad marketing director Malcolm Miller claims that the three advertisements, scheduled to be screened until December, "will humanise the products and highlight their benefits".

Of the money being spent on press advertising, Miller says that £500,000 alone will be used to promote the PCW8256, with TV advertising possibly to follow. Press coverage will reflect the TV advertising.

Specialised packs are selling well



Butcher Gary Lynes operating his Amstrad based business system

THE new CPC6128 could soon be a permanent fixture at your local butcher's shop.

Hundreds of Britain's 17,000 butchers have expressed an interest in a vertical software package specially designed for them by MASS Business Systems and promoted through their national magazine, Meat Trades Journal.

The system, which consists of a CPC6128, a micro printer and software, has already been installed in several shops.

The low cost of the Amstrad hardware has helped to quarter the price of the system compared to similar vertical packages.

The £899 cost includes delivery, tuition and backup services, says MASS director Mike Lake.

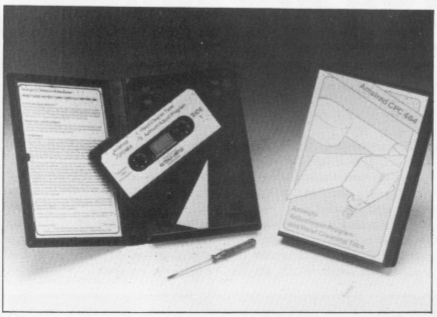
"Backup is very simple, with immediate exchange. Because of the low cost hardware it is quite practical to have a few complete systems on standby. Any problems and we give the customer a new one", said Lake.

"This is one of several systems sold by us using this package approach and the Amstrad computer fits in very well indeed.

"Packages are also available on other computers such as the Apricot, but the Amstrad is our favourite.

"We currently have available, or are working on, vertical systems for firms involved in vehicle recovery, exhaust fitting, and taxi operations as well as public houses, petrol stations and newsagents.

"We're waiting with bated breath for someone to bring out a hard disc to provide what we consider to be the final link in ultra low price business systems".

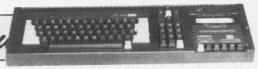


Kit keeps tape deck in tune

A COMBINED cassette cleaner and azimuth alignment tape for the CPC464 has been produced by Kiltdale.

Priced at £4.99, the product completely eliminates the problems of recording head misalignment, says the manufacturer.

The package includes a combined head cleaner and azimuth tape, a screwdriver, indicator arrow and full instructions.



Mr MEN FOR THE AMSTRAD

THE early-learning program, First Steps with the Mr Men, is now available for the Amstrad from Mirrorsoft.

It features Mr Greedy, Mr Silly and Mr Forgetful in four games for the under-fives. They are not only fun to play but also help the young develop skills, claims the company.

Obstacles have to be manoeuvered round in Mr Greedy's Ice Cream Hunt, colours and shapes matched in Mr Silly's Hat Game and hidden pairs found and matched in Mr Forgetful's Wardrobe and Letter games.

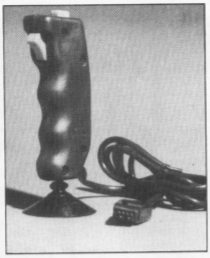
Priced at £8.95, the pack comes with a 12 page illustrated storybook cum instruction manual and a colour-coded key guide.

Mini Wordstar

A LOW price version of WordStar has been released by MicroPro for the Amstrad.

Called Pocket WordStar because of its £149 price tag it is being aimed at the cost-conscious businessman. The original Wordstar costs £295.

The company says the new package offers all the features of WordStar and MailMerge.



SPORTY STICK

A BASELESS joystick created for single hand use is now available for the Amstrad.

Merseyside company Lightwave Leisure signed an exclusive deal with West German manufacturer John Hall to import the device, known as "The Stick".

It has no console or trigger, consisting simply of a shaft with non-slip grip containing small mercury switches which operate when tilted.

There are two buttons, for the thumb and index finger.

Lightwave says the Stick allows for brief, gentle movements of the wrist to operate controls directly and precisely, making it ideal for sports and simulation games.

It costs £12.99 and carries a one year limited warranty.

Dr Who's trying to beat pirates

THE about-to-be released Dr Who and the Mines of Terror game for the Amstrad will contain a number of new anti-piracy devices.

Producers Micropower say the devices have been incorporated to prevent illegal copiers stealing a march on its dealers. Naturally the company is keeping very quiet as to what precisely they are.

To ensure that no one outside the company gets a full version of the game before its launch even reviewers are only being sent a threequarter length copy.

The company describes Dr Who as an adventure game depicted in arcade graphics. True to the television series, it has a large number of problems to solve and according to the company is five times as complex as its last release, Castle Quest. It costs £14.95.

Full screen editor for disc models

NOW available for the Amstrad disc computers are HiSoft's full screen program editor, ED80, and range of programming tools.

The screen program editor runs under CP/M and its features include Wordstar compatible editing commands, help screens, copy, delete, restore, cut and paste, tab handling and side scrolling.

For the CPC464, 664, 6128 and PCW8256, it is able to list disc directories and erase files from within ED80. Price: £29.95.

The new programming tools are for the PCW8256 and CPC6128 only and include Pascal80, Devpac80 and The Knife.

The first is a virtually standard implementation of Pascal and comes with the ED80 and introductory Pascal programs including an intelligent disassembler.

Devpac80, a Z80 assembler development package, consists of a fast macro assembler with conditional assembly, a full range of arithmetic operators, front panel debugger and disassembler, ED80, memory search and copy.

The Knife is a disc track and sector editor that allows the user to recover erased files, cut and paste and search for patterns.

Both Pascal80 and Devpac80 cost £39.95, The Knife costs £12.95.

Software firm helps the starving

THE plight of the starving in Ethiopia seems to have been forgotten by the computer industry, says David Cruse, spokesman for Ixion Software, in a letter to *Computing with the Amstrad*.

"We are a little surprised that, what with the massive success of such projects as Band-Aid and Live-Aid, the computer industry appears to have done little or nothing to help", he writes.

Ixion had decided to make a first step towards a united fund-raising effort from the industry. It is to donate 50 per cent of profits from its complete range to the Ethiopian Relief Fund.

This will include the company's newly-released game The Avaunting and the more familiar Community, The Cliche and Bridge to the East.

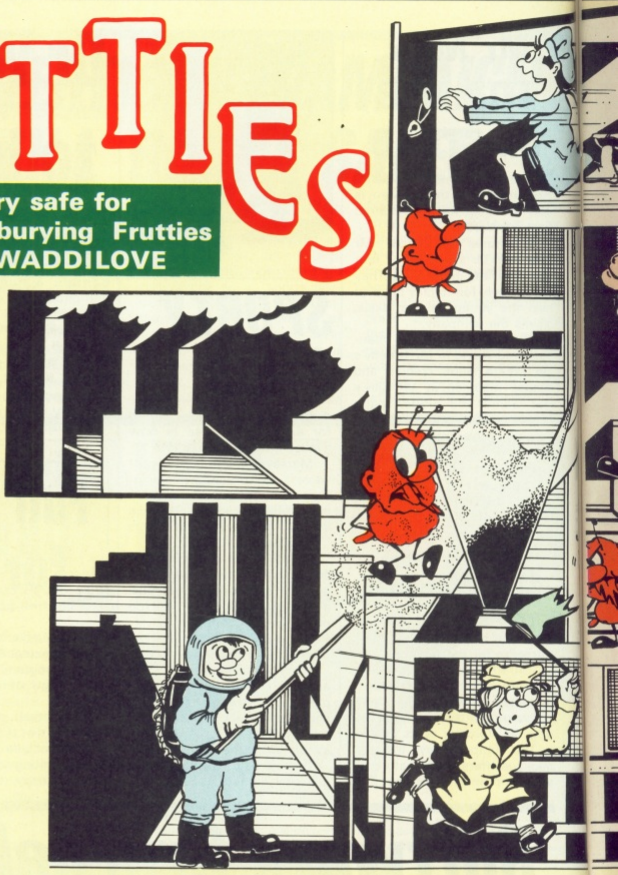
If sales go well Ixion will increase its percentage to the fund to up to 75 per cent.

"Hopefully the coverage of our actions will influence other companies to adopt similar policies", said Cruse.

FRUTTIES

Make the factory safe for humans – by burying Frutties

with ROLAND WADDILOVE



FRUTTIES is a fast, colourful, exciting, ladders and levels arcade game. The action takes place in a factory which has been overrun with Frutty monsters. These fearsome creatures have gobbled up all the machinery and are now after the workers.

All the doors and windows in the rooms have been sealed to prevent them from escaping. Your task is to enter each room, wearing breathing equipment, and destroy all the Frutties.

The Frutties naturally enough, aren't very happy at the prospect of being bumped off one by one. So watch out, they'll try and make a meal of you if they can.

The only way of disposing of a Frutty is to dig a deep hole in its path and lure it across. Be careful though, if you fall into one of the holes it's curtains for you as well!

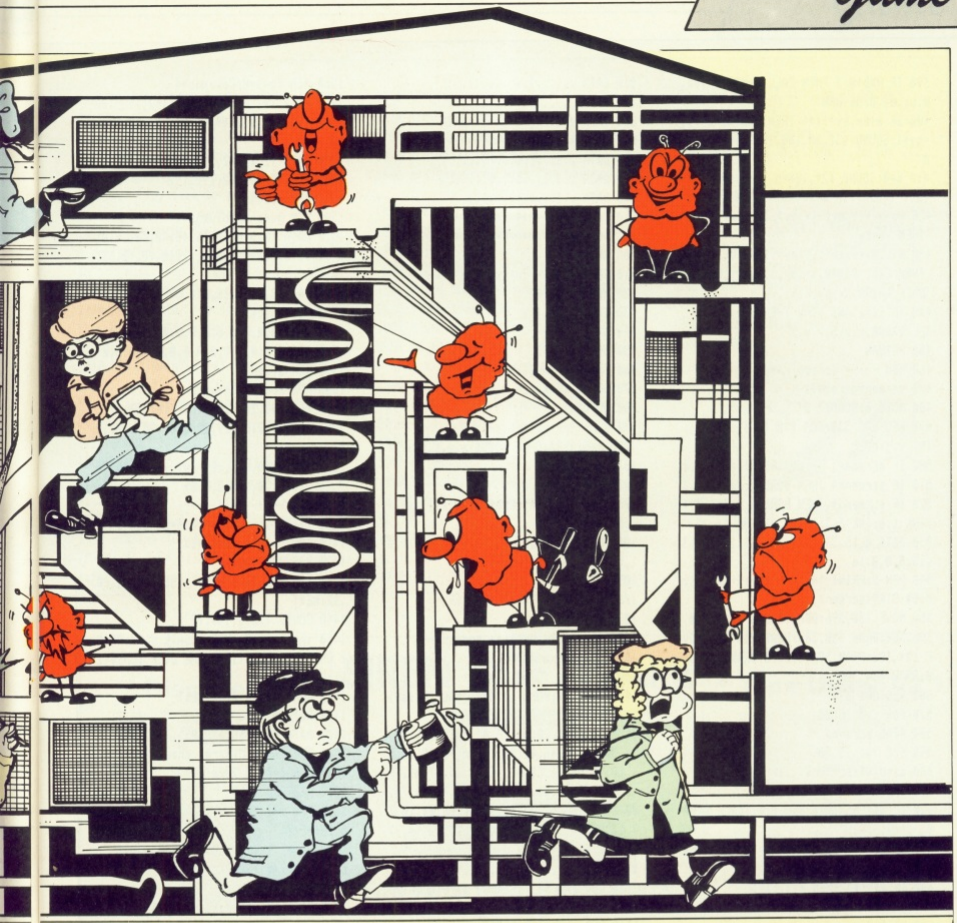
The number of Frutties increases with each screen cleared.

The game features fast, multi-coloured graphics, lively music and increasing difficulty levels. There are a couple of data statements with machine code. Be careful when entering these and remember to save the game before running it just in case you've made a typing error. The result could be fatal!

VARIABLES

x,y	Your coordinates.
b(15,1)	Frutties' coordinates.
d(15)	Whether a Frutty is dead or alive.
dead	Whether you are dead or alive.
air	How much air you have left.
screen	Screen number.
lives	Number of lives left.
m(21,25)	A character map of the screen.

```
10 REM Frutties
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 GOSUB 910:REM Initialise
50 GOSUB 1570:REM Instructions
60 WHILE NOT(TRUE AND FALSE)
70 lives=3:screen=0
80 WHILE lives
90 GOSUB 460:REM Set up screen
100 GOSUB 990:REM Start
110 WHILE NOT dead AND k(n AND air>13
4
120 GOSUB 200
```



```
130 GOSUB 310
140 WEND
150 IF dead THEN GOSUB 1390 ELSE GOSU
B 1490
160 WEND
170 GOSUB 1940:REM Game over
180 WEND
190 END
200 REM ----- Move man -----
210 PLOT air,390,0:PLOT air,392,0:air
=air-1:IF air<136 THEN dead=TRUE:RETU
RN
220 IF m(x,y)=4 THEN dead=TRUE:RETURN
 ELSE IF m(x,y+1)=4 THEN xx=x:yy=y+1:
GOTO 280
230 IF INKEY(70)>-1 THEN LOCATE x-1,y
+1:PRINT " ":m(x-1,y+1)=4:RETURN
240 IF INKEY(30)>-1 THEN LOCATE x+1,y
+1:PRINT " ":m(x+1,y+1)=4:RETURN
250 xx=x+(INKEY(39)>-1)-(INKEY(31)>-1
):IF m(xx,y)<2 THEN xx=x
260 yy=y+(INKEY(69)>-1)-(INKEY(71)>-1
):IF m(xx,yy)<2 THEN yy=y
270 IF x=xx AND y=yy THEN RETURN
280 IF m(x,y)=2 THEN CALL &A000,x,y,3
,xx,yy,1 ELSE CALL &A000,x,y,4,xx,yy,1
290 SOUND 130,500,10,10:x=xx:y=yy
300 RETURN
310 REM ----- Move bugs -----
320 FOR i=0 TO n
330 IF d(i) THEN GOSUB 360
340 NEXT
350 RETURN
360 REM ----- Bugs -----
370 bx=b(i,0):by=b(i,1):IF bx=x AND b
y=y THEN dead=TRUE:CALL &A000,x,y,5,x
,y,5:RETURN
380 IF m(bx,by)=4 THEN k=k+1:CALL &A0
00,bx,by,4,bx,by,4:d(i)=FALSE:RETURN
```

```
390 IF RND>0.3 THEN CALL &A000,bx,by,
0,bx,by,0:RETURN
400 IF m(bx,by+1)=4 THEN h=5:xx=bx:yy
=by+1:SOUND 132,50,100,10,0,1:GOTO 43
0
410 h=0:SOUND 129,100*n,5,10:xx=bx+(x
<bx)-(x>bx):IF m(xx,by)<2 THEN xx=bx
420 yy=by+(y<by)-(y>by):IF m(xx,yy)<2
 THEN yy=by
430 b(i,0)=xx:b(i,1)=yy:IF m(bx,by)=2
 THEN CALL &A000,bx,by,3,xx,yy,h ELSE
 CALL &A000,bx,by,4,xx,yy,h
440 IF xx=x AND yy=y THEN dead=TRUE:C
ALL &A000,x,y,5,x,y,5
450 RETURN
460 REM ----- Screen -----
470 screen=screen+1
480 MODE 0:BORDER 0
490 RESTORE 530:FOR i=0 TO 15:READ j:
INK i,j:NEXT
500 IF screen=3 THEN BORDER 1:INK 0,1
510 IF screen=4 THEN BORDER 3:INK 0,3
520 IF screen>4 THEN BORDER 4:INK 0,4
:INK 1,6:INK 3,3:INK 4,21:INK 11,25
530 DATA 0,15,23,6,24,26,8,2,11,20,13
,18,0,0,0,16
540 PEN 5:PRINT "Air:";TAB(15)"Scr:";
MID$(STR$(screen),2)
550 MOVE 130,396:DRAW 398,396,4:DRAW
398,386:DRAW 130,386:DRAW 130,396:MOV
E 134,392:DRAW 394,392,15:DRAW 394,39
0:DRAW 134,390
560 RESTORE 670
570 FOR i=3 TO 25
580 READ screen$
590 FOR j=1 TO 20
600 a$=MID$(screen$,j,1)
610 IF a$="." THEN m(j,i)=0
620 IF a$="=" THEN m(j,i)=1:CALL &A00
0,j,i,2,j,i,2
630 IF a$="I" THEN m(j,i)=2:CALL &A00
0,j,i,3,j,i,3
640 IF a$="*" THEN m(j,i)=3
650 NEXT
660 NEXT
670 DATA ....................
680 DATA **|*********|*******
690 DATA ==|*********|=======
700 DATA ..I.........I.......
710 DATA ..I.........I.......
720 DATA ..I.........I.......
730 DATA **|*|*******|***|***
740 DATA ====|***********|===
750 DATA ....I...........I...
760 DATA ....I...........I...
770 DATA ....I...........I...
780 DATA ****|*|*********|*|*
790 DATA ======|***********|=
800 DATA .....I...........I.
810 DATA .....I...........I.
820 DATA .....I...........I.
830 DATA **|***|******|****|*
840 DATA ==|==========|======
850 DATA ..I.........I.......
860 DATA ..I.........I.......
870 DATA ..I.........I.......
880 DATA **|*********|******
890 DATA ====================
900 RETURN
910 REM ------- Initialise -------
920 MEMORY &9FFF
930 ENT 1,200,1,1
940 DEFINT a-z
950 DIM m(21,25),b(15,1),d(15)
960 TRUE=-1:FALSE=0
970 GOSUB 1080:REM Machine code
980 RETURN
990 REM ----- Start -----
1000 x=5:y=24:CALL &A000,x,y,1,x,y,1
1010 n=screen
1020 FOR i=0 TO n
1030 b(i,0)=1+INT(RND*20):b(i,1)=4+IN
T(RND*4)*5
1040 CALL &A000,b(i,0),b(i,1),0,b(i,0
),b(i,1),0:d(i)=TRUE
1050 NEXT
1060 k=-1:air=394:dead=FALSE
1070 RETURN
1080 REM ----- Machine code -----
1090 RESTORE 1130
1100 FOR i=0 TO 80
1110 READ a$:POKE &A000+i,VAL("&"+a$)
1120 NEXT
1130 DATA 26,C0,DD,7E,0A,3D,87,87,6F,
11,50,00,DD,46,08,05,19,10,FD,DD,7E,0
6,CD,2F,A0,26,C0,DD,7E,04,3D,87,87,6F
,11,50,00,DD,46,02,05,19,10,FD,DD,7E,
00,87,87,87,87,5F,16,A1,06,08,1A,7
7,13,23,1A,77,13,23,1A,77,13,23,1A,77
,13,78,01,FD
1140 DATA 07,09,47,10,E9,C9
1150 FOR i=0 TO 191:READ j:POKE &A100
+i,j:NEXT
1160 REM alien
1170 REM Rows=8/Columns=4
1180 DATA 80,228,216,160,216,237,222,
228,218,237,222,229,207,207,207,207,1
38
1190 DATA 207,207,69,207,0,0,207,69,2
07,207,138,138,0,0,69
1200 REM man
1210 REM Rows=8/Columns=4
1220 DATA 0,16,0,0,0,48,32,0,0,16,0,0
,20,124,60,0,40,124,40,40,32,65,0,32,
0
1230 DATA 130,130,0,69,138,207,0
1240 RETURN
1250 REM brick
1260 REM Rows=8/Columns=4
1270 DATA 192,192,132,192,192,192,132
,192,192,192,132,192,192,192,132,192,
12
1280 DATA 12,12,12,192,72,192,192,192
,72,192,192,192,72,192,192
1290 REM ladder
1300 REM Rows=8/Columns=4
1310 DATA 16,0,0,32,16,48,48,32,16,0,
0,32,16,0,0,32,16,0,0,32,16,48,48,32,
16
1320 DATA 0,0,32,16,0,0,32
1330 REM space
1340 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
1350 REM alien
1360 REM Rows=8/Columns=4
1370 DATA 80,228,220,160,216,237,222,
228,218,237,218,229,207,207,207,207,2
07
1380 DATA 138,138,207,207,69,69,207,6
9,207,207,138,138,0,0,69
1390 REM ----- Dead -----
1400 lives=lives-1:screen=screen-1
1410 FOR i=0 TO 2000:NEXT
1420 RESTORE 1480
1430 FOR i=1 TO 6:READ j:SOUND 1,j,20
,12:NEXT
1440 SOUND 1,478,100,12
1450 IF lives=0 THEN LOCATE 7,12:PRIN
T CHR$(22);CHR$(1);"GAME OVER";CHR$(2
2);CHR$(0)
1460 FOR i=0 TO 3000:NEXT
1470 RETURN
1480 DATA 956,758,638,851,716,568
1490 REM ----- Cleared sheet -----
1500 FOR i=0 TO 2000:NEXT
1510 RESTORE 1560
1520 FOR i=1 TO 15:READ j:SOUND 1,j,2
0,12:NEXT
1530 SOUND 1,119,100,12
1540 FOR i=0 TO 3000:NEXT
1550 RETURN
1560 DATA 60,63,60,63,60,71,63,80,71,
89,80,95,89,106,95
1570 REM ----- Instructions -----
1580 MODE 1:CALL &BC02:CALL &BB4E:INK
 0,0:BORDER 0
1590 PEN 2:INK 2,0:LOCATE 1,25:PRINT
"Frutties!":PLOT -10,-10,1
1600 FOR i=0 TO 140 STEP 2
1610 FOR j=0 TO 16 STEP 2
1620 IF TEST(i,j) THEN PLOT 180+i*2,3
50+j*2:PLOT 180+i*2,352+j*2:PLOT 182+
i*2,350+j*2:PLOT 182+i*2,352+j*2
1630 NEXT
1640 NEXT
1650 LOCATE 1,25:PRINT SPC(10):INK 2,
20
```

```
1660 MOVE 0,0:DRAW 0,398,1:DRAW 638,3
98:DRAW 638,0:DRAW 0,0
1670 WINDOW #0,2,38,6,24
1680 PEN 2:PAPER 0:PRINT "Can you rid
 the factory of the plagueof Frutty M
onsters? They seem to be everywhere
causing havoc and mayhem."
1690 PRINT:PRINT "The factory has bee
n sealed to stop them from escaping
so you must wear oxygen tanks to hel
p you to breathe."
1700 PRINT:PRINT "To get rid of a Fru
tty, dig a hole inits path so it'll f
all into it."
1710 PEN 3:PRINT:PRINT " A=up    Z=do
wn   <=left    >=right":PRINT:PRINT
"     Caps Lock=dig hole on left":PRI
NT:PRINT"     ?=dig hole on right"
1720 GOSUB 1770
1730 WINDOW #2,2,10,2,24:WINDOW #3,11
,20,2,24:WINDOW #4,21,30,2,24:WINDOW
#5,31,39,2,24
1740 LOCATE #3,1,23:LOCATE #5,1,23
1750 FOR i=1 TO 25:PRINT #2,CHR$(11);
:PRINT #4,CHR$(11);:PRINT #3,CHR$(10)
;:PRINT #5,CHR$(10);:NEXT
1760 RETURN
1770 REM ----- Scroll -----
1780 RESTORE 1900
1790 PEN #1,1:PAPER #1,3
1800 a$="Computing With The Amstrad..
......"
1810 b$=a$
1820 SOUND 2,239,10000,10:SOUND 4,319
,10000,10
1830 WHILE INKEY$()"":WEND
1840 WHILE INKEY$="":READ j:IF j=0 TH
EN RESTORE 1900:READ j
1850 SOUND 1,j,15,12:LOCATE #1,6,24:P
RINT #1,MID$(b$,1,30):b$=MID$(b$,2):I
F LEN(b$)<31 THEN b$=b$+a$
1860 WEND
1870 SOUND 129,0,0,0:SOUND 130,0,0,0:
SOUND 132,0,0,0
1880 x=FRE("")
1890 RETURN
1900 DATA 60,63,60,63,71,63,71,80,71,
80,89,80,89,95,89,95,106,95,106,119,1
06
1910 DATA 239,213,190,179,159,142,127
,119
1920 DATA 190,179,159,142,127,119,106
,95, 159,142,127,119,106,95,89,80
1930 DATA 0
1940 REM ----- Scores -----
1950 MODE 1:INK 0,13:INK 1,26:INK 2,2
0:INK 3,2
1960 DRAW 0,398,1:DRAW 638,398:DRAW 6
38,0:DRAW 0,0
1970 LOCATE 16,5:PEN 2:PRINT "Well do
ne!":PEN 3:LOCATE 11,10:PRINT "You go
t to screen";screen+1
1980 PEN 1:LOCATE 7,15:PRINT "Press a
ny key for another game"
1990 GOSUB 1770:REM scroll
2000 RETURN
```

Give your fingers a rest . . .

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

See our special offer on Page 101.

One of America's most popular games

THE ULTIMATE IN BATTLE ACTION...

BEACH-HEAD™

C64 AMSTRAD

- Incredible 3-D Graphics
- Unbelievable Sound Effects
- Unique Games Concept
- Multiple Screens
- High Resolution Scenario
- 100% Machine Language

Voted by U.S. Billboard magazine as the best game for sound and graphics on the Commodore 64

It's a unique arcade experience in sound and vision and a stunning display of Amstrad capabilities.

Another quality product from ***ACCESS*** Software.

Available on **CASSETTE** **£9·95** for **AMSTRAD**

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

THE ULTIMATE IN AMERICAN SOFTWARE FOR YOU ON U.S. GOLD

U.S. Gold is stocked by all leading computer stores including:

BOOTS WHSMITH JOHN MENZIES WILDINGS WOOLWORTH

DEALERS! For information on how to become a U.S. Gold Stockist write to: CentreSoft Ltd., Unit 10, The Parkway Industrial Centre, Heneage Street, Birmingham B7 4LY. Telephone: 021-359 3020. Telex: 337268. Overseas enquiries welcome.

U.S. GOLD

All American Software

Part X of MIKE BIBBY's introduction to machine code

Relative branching doesn't relate to family trees

THIS month we're going to learn some more about representing numbers in machine code.

If you take a look at Program I you'll find it's not too difficult to follow – we're just adding 6 to &FF (255) and storing the answer in &2FF8.

When you examine &2FF8, it won't surprise you to learn that it

```
address  hex code  mnemonics
3000     3E 06     LD A, &06
3002     C6 FF     ADD A,&FF
3004     32 F8 2F  LD (&2FF8),A
3007     C9        RET
```

Program I

contains 5. If you think of 6 as 1+5, the sum you've done is:

255 + 1 + 5

Now 255 + 1 takes you round the clock to 0, so the above sum boils down to:

0 + 5 = 5

In fact, what happens is that you end up with one less than you started with in the A register. When you think about it this will happen whenever you add &FF to the A register no matter what's in it.

If you like, the &FF grabs 1 from the number to complete its cycle round to zero, leaving the number in the A register one less.

So to take 1 away from a number, you just add &FF to it. You can extend this idea – to take 2 from a number, you just add &FE (254). The &FE will grab 2 from the number to complete its cycle round to 0, leaving the number in the A register 2 less.

For instance:

30 + 254
= 28 + 2 + 254
= 28 + 0
= 28

Try it out in practise, the codes are shown in Program II.

```
address  hex code  mnemonics
3000     3E 1E     LD A,30
3002     C6 FF     ADD A,&FE
3004     32 F8 2F  LD(&2FF8),A
3007     C9        RET
```

Program II

You won't be surprised to learn that to take 3 from a number you add &FD (253) to and so on.

We can use these ideas to give us a rudimentary negative number system. So far all the numbers we've been handling have been positive (0 to 255). Sometimes however, it's useful to have negative numbers (–1, –2, –3 and so on). I find them particularly useful for dealing with my bank account.

We've seen that adding &FF to a number is in effect taking one away from it. We can therefore consider &FF to represent –1 in our machine code world. &FE would then be –2 and so on.

Note the words "consider" and "represent" – what we're doing is absolutely arbitrary. Don't get schizoid trying to decide if &FF is –1 or 255 – it depends on context, and more often than not we choose that.

You should be used to looking at things in two ways by now. After all you're sure to accept that:

10

can mean either 10 or 16 depending on whether the context is decimal or hexadecimal. Similarly X can mean a consonant, a multiplication, wrong, 10 in Roman numerals, or can simply mark the spot!

And the system works quite well, even allowing us to do sums where the answer is itself negative. For instance 3–4 translates to:

3 + 252
= 255

and 255 represents –1, the correct answer.

However if you follow the logic inexorably down the line, you'll find you get to the stage where &01 represents –255 (think about it). The trouble with this is that we've not left ourselves any positive numbers to play with.

The compromise we'll use is to use the numbers &FF to &80 to represent the negative numbers –1 to –128 respectively, while &0 to &7F represent the normal numbers 0 to 127. Figure I shows the scheme.

Note that we're still using a byte to store 256 numbers, but instead of

Positive		Negative	
0	&0	-1	&FF
+1	&1	-2	&FE
+2	&2	-3	&FD
.	.	.	.
.	.	.	.
.	.	.	.
+126	&7E	-127	&81
+127	&7F	-128	&80

Figure I: How negative numbers are represented

having 0 to 255 – all positive – half of them are negative and half positive.

There's something else to notice

too – the way we've rigged things all the negative numbers have their top bit (bit 7) set, while all the positive numbers have bit 7 clear. As we'll see, this can be very useful.

All this might seem rather complicated, and doing sums with these signed bytes, as they are known, can be tricky until you get used to it. I suggest you don't try until you're fairly experienced at machine code.

However there is a use that we can put this signed arithmetic to – relative branching, which has nothing to do with family trees. Take a look at Program III.

Here we simply JumP to location &3006 and RETurn – so nothing fancy happens. The NOPs, by the way (operand &00) are the micro's shorthand for "do nothing" – they're only in the program to give it something to jump over.

Now take a look at Program IV. It does exactly the same as Program III but uses JR (opcode &18) for Jump Relative instead of JP.

Before we go into this, we ought to introduce the idea of a program counter. The best way to think of this is as a marker telling the Z80 where it's up to in a program. In fact it's a 16 bit register holding the address of the current byte of the program.

After all, machine code doesn't have line numbers like a Basic program. So, since the program itself is stored separately in memory, the Z80 monitors how far it's got by keeping track of the location it has reached in the program counter.

address	hex code	anemonics
3000	C3 06 30	JP &3006
3003	00	NOP
3004	00	NOP
3005	00	NOP
3006	C9	RET

Program III

address	hex code	anemonics
3000	18 03	JR here
3002	00	NOP
3003	00	NOP
3004	00	NOP
3005	C9	here RET

Program IV

Although not strictly accurate it's easiest to think of the program counter as pointing to the address of the next instruction (or opcode) that the Z80 has to perform.

Normally the Z80 just goes along in a straight line incrementing the program counter by the required number of bytes for each opcode, so that it points at the next opcode to be done.

However when it meets a JP this orderly progress stops. The Z80 loads the address you've specified after the JP into its own program counter so a JP &3006 will cause the program

counter to be loaded with &3006.

Now the whole point of a program counter is that the micro gets the address of the next opcode from it. So having finished the JP instruction, the Z80 gets its next instruction from the address given in the program counter – which is the address you've specified in the JP. So the Z80 now starts the code from the new address. We say the program has jumped or branched to this new location.

What we're using in Program IV is a shorthand way of specifying a jump by using a relative branch. In other words, instead of going to a definite address, as in: "Pop this letter in the door of number 8", we're saying: "Pop this letter in next door but one".

The first is an absolute address since you have fully specified the house you want – number 8. The second is, of course, relative to where you are. If you are in number 112, the letter won't go to number 8 that's for sure.

However, providing you do know where you are, relative addressing works fine.

In the case of a relative jump (JR) a one byte offset is given after the opcode – which is added to the contents of the program counter to form a new address in the program counter. The program then branches to this new address.

In Program IV the opcode 18 is followed by 03 which means we add 3 bytes to the program counter to decide where we carry on from.

Yes, you might say, but doesn't that make us jump to &3000 + 3 – &3003? If we want it to be the same as Program III shouldn't it go to &3005?

Quite right, and it does! Remember the way to think of the program counter is as pointing to the next opcode, which in this case is the first NOP, at &3002. So when we add our offset of 3, we get the address we want to jump to, &3005.

The trick to using relative jumps is in calculating the offset. The way to do it is to write your JR as:

18 __

You can fill in the underline with the offset later. Then, starting as the byte following these two, which is the next opcode, count 0, 1, 2, and so on, for each byte up to and including the byte you want to branch to.

The number you finished at goes in where you put the blank line. Notice the way I've used the label *here* after the JP and before the destination byte – meaningful tags are vital for keeping track of programs.

Just as you can branch forward, you can also branch back. This time you'll want to subtract something from the program counter since we're branching to earlier in the program, so we need our negative numbers.

Take a look at Program V – but don't run it as it puts you in an endless

```
address  hex code     mnemonics
  3000     00       there NOP
  3001     00             NOP
  3002     00             NOP
  3003     18 FB          JR there
  3005     C9             RET
```

Program V

loop. After the Z80 encounters the JR and before the offset is added, the PC will be pointing to the RET (&3005).

To get it to point at *there* (&3000) we need to take 5 bytes off the program counter. Using the negative numbers convention we need to branch back &FB bytes, since &FB is equivalent to –5. Hence JR translates into:

18 FB

To calculate backward branches, rather than remembering all the negative numbers, I start counting down from zero – &0, &FF, &FE and so on – starting at the first instruction after the branch bytes, and finishing at the destination byte.

Of course the two branches we've created so far have been fairly useless. However just as JP could have conditions tacked on to it as in JP Z and JP NC, so you have conditional branching. This gives you a lot more scope. Table I shows the required opcodes.

We can replace all the jumps we've used so far in the series by relative branches. Program VI is the relative branching version of our first loop. As homework try replacing the JPs in the last two month's programs with JRs, but take care calculating your branch offsets.

```
address  hex code     mnemonics
  3000     3E 20          LD A,&20
  3002     CD 5A BB  here CALL &BB5A
  3005     C6 01          ADD A,1
  3007     30 F9          JR NC,here
  3009     C9             RET
```

Program VI

Don't think that JP is redundant though. Relative branches, because they use our negative number convention, can only branch forward a maximum of 127 bytes and back 128 bytes. JPs don't have these restrictions.

If you think all these calculations seem a bit tedious, you're right! However it's a good idea to get some practice in this way.

Next month we'll talk about using an assembler, which not only does all the calculations for you, but lets you use labels.

So, for additional homework, if you haven't already got an assembler get one. We published a fine assembler written by Roland Waddilove in the July issue of *Computing with the Amstrad,* so get typing.

Oh yes, last month I asked you to alter Program XII to print out an upright triangle of asterisks. Program VII shows how it's done – and I've used relative jumps.

```
JR       &18
JR NZ    &20
JR Z     &28
JR NC    &30
JR C     &38
```

Table I: Relative jump opcodes

```
address  hex code     mnemonics
  3000     0E 01          LD C,1
  3002     41        back LD B,C
  3003     3E 2A     over LD A,&2A
  3005     CD 5A BB       CALL &BB5A
  3008     05             DEC B
  3009     C2 03 30       JP NZ,over
  300C     3E 0A          LD A,&0A
  300E     CD 5A BB       CALL &BB5A
  3011     3E 0D          LD A,&0D
  3013     CD 5A BB       CALL &BB5A
  3016     0C             INC C
  3017     79             LD A,C
  3018     FE 09          CP &09
  301A     20 E6          JR NZ,back
  301C     C9             RET
```

Program VII

ACCESS ORDERS NOW TAKEN

Haystack Peripherals

Amstrad Software

Haystack Peripherals

8 Midgrove, Delph, Oldham OL3 5EJ. Tel: 04577 5229

Access

DISC DRIVES

	Our Price
DD-1	154.95
FD-1	96.95

SPEECH SYNTHS

	Our Price
Amstrad	26.95
DK'Tronics	34.95

JOYSTICKS

	Our Price
Pro Ace	11.95

SUPERPOWER ROM SOFTWARE

	Our Price
Sideways ROM Card	36.95
Ass/Disass/Mach Code	36.95
Database*	36.95
Disc User Utility	36.95
Mailing List	36.95
Prog. Toolbox	36.95
Word Processor*	36.95

DISCS

	Our Price
3" (Box of Ten)	44.95
3" (Single)	4.75

LIGHT PENS

	Our Price
Amstrad	17.95
DK'Tronics	24.95

CASSETTES

	Our Price
C15 (Box of Ten)	4.50

	Our Price Cass	Disc
ADDICTIVE		
Football Manager	7.95	
ADVENTURE INTERNATIONAL		
Gremlins	7.95	
Robin of Sherwood	8.95	
ALLIGATA		
Blagger	7.95	
Defend or Die	6.95	
ACTIVISION		
Ghostbusters	9.49	
Master of the Lamps	8.99	
ANIROG		
Jump Jet	8.95	
Slapshot	8.95	
Trivia	8.95	
AMSOFT		
Admiral Graaf Spee	7.95	
Airwolf	7.95	
Alex Higgins Pool	7.95	11.45
Alex Higgins Snooker	7.95	11.45
Alien	7.95	
Alien Break In	7.95	
American Football	8.95	
Amscalc	17.95	
Amsgolf	7.95	11.45
Amsword (Adv)	17.95	21.45
Amsword (Easy)	7.95	
Astro Attack	7.95	
Bridge It	7.95	
Catastrophies	7.95	
Centre Court	7.95	11.45
Classic Adventure	7.95	
Classic Racing	7.95	
Codename Mat	7.95	11.45
Concise Basic Spec	17.95	
Concise F/ware Spec	17.95	
Crazy Golf	7.95	
Cubit	7.95	
DDI 1 F/ware Spec	8.95	
Decision Maker	22.45	26.95
Detective	7.95	
Devpac	22.45	26.95
DFM Database	13.45	16.95
Dragons Gold	7.95	
Electro Freddy	7.95	11.45
Entrepreneur	22.45	26.95
Fantastic Voyage	7.95	11.45
Forth	22.45	26.95
Frank n'stein	7.95	
Fruit Machine	7.95	
Galactic Plague	7.95	
Gate Crasher	7.95	
Grand Prix Driver	7.95	
Guide to Basic I	17.95	
Guide to Basic II	17.95	
Guide to Logo	8.95	
Harrier Attack	7.95	11.45
Haunted Hedges	7.95	
Home Budget	17.95	
Home Runner	7.95	
Hunchback	7.95	11.45
Hunter Killer	7.95	11.45
Invostat	26.95	29.50
Jammin'	7.95	11.45
Jet Boot Jack	7.95	
Key Factor	7.95	
Laser Warp	7.95	
Loopy Laundry	7.95	
Masteralc	22.45	26.95
Master Chess	7.95	11.45
Masterfile	22.45	26.95
Micropen		44.00
Microscript		44.00
Microspread		44.00
Mutant Monty	7.95	11.45
Nominal Ledger		29.50
Oh Mummy	7.95	
Pascal	31.45	35.95
Pitman Typ. Tut.	8.95	12.45
Prize	7.95	
Project Planner	22.45	26.95
Punchy	7.95	11.45
Purchase Ledger		29.50
Quackajack	7.95	11.45
Rally II	7.95	
Roland Ahoy	7.95	11.45
Roland/Caves	7.95	11.45
Roland/Digging	7.95	11.45
Roland/Ropes	7.95	11.45
Roland/Run	7.95	11.45
Roland/Space	7.95	11.45
Roland/Sq. Bashing	7.95	
Roland/Time	7.95	11.45
Scout Steps Out	7.95	
Screen Designer	13.45	17.95
Snooker	7.95	11.45
Space Hawks	7.95	11.45
Spannerman	7.95	11.45
Splatt	7.95	11.45
Star Commando	7.95	
Star Watcher	17.95	21.45
Stock Aid	26.95	29.50
Stockmarket	7.95	11.45
Subterranean Stryker	7.95	
Sultans Maze	7.95	
Traffic	7.95	
Transact	26.95	29.50
Xanagrams	7.95	
3D Grand Prix	8.95	
3D Invaders	7.95	
3D Stunt Rider	7.95	
ARIOLASOFT		
Archon	9.45	
Hard Hat Mack	7.95	
One on One	7.95	
ARTIC		
World Cup	6.95	
BOURNE EDUCATIONAL		
Animal Vegetable Min.	7.95	13.45
Happy Letters	7.95	13.45
Happy Numbers	7.95	13.45
Happy Writing	7.95	13.45
Map Rally	7.95	13.45
Osprey	8.95	14.45
Timeman One	7.95	13.45
Timeman Two	7.95	13.45
Wordhang	7.95	13.45
Worldwise	7.95	13.45
CCS		
Arnhem	8.95	
Warzone	5.95	
CDS		
Steve Davis Snooker	6.95	
C P SOFTWARE		
Backgammon	7.95	
Bridge Player	8.95	
Draughts	7.95	
Pinball Wizard	7.95	
Superchess	8.95	
3D Voice Chess	11.45	
CRL		
Amstrad Artist	8.95	
Formula One	6.95	
Handicap Golf	5.95	
Juggernaut	6.95	
Rocky Horror Show	8.95	
Smugglers Cove	4.95	
Test Match	5.95	
CAMEL		
Flexifrend	6.50	
Grasp	7.50	
DATABASE		
Mini Office	4.95	
Red Arrows	7.95	11.45
DESIGN DESIGN		
Dark Star	6.95	
On the Run	6.95	
Tank Busters	6.95	
DIGITAL INTEGRATION		
Fighter Pilot	7.95	12.45
DK'TRONICS		
Minder	8.95	
DOMARK		
View To A Kill	9.49	
Codename Mat II	7.95	
DURELL		
Combat Lynx	7.95	
Death Pit	6.95	
ELITE		
Frank Bruno Boxing	7.95	
Int. Basketball*	7.95	
Scoobydoo*	7.95	
GORGOYLE		
Dun Darach	8.95	

	Our Price Cass	Disc
GEMINI		
Cashbook (inc. VATfile)	53.95	58.45
Database	17.95	22.95
Final Accounts	53.95	58.45
Home Accounts	17.95	22.95
Report Generator	17.95	22.95
VAT File	17.95	22.95
Combo Pack (Cashbook + Final Acc.)	80.95	89.95
GILSOFT		
The Illustrator	14.95	20.45
The Quill	14.95	20.45
GLOBAL		
Magicicans Ball	6.95	
GREMLIN		
Project Future	7.95	
Rocco	7.95	
HEWSON		
Dragontorc	6.95	
HISOFT		
C	31.45	35.95
Devpac 80		35.95
Ed 80		26.95
Font 64	7.95	11.45
Pascal 80		35.95
The Knife		11.45
The Torch		11.45
IMAGINE		
W. Series Baseball*	7.95	
INCENTIVE		
Confuzion	5.95	
Millionaire	5.95	
INTERCEPTOR		
Azimuth	7.95	
Fig Forth	8.95	
Forest at Worlds End	5.00	
Heroes of Karn	5.00	
Jewels of Babylon	5.00	
Message from Androm.	5.00	
Warlord	5.00	
INTRIGUE		
Tomb of Kustak	6.50	
KNIGHTSOFT		
Anim. Strip Poker	7.95	
KOSMOS		
French Mistress 'A'	7.95	11.45
French Mistress 'B'	7.95	11.45
German Master 'A'	7.95	11.45
German Master 'B'	7.95	11.45
Spanish Tutor 'A'	7.95	11.45
Spanish Tutor 'B'	7.95	11.45
KUMA		
Artwork	8.95	
Database	13.45	
Forth	17.95	
Fruity Frank	5.95	
Galaxia	4.95	
Home Budget	17.95	
Music Composer	8.95	
Star Avenger	5.95	
LEISURE GENIUS		
Scrabble	8.95	
LEVEL 9		
Adventure Quest	8.95	
Colossal Adventure	8.95	
Dungeon Adventure	8.95	
Emerald Isle	5.95	
Lords of Time	8.95	
Red Moon	5.95	
Return to Eden	8.95	
Snowball	8.95	
Worm in Paradise	8.95	
LLAMASOFT		
Psychedelia	5.00	
LOTHLORIEN		
Johnny Reb	5.95	
Red Coats	5.95	
Special Operations	5.95	
MARTECH		
Brian Jacks Superstar	6.95	
MELBOURNE HOUSE		
Hobbit	12.95	
Mordon's Quest	5.95	
Starion	8.95	
Exploding Fist	8.95	
MICROPOWER		
Ghouls	5.95	

	Our Price Cass	Disc
Killer Gorilla/Gaunt.	8.95	
MIKROGEN		
Everyones a Wally	8.95	
Herberts Dummy Run*	8.95	
Pyjamarama	7.95	
MIRRORSOFT		
Ashkeron	5.95	
Boulderdash	8.95	
Caesars Travels	7.95	
First Steps/Mr Men	7.95	
Here and There/Mr Men	7.95	
MOSAIC		
Secret Diary/Adrian Mole	8.95	
MYRRDIN		
Flight Simulation	10.45	
NEW GENERATION		
Machine Code Tutor	13.45	
OCEAN		
Daley Thom. Decath.	7.95	
Hunchback II	7.95	
PSS		
Battle for Midway	8.95	
Covenant	5.95	
Macadam Bumper	6.95	
PICTURESQUE		
Code Machine	17.95	
R & R		
Froggy	6.95	
REALTIME		
3D Starstrike	5.95	
SHARDS		
Mystery/Java Star	6.95	
SIREN		
Discovery	6.95	10.45
Tape Utility	5.95	9.45
SOFTWARE PROJECTS		
Blinky	6.95	
Jet Set Willy	7.95	
Manic Miner	7.95	
SUPERSOFT		
Interdictor Pilot	15.95	
SYSTEM 3		
Intern. Karate	5.50	
TASKSET		
Souls of Darkon*	7.90	
Superpipeline II	7.95	
TASMAN		
Tascopy	8.90	11.40
Tasprint	8.90	11.40
Tasword	17.95	22.45
THE EDGE		
Brian Bloodaxe	7.95	
ULTIMATE		
Alien 8	8.95	
Knightlore	8.95	
U.S. Gold		
Beach Head	8.95	11.45
Bruce Lee*	8.95	
Raid	8.95	
VIRGIN		
Sorcery	7.95	12.45
VORTEX		
Android I	6.95	
Android II	6.95	
Tornado Low Level	6.95	
WINTERSOFT		
Ring of Darkness	8.95	

** Please ring to check availability*
Overseas orders – Please send Sterling money orders

DUE TO LACK OF SPACE WE NO LONGER QUOTE RECOMMENDED RETAIL PRICE ALONGSIDE OUR PRICE.
All our prices have been reduced as follows:

RRP	Our Price
Up to £10	£1 off
£10-£15	£1.50 off
£15-£20	£2 off
£20-25	£2.50 off
etc	(except Disc Drives and Discs)

E & OE

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

Please Send: Cass/Disc | Price

TOTAL

ALL OUR PRICES INCLUDE VAT AND CARRIAGE

Cheques (Please write card no. on reverse)/PO's to:
HAYSTACK PERIPHERALS
8 Midgrove, Delph, Oldham OL3 5EJ
Tel: Saddleworth (04577) 5229

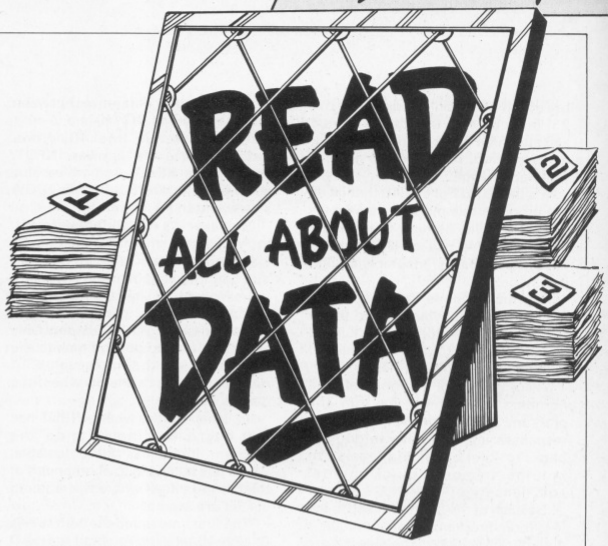
NAME .....

ADDRESS .....

.....

Phone No. .....

Access No. .....



THIS month we're leaving strings behind us and looking at how we can use READ and DATA to give values to variables. This may not sound too exciting, but by the end of the article you should be able to see how useful it can be.

To start with, have a look at Program I. All this does is to give, or assign, values to the numeric variables *x, y,* and *z,* add them and print the answer. It's as easy as 1, 2, 3.

```
10 REM Program I
20 LET x=1
30 LET y=2
40 LET z=3
50 LET sum=x+y+z
60 PRINT sum
```

Program I

All right, so it's hardly going to win the Program of the Year award, but what it lacks in complexity it makes up in efficacy. It works and it makes a point.

What I want you to notice is the program's rigidity. If I now wanted to add 6, 7 and 8 using the same structure I'd have to rewrite lines 20 to 40. As you can see from this, assigning values to variables using simple LET statements can be fairly inflexible.

Of course there are other ways of assigning values to variables. We've come across two of them already. Take a look at Program II, which uses the less obvious of the two methods.

```
10 REM Program II
20 sum=0
30 FOR loop=1 TO 3
40 sum=sum+loop
50 NEXT loop
60 PRINT sum
```

Program II

Here we're using the (I hope) familiar FOR . . . NEXT loop to add 1, 2 and 3. The loop cycles three times.

The first time round, *loop* is equal to 1 while *sum* is initially 0. The result is that line 40 stores 1 (1+0) in *sum.* Next time round *loop* is 2, while *sum* starts off as 1 and ends up as 3. I leave it to you to figure out what happens to *sum* during the third cycle when *loop* is 3.

So, we've used *sum* to hold the running total of successive values of *loop* and hence added 1, 2 and 3.

Incidentally, you don't really need line 20, the Amstrad automatically assumes that numeric variables are 0 until it's told otherwise. Not all micros do this, however, and some come crashing to a halt if you haven't already given a variable a value, even if it's only zero.

In Part 10 of our series for beginners PETE BIBBY looks at how to give values to variables

So, even if line 20 is redundant in this case, it's a good habit to get into. And your Amstrad's got enough memory to cope with a little bit of redundant code.

While all this may seem a bit longwinded when just adding 1, 2 and 3, try adapting Program II to add up all the integers (whole numbers) from 1 to 1000. You'll see that it beats the first program's method hands down.

The trouble is that while using a loop control variable to give values to a running total is both efficient and adaptable, it is a bit limited. If you think about it you'll see why.

Since the loop control variable increases by the same amount each time round, the values of *loop* are in a regular pattern. It's easy to add, say 1, 3 and 5 or 4, 8, 12 and 16 using this method. All you do is vary the STEP parameter of the loop to allow the loop control variable to take the required values. That's easier to do than to read about! In the case of Program II you'd use:

```
30 FOR loop=1 TO 5 STEP 2
```

in the first case and:

```
30 FOR loop=4 TO 16 STEP 4
```

for the second.

The problem arises when you try to use the method to add 1, 7 and 23. So while Program II may be better than Program I, it still has its drawbacks.

Program III shows a much more flexible method of getting information into a program. It has you actually typing it in at the keyboard when the Amstrad requests it.

When you run Program III you'll

```
10 REM Program III
20 sum=0
30 FOR loop=1 TO 3
40 INPUT "Number? ",number
50 sum=sum+number
60 NEXT loop
70 PRINT sum
```

Program III

see how it can handle adding 1, 7 and 23. In fact it's so flexible that it can add any three numbers that you think of, provided that they're in the Amstrad's range.

It's the INPUT of line 40 that gives the program this marvellous adaptability. However nothing in life is all good, and this use of INPUT does have its drawbacks, adaptable though it may be.

The major one is that it holds up the program until you respond to the request for keyboard input. And imagine trying to add a thousand numbers using this method . . .

Also you have to input all the numbers every time you run the program. One error typing in your responses and you have to go right back to the beginning again, or indulge in some quick mental arithmetic.

So each of the methods used in our first three programs seems to have a drawback. Wouldn't it be nice if there was a way to give values to variables that was flexible, would take any numbers, and wouldn't involve keying things in while the program is running?

Have a look at Program IV, which meets all these criteria.

```
10 REM Program IV
20 sum=0
30 FOR loop=1 TO 3
40 READ number
50 sum=sum+number
60 NEXT loop
70 PRINT sum
80 DATA 1,2,3
```

Program IV

As you can see, it has added up the numbers in the last line, our old favourites 1, 2 and 3, and printed out the result. What's interesting is the way in which it's done it.

Before we get to that, let's have a look at the familiar bits of Program IV. As before, line 20 isn't strictly needed but it is good programming practice. Lines 30 and 60 are identical to those of Program III, forming a FOR . . . NEXT loop that cycles three times.

Even line 50 is the same, using the variables *sum* and *number* to hold a running total. And line 70 just prints out the value of *sum* when the program drops out of the loop. So what's the big difference between Programs III and IV?

The answer is line 40. Here, instead of the old familiar INPUT, we've got a READ. And we've also got a new line, line 80, with its DATA statement and 1, 2 and 3.

However it's not all that different. In Program III, the INPUT of line 40 was there to get a value for *number*. You had to type it in at the keyboard.

In Program IV line 40's READ does exactly the same thing, getting a value and storing it in *number*. Only now the Amstrad doesn't look to the keyboard for the next value of *number*, it looks to line 80 where the data is held.

In other words, where INPUT has the micro looking outside the program for values for a variable, READ has it looking to another part of the program itself where the required values are stored.

The first time round the loop *sum* is 0. Line 40 tells the Amstrad to READ a value from the data line and put it in the variable *number*.

Obediently the micro searches through the program for the first occurrence of a line beginning with DATA and takes the first value it finds there. In this case it is 1. Then the program goes onto line 50 which works in the usual way, storing the running total, 1, in *sum*.

The second cycle of the loop sees *sum* starting off with the value 1. Line 40 again tells the computer to READ a value for *number*. As before, it looks to the line beginning with DATA, but this time it takes the second item it finds, in this case 2. So *number* becomes 2 and, after line 50, *sum* becomes 3.

The third time round the loop the READ of line 40 has the Amstrad looking for the third item in line 80's data list, the figure 3. This is stored in *number* and line 50 sees *sum* becoming 6. The loop is now ended and the PRINT of line 70 gives the answer.

Notice that line 80 – the one beginning with DATA – does nothing but hold the data. Without a READ somewhere to make use of it, it's barren.

To sum up, READ causes the program to look to a line beginning with DATA and to store the values that it finds there in the variable that follows the READ command. The micro keeps track of where it is up to in the data list and every time it obeys a READ command it takes the next unused value.

In other words, it moves sequentially along the list of data, never using the same one twice.

As we said before, the READ command works exactly like the INPUT command except that instead of looking at the keyboard the computer looks in the program itself for the value to be assigned to the variable.

In a way it's a combination of the best features of all three of the above methods, but without sacrificing any flexibility.

This adaptability comes from the fact that if we want to give the program different values all we have to do is to change the numbers after the DATA of line 80. So to add up 56, 78 and 345 we would alter it to read:

```
80 DATA 56,78,345
```

while:

```
80 DATA 11,22,33
```

gives us the sum of 11, 22 and 33.

We don't have to have all of our data in one line. Program V uses READ and DATA to add up 10 numbers.

```
10 REM Program V
20 total=0
30 FOR cycle=1 TO 10
40 READ figures
50 total=total+figures
60 NEXT cycle
70 PRINT total
80 DATA 12,5,34,6,7
90 DATA 3,67,54,1,2
```

Program V

You should have no problem seeing that the method used is the same as in Program IV. However as we are now adding up 10 numbers the loop cycles 10 times, each time READing in more data.

Notice that now there are two lines beginning with DATA, lines 80 and 90. This is easier for us to follow, two lines of five numbers looking better than one of 10.

To the Amstrad it makes no difference. When it comes across a READ it starts at the first item of data

after the first DATA it finds and then carries on doggedly in sequence. In this case it READs 12, then 5, 34, 6 and 7 in order. The sixth time round the loop line 60's READ demands another value for *figures*.

Undaunted by having reached the end of line 80, the Amstrad carries on, finds the next line beginning with DATA, and takes the first value following that. In this case the line is line 90 and the first value is 3. The next four cycles of the loop have the micro READing in 67, 54, 1 and 2 in turn.

You'll have noticed from the last two programs that the commas in the lists of data act as separators. They come between the numbers, telling the micro where one item of data ends and another one starts. Obviously they're very important and if one is left out or misplaced the consequences can be dire.

To see what I mean, try changing line 80 of Program V to:

```
80 DATA 125,34,6,7
```

This is what would result if we missed out the comma between the 12 and 5 when typing in the program. Now when you run it you get the message:

```
DATA exhausted in 40
```

and the program stops.

What has happened is that the FOR . . . NEXT loop has tried to cycle 10 times as directed. The first nine times there's no problem as there are, after our mis-typing, nine items in the data list.

The tenth time round the loop, line 40 tells the micro to READ another value from the data list. Unfortunately when the micro goes to line 90 the cupboard is bare. There's no data item available.

This leaves the micro in a quandry. It's been told to take 10 items from the list but there are only nine there. The result is a nervous breakdown and the program grinds to a halt with the aforementioned:

```
DATA exhausted in 40
```

message as some form of explanation.

But it's not a very good explanation. It points to the wrong line. After all, we made the mistake in line 80 and the program ran out of data items at the end of line 90, yet the message says the problem is at line 40.

In a way it's true, because the problem occurred when the micro tried to obey the READ of line 40 for the tenth time. So it can be justified.

However it's still a nuisance. Half the problems people have with typing in programs from *Computing with the Amstrad* occur when they mis-type something in DATA statements. The program crashes and the error message points to a READ that's miles away from the offending data.

The poor typist keeps looking at the READ and knows that it's correct even though the message tells him the fault lies at line 40. Hence they give up in despair, assuming the program doesn't work.

The moral is, if you come across an error message that points to a line with READ in it, check out the DATA statements carefully.

And if that potentially misleading error message wasn't enough, you can make another typing mistake in a DATA line that doesn't give a message at all. To see what I mean try changing line 80 to:

```
80 DATA 1,2,5,34,6,7
```

Here, instead of leaving out a comma and "glueing" two numbers together as before, our errant typist has added a comma between the 1 and the 2 of what was supposed to have been 12. Now if you count the number of data items in lines 80 and 90 you'll find there are 11. But the loop is only going to cycle 10 times so what's going to happen? Run the program and see.

The result is that the program works but it gives the "wrong" result. Actually the result isn't really wrong, you'll find that 180 is the result of adding together the first 10 items of data. The trouble is that the first 10 items of data weren't the 10 items of data we wanted to add! Our slipshod typist has messed it up.

This kind of mistake can be a real problem. Where there are too few items of data in a list then the Amstrad will tell you as it did before. The error message may be a bit misleading, but it will be there.

However if there are more data items than expected, all the micro does is take the number it wants and ignores the rest. There's no error message.

In our case the program returned the value 180 from the faulty data, instead of the 191 we'd have got if the typist hadn't been daydreaming. The computer has done its sums correctly, but it's done them with the wrong numbers. And without any error message it's easy for the mistake to slip through.

There's another problem that doesn't show up in this short program. The Amstrad keeps track of where it is up to in the data list by means of what is known as a data pointer. All this does is to "point" to the next item of the data list to be used. Now when our typist has made line 80:

```
80 DATA 1,2,5,34,6,7
```

Program V appears to work correctly and stops. The data pointer is pointing at the next item in the data list, 2, but as the program has ended, it is never used.

In longer programs, however, there can be more than one READ command. If Program V did have another READ command tucked away somewhere then the first data item it would take would be the one that the data pointer is indicating, that is, 2.

Because of the inadvertent comma all the succeeding READs will be one data item out. This can cause havoc in a long program and can be very hard to track down.

The moral of the above is type in DATA lines carefully.

So far we've had our data lists at the end of the program. However they don't have to be there, as Program VI shows.

Here the DATA lines are all over

```
10 REM Program VI
20 accumulator=0
30 DATA 12,23
40 FOR loop=1 TO 7
50 DATA 34,45,56
60 READ fromsomewhere
70 accumulator=accumulator+fromsomewh
ere
80 NEXT loop
90 DATA 67,78
100 PRINT accumulator
```

Program VI

the place. This makes no difference to the Amstrad as its data pointer keeps track of things. However it's good programming practice to keep your data lists together. This is usually at the end of short programs or, in the case of longer programs which might have several different data lists, in logically appropriate places.

Hence you may find all the data on salaries with the subroutine that calculates pay and all the data on employees' names and addresses near the subroutine that prints out the payslips.

While you've got Program VI in your micro, change line 50 so that it reads:

```
50 DATA 34,45,56,
```

that is, add a comma after the 56. Now run the program and see what happens. Instead of our previous total of 315 the result is 237. Yet we're still using the same data. Or are we? The difference between 315 and 237 is 78, which is the last item of data in line 90. Is it more than coincidence?

The answer, of course, lies with the commas we added to the end of line 50. As we saw before, commas are used as separators between data items. When the Amstrad comes across a comma it expects some data immediately after it. And if there's none there it takes it to be the value 0.

This is what's happened when we added the comma to the end of line 50. Even though there's no data there, the Amstrad has assumed that there is and given *fromsomewhere* the value 0. And, of course, this effectively adds an item to the middle of the data list so there are now eight instead of the previous seven. The loop only cycles seven times so poor old 78 gets left out.

If you find that hard to understand add:

```
65 PRINT fromsomewhere
```

and you'll see the 0 tucked between 56 and 67. Once again the moral is take care with commas in DATA lines.

And that's all we're going to cover for this month. I hope that you'll have had a glimpse of how powerful and versatile the READ and DATA commands can be. And also how careful you have to be with data lists.

We'll be dealing with them again next time, but for the present I'll leave you with Program VII.

```
10 REM Program VII
20 sumone=0:sumtwo=0
30 FOR loop=1 TO 5
40 READ first,second
50 sumone=sumone+first
60 sumtwo=sumtwo+second
70 NEXT loop
80 PRINT sumone,sumtwo
90 DATA 1,2,3,4,5
100 DATA 6,7,8,9,10
```

Program VII

Here the READ is followed by two variables, *first* and *second*. How does it work? Does *first* take its values from the data in line 90 while *second* uses line 100? Or is it done another way? We'll find out next time.

Maxam

The ROM solution to creating assembly language programs

ARNOR'S Maxam is a 16k ROM containing an assembler, monitor and text editor. It has been designed to enable Z80 machine code programs to be written, run and debugged, with the emphasis on ease of use.

This version of Maxam has all the features of the tape version which was briefly reviewed in the April issue of *Computing with the Amstrad*, plus many more. It's now even better.

It is available either as a bare ROM which can be plugged into a ROM expansion board, or on its own small board which plugs into the expansion port. The latter is fully compatible with the disc drive, and the disc interface can be plugged into the rear of the board.

Maxam reserves 256 bytes of memory for itself on power up. If this interferes with software it can be disabled with ¦MAXOFF. The lost memory is regained and the ROM no longer responds to any commands.

The ROM is classed as a background ROM. This enables Maxam to be entered and left without disturbing the foreground program – usually Basic.

¦Maxam or ¦M is used to enter the ROM. An optional parameter can be added to specify the screen mode. There are two menus, the first has 16 options and the second 17.

Any ROM can be enabled so that it may be examined, with the exception of Maxam. It won't allow you to look at itself.

Any area of the memory map, ROM or RAM can be disassembled and listed to either the screen or printer. RAM can be displayed in both hex and Ascii and edited by moving the cursor to the desired byte and typing over whatever is there.

The disassembler disassembles all instructions correctly. This shouldn't need stressing, but it is important as the Amstrad uses some RST instructions to pass parameters. These parameters are listed after the RST.

The memory can be searched for a hexadecimal or Ascii string and all occurrences are listed to either screen or printer. Wildcards are accepted in place of a byte or character.

A block of memory can be initialised to any value, moved or compared. It's possible to move from ROM to RAM and compare ROM and RAM. When comparing two blocks, all differences are listed showing their different values.

If a block of memory contains machine code, instead of simply moving it Maxam can be instructed to attempt to relocate it as well. It isn't perfect – that would be next to impossible – but it does take a lot of the hard work out of it.

The text editor is used for entering assembler text. This is better than some word processors I've seen. It is so easy to use and makes such a

By ROLAND WADDILOVE

fantastic difference to writing machine code it makes other systems look primitive.

All the main word processing functions are available with the exception of justification, which is undesirable in an assembler.

The cursor keys and delete keys operate a character at a time. The cursor can be moved up, down left or right throughout the text and characters can be deleted forwards or backwards.

Holding down one of the Shift keys makes all commands operate on lines of text – which can contain multiple statements. Shift+cursor up moves the whole text up one line for instance. Shift+Delete deletes a whole line.

Ctrl enables the commands to operate on blocks. Defining a block is simply a matter of moving to the start of the text and pressing Shift+Copy, moving to the end of the text and pressing Shift+Copy again.

Ctrl+Delete will delete a whole block, Ctrl+Clr will move a block and Ctrl+Copy will copy a block of text.

The whole text can be loaded or saved or a block of text can be loaded or saved. Text can be loaded to the current cursor position in the text. This enables assembler files to be merged.

The editor has some powerful search and replace commands. Any string of text can be searched for and either a global or selective replacement is possible.

With a selective replacement the editor waits for confirmation before the text is replaced with the new string.

The text can be printed out as it appears in the editor or a listing can be made when the text is assembled. Any block of text can be printed and the listing can be switched on and off within the text when it is assembled.

The text editor has a couple of amazing commands which enable Basic listings to be manipulated.

If a Basic program is saved in Ascii format it can be loaded into the text editor. Then, using a single command, all the line numbers can be stripped away.

Whole chunks of Basic listing can be copied, deleted, loaded, saved, moved and inserted with the text editor's powerful commands.

When you've finished, line numbers can be inserted throughout the whole program with a single com-

mand and the listing saved. It can now be loaded and run as a normal file.

Basic and assembler can be mixed quite easily. ¦ASSEMBLE in a Basic program will enter the assembler and any subsequent Basic lines starting with a single quote are treated as text for the assembler. Control is passed back to Basic when the text has been assembled.

This is a very powerful feature and is ideal for short machine code routines.

The assembler itself is excellent and has many advanced features not available in some other packages. The code can be assembled with or without the object code being placed in the memory. The code can be assembled at one location but as if it were at another.

BRK is an additional instruction which provides a breakpoint. Maxam is entered after a BRK is met and the contents of the registers are displayed. Pressing a key will allow the program to continue.

'Recommended for all serious machine code programmers'

Conditional assembly is possible using IF-ELSE-ENDIF. The logical operators AND, OR and XOR can be included as well. There is also a special form of the IF statement, IF1 is true on pass one of the assembler and IF2 is true on pass two.

Using READ and WRITE an assembler file on disc can be assembled and the object code placed in a new file back on the disc without being placed in the memory. This allows very large files or several smaller files to be assembled.

In addition to the assembler, monitor and editor, the Maxam is able to list the ROMs present and their commands. Any number of ROMs can be disabled using ROMOFF,<list of ROMs>.

The screen memory can be moved to &4000 freeing &C000 to &FFFF. This enables programs residing in this area, such as ROMs and sideways RAM to be loaded and edited.

Maxam is a superb ROM which is recommended for all serious machine code programmers. It contains a powerful assembler and an excellent editor which is a joy to use.

An assembler in ROM is an absolute necessity for creating longer, more-complex programs. Arnor's is one of the best and worth every penny you'll pay for it.

DYNAMITE DAN

94% Crash Smash

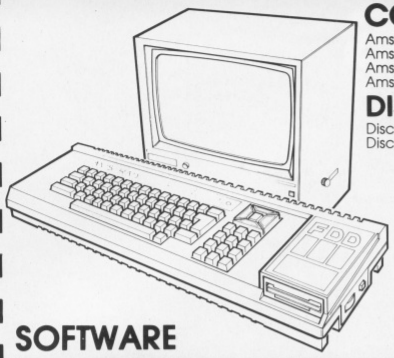
Amstrad/C64	£7.95
48K Spectrum/Spectrum+	£6.95

Popular Computing Weekly Pick of the Week

MIRRORSOFT

Maxwell House, Worship Street, London EC2A 2EN. Tel 01-377 4644
Trade orders to: Mirrorsoft Purnell Book Centre, Paulton, Bristol BS18 5LQ.

TRANSFORM LTD. Amstrad

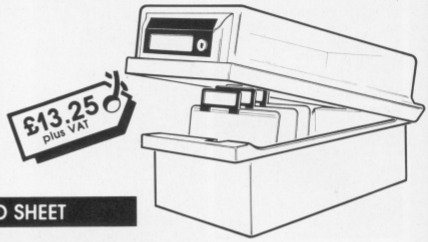


COMPUTER AND MONITOR

Amstrad 464 64k complete with green monitor and cassette £199.00p
Amstrad 464 64k complete with colour monitor and cassette £299.00p
Amstrad 6128 128k complete with green monitor and disc £299.00p
Amstrad 6128 128k complete with colour monitor and disc £399.00p

DISC DRIVE

Disc drive complete with controller £199.00p
Disc drive without controller £159.00p



- 3" Disc storage Box
- Holds 20 3" discs
- Lockable
- Brown with tinted removable lid

SOFTWARE

WORDPROCESSING

TASWORD D
Wordprocessing program complete with full screen prompts and tutor. This program will allow you to produce letters forms etc fully compatible with all printers. Features include page numbering, search and find, search and replace, block copy, block move and mail merge. The tutor will give full training on the use of the above functions.
Supplied on disc £24.95p
Cassette version without mail merge £19.95p

TASPRINT
The Tasword style writer allows you to print 5 extra fonts with dot matrix printers including COMPACTA bold and heavy for emphasis, DATA-RUN a futuristic script LECTRA LIGHT clean and pleasing to read Medlan a serious business like script, Palace Script a distinctive flowing font.
Supplied on cassette £9.90p
Supplied on disc £12.90p

SCREEN COPIES

TASCOPY
This program will allow you to copy your screen with shaded printing, including poster size screen copies using 2 or 4 sheets.
Supplied on disc £12.90p
Supplied on cassette £9.90p

DATABASES

MASTERFILE
This database has set the standard for micro computers. With 100% machne code it will allow 240 characters per field 40 fields record 34000 characters per file. This will enable you to keep a record of 500 full names and addresses. Features include full search facilities, sort in ascending or descending order, column totals, 1 or 22 records per screen.
Supplied on cassette £21.25p
Supplied on disc £25.45p

MASTERFILE EXTENSION
For those who want to stretch Masterfile, this packed import, export files to Tasword, merge files, data handling and many other features allowing calculations between fields etc.
Supplied on cassette £9.95p
Supplied on disc £13.95p

SPREAD SHEET

MASTERCALC
Spread sheet program from the same software house as Masterfile. Features include 3000 cells 8 byte floating precision 40-80 column display, direct totals and sub totals 3 bar histograms, cursor direction keys, add or delete columns-rows high resolution screen copies on Epson compatible printers.
Supplied on cassette £21.25p
Supplied on disc £25.45p

MASTERCALC EXTENSION
Allows you to transfer data from MasterCalc to Tasword and user access to MasterCalc data.
Cassette £9.95p
Disc £13.95p

PRINTERS

DOT MATRIX
Brother M1009 ..... £163.00p
Epson LX80 ..... £199.00p
Taxan Kaga KP810 ..... £250.00p

DAISY WHEEL PRINTERS
Smith Corona TP1 ..... £181.00p
Brother HR15 ..... £349.00p

Please add VAT to price of printers and £5.75p delivery.

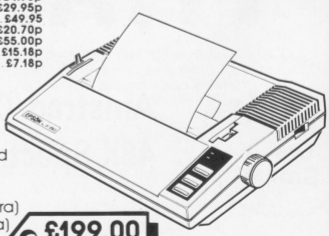
ACCESSORIES

ACCESSORIES
Centronics Lead £9.90p
3" discs £4.95p
Modulator £29.95p
RS 232 interface £49.95
Epson LX 80 tractor feed £20.70p
Epson sheet feed £55.00p
M1009 tractor feed £15.18p
M1009 roll holder £7.18p

DISPLAY LANGUAGE

DLAN
Advertise your products, services with this program by Campbell Systems. The print facility allows you to produce leaflets, labels, menus using 15 different fonts.

Cassette £9.95p
Disc £13.95p



EPSON LX80

- Near Letter Quality
- 100 characters per second
- 1k buffer
- Friction feed
- Tractor feed (optional extra)
- Sheet feed (optional extra)

£199.00 plus VAT

● Mail Order Only

Access

TRANSFORM LTD (Dept.CA) 01-658 6350
24, West Oak, Beckenham, Kent BR3 2EZ

VISA

A ROM card for your Amstrad

THE Superpower ROM card from Micro Power allows seven additional ROMs to be added to the Amstrad CPC464. It can also be used on the CPC664 but requires a bus expander to connect correctly.

If you're not familiar with the advantages of ROMs then here's some information you may find interesting.

ROM-based software only requires a small amount of RAM for workspace. Thus, much more memory is available for data than with programs on disc or tape. This is because the program doesn't reside in the main RAM block but in a ROM just like Basic.

So if you find the Amstrad's memory rather limiting, ROM-based software may provide a solution.

At present there aren't many ROMs available for the Amstrad. But if the BBC Micro is anything to go by there will be about a hundred ROMs available for the Amstrad this time next year.

Since the Amstrad can support well over 200 ROMs it should be interesting to see if anyone produces a board which can cope with the maximum capacity.

One thing worth bearing in mind is that you'll have to be prepared to pay between £25 and £50 for each ROM . . .

Fitting the Superpower ROM-card is a simple task as it connects to the expansion port at the rear of the Amstrad – no awkward soldering required.

If a disc interface is present it is slotted into the back of the ROM box in piggy-back fashion.

The ROM card is mounted in a grey plastic box which matches the Amstrad's case very well. The inside of the box can be examined simply by separating the two halves of the casing.

This reveals a neatly-laid-out circuit board containing four logic ICs, seven 28-pin sockets, seven address links and an edge connector.

Each ROM, which can be either 8 or 16k, has a ROM select number which is determined by its socket position on the board and the setting of the ROM's link.

However, the operating system requires the ROMs to conform with several rules. These concern the socket numbers and position of foreground and background ROMs.

A foreground ROM, such as Basic, is capable of taking total control over the machine. A background ROM is designed to be used from a foreground ROM to provide extra facilities. For this reason it is usually known as a service ROM.

In fact, the only problems I had with the ROM card was sorting out the link settings to ensure the various ROMs conformed to the rules. The instructions weren't very clear on this subject – however the documentation I had was only provisional.

Once the ROMs have been installed and the links connected, the unit can be plugged on to the expansion port. When the Amstrad has been turned on, the ROMs will be active and ready for use.

As it stands, the Superpower ROM board is very easy to fit and will be perfectly adequate for most people when they want no more than seven ROMs – and there aren't that many written yet.

What I'd like to see is a larger capacity board, with RAM. With such a board, ROM-based software can be loaded from disc or tape into a special section of RAM and will operate exactly as if it's a ROM.

If this happens, the price of ROM software could be almost halved because the expensive ROMs, and EPROM programming time, are no longer needed.

Still this is all pie in the sky. If you're only going to use up to seven ROMs – and who can afford more? – the Superpower ROM-card is perfect.

Kevin Edwards

...and a useful disc utility to pop in it

NOW that ROM expansion boards have started appearing for the Amstrad, several companies are releasing software on ROM.

One of the first few ROMs on the market is Disc Power by Micro Power, a disc utility designed for the CPC664 and disc-based CPC464s.

```
1  Directory
2  Read Disc Sector
3  Read Disc File
4  Read ROM
5  Edit Memory
6  Write Sector to Disc
7  Disassemble/Tabulate
8  Search Memory
9  Utilities
0  Screen (40/12)
```

Figure I: Main menu

The ROM is supplied in a video cassette-type box with an 18-page manual and a shorting link for the Superpower ROM card.

Once installed, Disc Power can be called up with the command |DP, after which a menu with ten options is displayed. These are shown in Figure I.

Each option is selected simply by pressing the key corresponding to the reference character at the side of the option.

The first menu option, specifically designed to help recover deleted files from disc, loads the directory of the disc into a 1k buffer at location &A000 then enters the Edit mode – option 5.

Option 2 permits any sector from any track to be edited in the same way as for the directory. Disc Power also uses the buffer for several other utilities, as we'll see.

Once the desired sector has been successfully edited, it can be saved back to disc using option 6 – Write Sector to Disc.

The editing screen, which

nearly all the commands are based on, consists of a hex and Ascii dump of a selected area of memory, the start of which can be changed by either scrolling through memory using the cursor keys or by entering a totally new base address. For most menu options this will be &A000 – Disc Power's own 1k buffer.

Editing is a simple task and involves positioning the cursor, using the arrow keys, at the location to be edited, and entering the new value. This can be done in either Ascii or hexadecimal.

The altered bytes are not written to memory until the Enter key is pressed on the same line.

If during editing you require a dump of the screen you can do so simply by pressing the Tab key.

Menu option 4 allows a selected ROM to be examined. Once a ROM has been chosen from a given list, the first 1k of it is copied into the buffer at &A000.

The editing mode is now called up and the ROM can be examined. The actual address of the ROM, as opposed to the buffer address, is shown on the top line of the screen. The rest of the ROM can be read into memory by pressing the appropriate cursor arrows in conjunction with the Ctrl key.

Option 3 loads 1k from the start of any file on disc into the buffer and enters the editor. The start track and sector of the file are calculated by Disc Power and displayed at the top of the screen.

The disassemble/tabulate option allows any section of memory to be displayed in the selected format on the screen or printer. This permits programs in ROM, RAM or on disc to be examined – assuming they've been copied into the buffer beforehand.

Pressing a key between 1 and 9 will display lines in that multiple of six at a time. In addition to this continuous output can be produced.

The disassembler mode can also be made to follow all jumps. This means that instructions such as JP, JR and CALL can be followed. This is extremely useful for debugging simple routines.

More-complex programs would have to be traced using a machine code monitor of some kind. Even so, the disassembler trace is very useful.

The search memory option allows an Ascii string or sequence of hex bytes to be located within a specific area of memory. The search string can contain a wildcard character – ?.

Thus a string search for J?K would find any string of three letters which starts with J and ends with K – such as JAK and JUK.

The addresses of all occurrences are then printed ready for interrogation.

Menu option 9 offers various utilities mainly for use with discs. These include a formatter, backup routine, simple calculator and a very clever disc/file map – the formatter and backup options don't require a CP/M utility disc.

The disc/file map produces a pictorial diagram of all 40 tracks of the disc and highlights specific areas. For example, a disc file can be selected and a diagram will be produced highlighting the tracks and sectors containing the file.

It can also show areas containing files which have been erased.

These can then be loaded from disc into memory and examined using the disc editor.

Provision has also been made to call routines in background ROMs by use of the bar commands. Foreground ROMs can also be entered but all data in memory will be lost.

The last two utilities allow files to be transferred between disc and memory.

The final option in the main menu permits the screen format to be altered. This allows a 40 or 80 column screen with either 12 or 24 lines.

In addition to the screen format the screen inks can be altered – specially designed for those of us who like the Amstrad's pretty colour palette.

The only real moan I have, though, is that the 18-page manual is poor and lacks informative examples – nice ROM, shame about the manual.

Otherwise, Disc Power is well presented and contains some useful utilities. It would have been nice if they had included a disc string search and an option to transfer programs from tape to disc and vice versa, but you can't have everything.

All in all, Disc Power is well worth considering.

Kevin Edwards

The disc/file map highlights specific areas

'Nice ROM, shame about the manual'

AT LAST!

The Amstrad mouse has arrived!

HERE at *Computing with the Amstrad* we're used to seeing innovative pieces of new hardware and software for the Amstrad – with so powerful and versatile a machine it's only to be expected.

However, occasionally a product reaches us that is so outstanding that the office is in a turmoil for days as the battle rages to see who's going to review it.

Such was the case with the AMX Mouse – not a furry little rodent in this case but a superb device that threatens to do away with the keyboard as we know it.

Like most great ideas, the concept behind it is simple: you use the mouse to control the position of a cursor on the screen.

The mouse itself is an easily held creature that you move over the surface of a desk on a smooth-running sphere.

As it moves so does the cursor, in a much more natural and easy to use manner than with a joystick.

However, it's not just a high-powered joystick. It comes complete with superb software that exploits the mouse's potential to the full.

In its simplest mode, the program enables the mouse to take over the normal cursor keys' functions.

It also checks for the three user buttons on the back of the mouse – which can be defined as any of the keyboard's own keys.

The mouse really comes into its own when combined with icons, however. These are mini-pictures which allow you to talk to the micro in picture language.

Instead of picking from a menu by choosing 1, 2 or 3, you are simply given an icon representing each selection and use your mouse to choose it. You just move the cursor over your choice with the mouse and press one of its keys.

The software already has 32 ready built icons, plus routines to allow you to incorporate them easily into your Basic and assembly language programs.

The package also contains an icon designer and pattern creator with which you can create your own icons, simply and quickly.

Perhaps the most striking demonstration of the package's power comes with the fascinating AMX Art software.

To describe this as a drawing program must be the ultimate understatement. As you can see from the illustrations on this page, the options offered by this powerful combination of mouse, software and the Amstrad's graphic capabilities are superb.

And the potential is tremendous. We just can't stop playing

with ours. We seem to come up with a new application for the mouse every five minutes – and so will you!

Frankly, we think that the AMX Mouse is the most exciting add-on that has been created for the Amstrad to date.

In fact, we've so much faith in it that we can offer the mouse, together with the AMX Art software, to *Computing with the Amstrad* readers at an amazingly low price.

If you want to REALLY make the most of your Amstrad, this is an opportunity you shouldn't miss.

Now turn to Page 101 for details of our exclusive value-for-money offer.

Gandalf

Shedding more light on mazes

LAST time we looked at mazes that are solved by movement, either dropping objects, making random moves or moving and making a map based on whether or not the room descriptions change. This month we'll look at three mazes that require you to think more carefully about what you're doing.

I'm not giving away too much, as the games themselves aren't out yet on the Amstrad. The ideas they involve should help you in your own explorations, though.

One particular adventure has two mazes, both of which require you to do things with some of the objects you have found, or should have found. If you haven't found them, you can't get very far.

In the first of the mazes you need to have found a lamp and lit it before you can even enter the maze.

If you try to enter without the lamp you don't live very long. Once in, however, you have to turn the lamp off, LOOK and then turn the lamp back on.

Why? Because the directions you need to get to the next location successfully and, therefore, through the maze, are written on the wall of each room. In phosphorescent paint.

So you have to turn the lamp off to see the direction you need to make next – and then you have to turn it back on to avoid being "eaten in the dark by a huge spider".

After getting through this maze you manage to collect several objects, one of which, when WAVED, emits "a cloud of dense white smoke".

You discover this because it is good practice in any fantasy adventure to RUB and WAVE everything.

Later, you find yourself in "a featureless black room". You find that you can't return the way you came, so you set off bravely to explore this new maze.

After wandering around aimlessly for a while you remember the rod and in desperation you WAVE it again.

This time you get "a cloud of dense green smoke", and after wandering through various locations, waving the rod as you go, you discover that the rod emits seven different colours:

Blue, green, red, orange, violet, indigo and yellow.

That looks familiar you think, so you arrange it thus:

Red, orange, yellow, green, indigo, blue and violet.

And what have you got? You've guessed it! The colours of the spectrum.

Now all you have to do is make a map based on these colours. So you return to your WAVEing and make a map of each of the colours in each of the locations.

When you've finished, you go back to the starting point and move in the order of the colours – that is, red, then orange, next yellow, followed by green and so on.

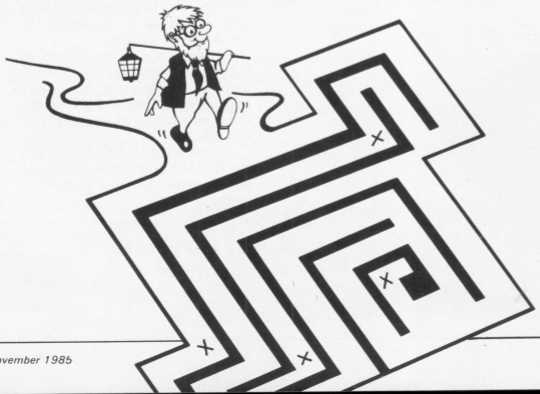
When you get to the maze exit you find it is a small cylindrical room whose only exit is back the way you came – in other words, back into the maze.

What's needed here is a password and since there is always a clue to any password somewhere, you think of the spectrum and eventually arrive at the password – "Rainbow" – and you're out!

Obviously, if you hadn't solved the maze you wouldn't have been able to get out.

The last kind of maze I'll look at is the type that gives you clues, if you can find them.

Often these clues have been given previously somewhere, perhaps as a reward for solving a puzzle, but they



can even be given in the maze itself.

In the extract below, you have almost completed the adventure and you are in the final room, which, wouldn't you know it, just happens to be a maze . . . The way out is given. Look closely and see if you can find it.

● You are in an octagonal room with exits in each wall. A plaque reads "WARNING: Keep moving or you'll regret it!"

Another sign reads: "The guardian waits forever here, eight ways to choose, one way is right, no time to lose, here ends your light, so walk where one can't see or hear".

The clues are all written on the second plaque. There are "eight ways to choose, that is, eight directions you can take, but it also means that eight moves will get you out.

Look at the last line. "So walk . . . *where one can't see or hear*".

The last part of the line is the important part. "WhErEoNE caN't SE-E or hEar".

Remember those eight moves? "W E E NE N SE E E"

Clever these programmers, aren't they?

Obviously, I can't cover every type of "thinking" maze you are likely to come across. The three I've covered above should give you an idea of what to look out for and the ones I've covered over the last two articles should help those of you who have written in.

And now to software . . .

Interceptor are very much in the news this month. To start with, they've released their fifth adventure, Warlord, which I hope to see shortly – hint.

I've just received their other four games, so I hope to be able to help those of you who have written in with problems soon.

M. Bannister has written in to say that Interceptor run their own adventure club with a newsletter. Send an SAE to them – NOT me – for more information if you want to join. I'm going to.

Kuma have released two budget-priced adventures: Shadow of the Bear and North Sea Bullion Adventure. I've just got my copies and will let you know if they are as good as Gems of Stradus.

I know that adventurers are supposed to be able to solve puzzles, but sometimes it can get a bit much. This month one reader has written in to say that he's stuck in one of Interceptor's adventures – but he doesn't say which one. And another reader is stuck in two of their adventures, but doesn't say where.

Please, when you write in tell me exactly where you are stuck and enclose an inventory and a map, or at least the solution to the problems you have encountered so far.

Clive Woodings wants to know what to do with the statue in the clearing in Fantasia Diamond. Can anybody help?

H. Woods has written asking for help with Technician Ted and Jet Set Willy. In TT he can't find the way into the diffusion furnace and in JSW he can't find a way into the sewer other than up the rope in the cold room.

Can anyone help him – and does anybody else think these two qualify as adventures?

Richard Hyams would like to know how to make the short rope into a long one in Emerald Isle. I thought there were two ropes – does anyone know differently?

Sean Lewis has got further than me and wants to know what the significance is of the office with the large desk at the top of the skyscraper. Can anyone help?

Lea – I think – Somerfield can't get into the inner castle in Dun Durach. I have to confess that I'm struggling with this game and would welcome any hints and tips.

D.C. Maggetti, L. Oliphant and many others – including D. Brown himself – have written in with the answer to D. Brown's problem with the chest in FAWE. Simply UNLOCK and OPEN CHEST.

Finally, please remember to enclose an SAE if you want a copy of our Sorcery hint sheet.

Agony column

G. Wilson, L. Mullen and Stephen Duffy keep trying to cross that river in Fantasia Diamond. *You can't yet! If you can't get over it, try to find a way of going under it.*

B. Macdonald is having problems with the cannibals and the vertical slab of rock in Jewels of Babylon. *The cannibals will give you a ticker-tape welcome if you give them a present, and is your rock the stone?*

If so, you need some leverage to get round this problem.

Darren Harris is having trouble mapping the jungle and thicket mazes in the same game. *There are only about 10 locations in each for you to map, so read my columns for advice on how to map mazes . . .*

Gary Clark wants to know how to keep warm and get past the mammoth on level 2 of Lords of Time. *If you strike a light you may find a means of solving both problems at once.*

Linda Wright is having trouble in Return to Eden. She wants to know how to get the skyhook, how to avoid getting killed by the helicopter gunships, and how to increase her strength so that she can carry more. *You can't, wait, have a beanfeast.*

Linda's also having problems avoiding getting drowned in the end-game in Colossal Adventure. *Kill the bad guys, get out of the way of the water, rescue the good guys, kill the spider and use its own means of transport.*

Tim Holland is also having a lot of problems with Colossal Adventure. *To avoid losing two treasures to the troll, give the eggs to get across, talk giantish and use the bear to get back. The spices are a valuable treasure.*

To complete the adventure – solve all the puzzles, find all the treasure and take it back to the building then solve the end-game.

SUPERCHARGE

SUPERPOWER SIDEWAYS ROM CARD (Ref B101)

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.

The SUPERPOWER Sideways Rom Card has the following features:

- Matching case, with easily detachable cover.
- Fits snugly to rear of computer.
- Bus extension for fitting of Disk Interface etc.
- Houses up to 7 Roms. (Foreground, Background & Extension).
- Any mix of 8K or 16K Eproms.
- No additional power supply necessary.

SUPERPOWER ROM BASED SOFTWARE

CURRENTLY AVAILABLE:

Programmer's Toolbox; Disk User's Utilities; Mailing List & Club Membership; Assembler, Disassembler & Machine-code Monitor.

COMING SOON:

Word Processor; Database.... SEPTEMBER
Spreadsheet; Graphics/Statistics....OCTOBER

Rom-based Software has the following important advantages:—

1. Written in machine-code, it is very fast in operation.
2. Programs are instantly available from the keyboard.
3. The program code does not use RAM, thus permitting much larger files to reside in memory, reducing the number of disk accesses and saving time when manipulating files.
4. The program itself cannot become corrupted.

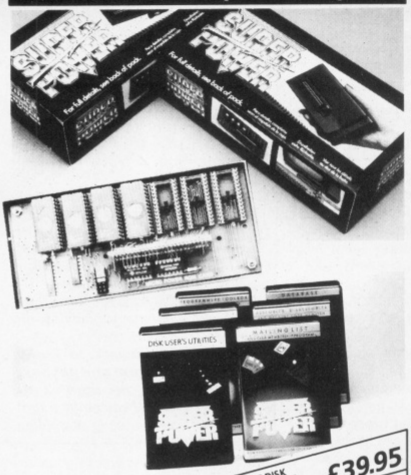
SUPERPOWER DISK USER'S UTILITIES (Ref B103)

Program allows detailed inspection and modification of information held on disk and is of particular use for recovering data from corrupted disks. Individual sectors can be read from and written to. All data can be output to the screen and/or printer. Program also contains a number of functions of use to assembly language programmers.

TABULATE DIRECTORY: Displays directory and enters EDIT mode. READ DISK SECTOR: Read sector and enter EDIT mode. LOAD DISK FILE: Load first sector into buffer and remainder to memory for fast access. Enter EDIT mode. LOAD UPPER ROM: Catalogues resident roms, prompts for rom selection and enters EDIT mode. EDIT MEMORY: Displays current buffer. Data displayed is Buffer Address, Hex and ASCII. Comprehensive editing facilities. 'COPY' key gives Intelligent Data Copy. 'TAB' key gives printer output. WRITE: Write sector to disk. DISASSEMBLE: Disassembles code from specified address, giving address, object code, menmonics and ASCII. Screen and/or Printer output. SEARCH MEMORY: Search sector or a complete file for ASCII string or series of Hex codes. UTILITIES: Includes Format, Hex/Decimal conversions, jump calculations etc.

SCREEN UTILITIES: Select from four display modes; choose background and foreground colours.

SUPERPOWER SIDEWAYS ROM CARD (Ref B101)



SUPERPOWER MAILING LIST AND CLUB MEMBERSHIP PROGRAM (Ref B102)	£39.95	SUPERPOWER DISK USER'S UTILITIES ROM (Ref B103)	£39.95
SUPERPOWER PROGRAMMER'S TOOLBOX (Ref B104)	£39.95	SUPERPOWER ASSEMBLER, DISASSEMBLER & MACHINE CODE MONITOR (Ref B105)	£39.95
SUPERPOWER WORD PROCESSOR (Ref B106)	£39.95	SUPERPOWER DATABASE (Ref B107)	

SUPERPOWER MAILING LIST AND CLUB MEMBERSHIP PROGRAM (Ref B102)

Program handles thousands of name and address records (label and non-label fields). Twenty classification indicators make possible sophisticated selective examination, counting and printing of records. Alphabetical order is dealt with on data entry, allowing user to select 'key' word. Works with single and double disks as well as cassette.

Main Commands ...
ENTER: Data entry(*) GET: Load new file. COUNT: Selective Count. FIND: Find 'Name' or any String(*). LIST: List current file on screen (*). PRINT: Print label data or whole records selectively. MERGE: Merge and Sort files. SAVE: Write a file to disk or cassette. RESET: Reset colours, label sizes, class definitions, string constants etc. (*Editing facilities available).

SUPERPOWER ASSEMBLER, DISASSEMBLER & MACHINE-CODE MONITOR (Ref B105)

This suite of routines represents the complete Development Package for the Amstrad Z80 programmer. The assembler has a sophisticated text editor, a comprehensive set of options, is very fast and incorporates special techniques enabling large source files to be handled in memory. The full-feature disassembler produces files which can be edited and then re-assembled. The Machine-code Monitor routines are extremely powerful, including the setting of conditional breakpoints (including loop counter option), single-step execution, alternative five column binary and mnemonic formats. Both sets of Z80 registers can be displayed. Other options include Intelligent Move, Modification of code to run at a new address and colour selection of border, paper and pen.

SUPERPOWER PROGRAMMER'S TOOLBOX for the CPC 464 (Ref B104)

ADDITIONAL BASIC

TURTLE: Logo-like turtle graphics. TUNE: Output music string. CIRCLE: Draw a circle. FILL: Fill an area surrounded by foreground colour. GRAPHICS PEN/GRAPHICS PAPER: Set graphics and background colours. COPYCHR: Read a screen charter. ECHO ON/ECHO OFF: Printer output on and off. PAGE ON/PAGEOFF: Paged screen output on and off. CLEAR INPUT:— clear input buffer. (+extra graphics commands).

PROGRAMMER'S AIDS

EDITOR: Gives 2 additional windows for program editing. FIND (& REPLACE): Find a string and optionally replace. XREF: List references to particular line numbers (GOTO's, GOSUB's etc). COMPACT: Remove REM statements. FRAME: Synchronise screen update with frame fly-back. PRINTER: Epson-compatible or Amstrad printer for dump routines. FDUMP: Fast two-tone screen dump, with user-definition of 'pen' and 'paper'. CDUMP: Shaded screen dump depicting up to 9 colours. PLOAD: Load program saved under 'P' option. INFO: Give details on specified disk file. MEDIT: Comprehensive HEX and ASCII memory editor. LTRON (LTROFF): Send TRACE output to printer. HELP: List commands, functions and their parameters.

ROM-BASED SOFTWARE FULFILS THE PROM

YOUR AMSTRAD.

SUPERPOWER WORD PROCESSOR (Ref B106)

This program incorporates the most useful facilities offered by the best Word Processors currently available on the BBC Micro and other up-market Home Computers.

Text Entry/Editing. User choice of 40 or 80 column mode. Document formatting achieved through use of embedded commands. Text can be formatted and justified on screen, if required. Find <and Replace>with 'wild-card' option. Word count, character count and page numbering reported.

Formatting. Multiple rulers are provided, allowing sophisticated layout of documents e.g. different combinations of TAB stops at different points in the text. All the normal controls are available e.g. page lengths & width, margins, indents, tab stops, justification, headers, footers etc., together with output of control characters to the printer.

Virtually UNLIMITED Document Length. The disk routines in the program handle multiple data files representing parts of a total document, with loading and saving carried out automatically.

Simple Calculator. A calculator window can be called to carry out simple arithmetic calculations. It is also possible to embed a calculation in the text, with the result being calculated and printed on output.

Data Exchange. As part of an integrated suite, the program will be able to handle ASCII files originating from the Spreadsheet, Database and Graphics/Statistics Packages.

Disk/External Commands. Direct Access available.

MANY MORE FEATURES. Ask your dealer for information.

SUPERPOWER DATABASE (Ref B107)

This menu-driven program has been designed to be the most comprehensive and flexible Database which can be achieved in 16K of machine code. Of special note are the PRODUCTS which have been implemented, enabling the more advanced user to write simple structured programs to manipulate the database information in order to meet his more specialized requirements. Databases need not be limited in size since the program contains routines for multiple file handling.

Field types — alphanumeric, integer, currency, floating point, logic, calculated and date.

MAIN FILE COMMANDS
— Create/Modify database structures, reports and procedures.
— Create ASCII file from a Database, enabling data to be passed to the Word Processor, Spreadsheet and Graphic/Statistics programs.
— Copy data to new Database.

RECORD EDITING
— Add, amend, delete, insert, view and duplicate.
— Search and Sort. Due to intelligent file structure and the indexing system adopted, both SEARCH and SORT are extremely fast.

REPORTS
Standard — selected fields may be output, together with totalling on those nominated. (Value fields only). Screen or printer output.

Labels — user-selected fields for printing. User-defined label size, number on sheet etc.

User-defined — user created report-format, combining entered text and designated fields. Editing and storing facilities.

SUPERPOWER ADVICE CENTRES

AVON
K & K Computers,
Weston Super Mare
0934 419324.

BEDFORDSHIRE
Chiltern Computers
Luton 0582 455684.

Hobbyte Limited
Luton 0582 457195.

Zedem Computers
Bedford 0234 213645.

DJ Computers
Leyton Buzzard
0525 383929.

BERKSHIRE
Gadgets
Windsor 07535 67211.

CHESHIRE
Northern Computers
Frodsham 0928 35110.

Pyramid Micros
Warrington 0925 35713.

CLEVELAND
Multicoloured Micro Shop
Redcar 0642 486643.

DEVON
Computerbase
Plymouth 0752 672128.

Computer Systems (Torbay)
Paignton 0803 524284.

Georges Booksellers
Exeter 0392 71072.

Georges Booksellers
Plymouth 0752 661011.

Computerware
Exeter 0392 210507.

Syntax Limited
Plymouth 0752 28705.

Seven Counties
Exeter 0392 211211.

DORSET
Seven Counties
Dorchester 0305 66022.

Lansdowne Computer Centre
Poole 0202 670901

Lansdowne Computer Centre
Bournemouth 0202 290165

EIRE
Bit by Bit
Bray 001 868 541.

ESSEX
Clarke Coyston Computers
Southend-on-Sea 0702 337161.

Estuary Personal Computers
Southend-on-Sea 0702 614131.

Computerama
Southend-on-Sea 0702 335443.

Colchester Computers
Colchester 0206 47242.

Mirage Microcomputers
Braintree 0376 48321.

Computer Centre
Hornchurch 04024 75613

GLOUCESTERSHIRE
Screen Scene
Cheltenham 0242 528979.

HAMPSHIRE
Electronequip
Fareham 0329 230671.

Alton Computers
Alton 0420 87993.

Timatic Systems
Fareham 0329 236727.

HERTFORDSHIRE
DJ Computers
Stevenage 0438 65501

DJ Computers
Welwyn Garden City
0707 328435.

Hobbyte Ltd.
St. Albans
0727 41396

HUMBERSIDE
Computing World
Beverley 0482 881831.

Ashby Computer Centre
Scunthorpe 0724 871756.

ISLE OF WIGHT
V M S Computer Centre
Newport 0983 525450.

KENT
The Village House of Computers
Bromley 01 460 7122.

Microway Computers
Rainham 0634 376702.

Marlowe Micros
Canterbury 0227 453531.

Adventurer's Guild
Gravesend 0474 334008.

Numbus Computing
West Wickham
01 777 3160.

MD Micros
Crayford 0322 524123.

LANCASHIRE
Computer World
Bolton 0204 494304.

I M O Computers
Burnley 0282 54299.

4MAT Computing Ltd.
Preston 0772 561952

LONDON
Adams World of Software
London N12 01 446 7170.

Colmore & Microtanic
Dulwich SE22
01 299 1419.

Vic Oddens
London SE1 01 403 1988.

Computers Inc.
London NW11
01 209 0401.

G & B Computers
London W1 01 580 3702.

Ramco Electronics
London NW7 3TG
01 906 3363.

LINCOLNSHIRE
M K D Computers
Lincoln 0522 25907.

MANCHESTER
Mighty Micro
Fallowfield 061 224 8117.

Celcom Systems
Manchester
061 834 2808.

MERSEYSIDE
MicroByte
Wallasey 051 6306933.

Hargreaves Computers
Liverpool 051 525 1782.

MIDDLESEX
J K L Computers
Uxbridge 0895 51815.

Screens Microcomputers
Northwood
09274 20664.

Trionic Limited
Harrow 01 861 0036.

NORFOLK
Abacus
Norwich 0603 614441.

Thetford Micros
Thetford 0842 61645.

NORTHERN IRELAND
Botanic Computer Centre
Belfast 0232 240798.

Fun-cational Computers
Lurgan 07622 28543.

NORTHUMBERLAND
Alnwick Computers
Alnwick 0665 604816.

NOTTINGHAM
G A Computers
Ruddington 0602 211225.

Computer Market
Nottingham
0602 586454.

S P Electronics
Nottingham 0602 640377

SCOTLAND
Micro Mart
Kirkcaldy 0592 206251.

SOMERSET
The Forum
Chard 04606 5009.

STAFFORDSHIRE
Leek Computer Centre
Leek 0538 387859.

Computer Cabin
Newcastle under Lyne
0782 636911.

Town Computer Centre
Stoke on Trent
0782 287540.

SURREY
Computerama
Burgh Heath
07373 54717.

Computersolve
Surbiton 01 390 5135.

Micro Workshop
Epsom 03727 21533.

SUSSEX
Gamer
Brighton 0273 698424.

Computermania
Chichester 0243 779263.

Computermania
Littlehampton
0903 716187.

TYNE AND WEAR
Sunderland Computer Centre
Sunderland
0783 655711/655736.

WALES
Gwynedd Computers
Caernarfen 0286 2610.

Cardigan Electronics
Cardigan 0239 614483.

Bucon Limited
Swansea 0792 467980.

Teleview
Neath 0639 53282.

Cwmbran Computer Centre
Cwmbran 06333 60996.

WARWICKSHIRE
O E M Computer Systems
Rugby 0788 70522.

Micro City
Nuneaton 0203 382049.

WILTSHIRE
Barbury Computers
Swindon 0793 611487.

Trowbridge Computer Shack
Trowbridge 02214 67299

YORKSHIRE
Micro Power
Leeds LS2 0532 458800.

Danum Computer Systems
Doncaster 0302 20355.

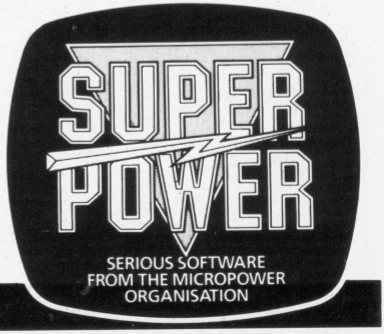
The Computer Room
Barnsley 0226 204000.

Just Micro
Sheffield 0742 752732.

DEALERS. Become a SuperPower Advice Centre and help us promote your expertise in home computing, with particular reference to the Amstrad market. Contact Eileen Garfield on 0532 434006 for details of our dealer support package.

HOW TO ORDER

Phone or write to: The SuperPower Project Office, Micro Power Ltd., Northwood House, North Street, Leeds LS7 2AA. TELEPHONE (0532) 434006.



MISE OF YOUR AMSTRAD.

pride utilities LTD.

Some programs are sold on the understanding that they must not be used to infringe copyright.

SYSTEM X — 32 new extended Basic commands.

Yes, at last you can upgrade your 464 or 664 with this massive range of commands whilst using approximately 3K of memory. Take some of the hard work out of your programming, use them in your programs and take pleasure in computing with your much enhanced Amstrad — even gives your computer a Circle and Fill command. **NEW COMMANDS:**

I FRAME	I RSCREEN	I LSCREEN	I USCREEN
I DSCREEN	I CIRCLE	I FILL	I INVIS
I VIS	I GETKEY	I CURON	I CUROFF
I R	I RESET	I FLUSH	I PROTEC
I DEPRO	I CAPON	I CAPOFF	I GPEN
I HELP	I GETCHAR	I PRON	I PROFF
I RPEEK	I DPEEK	I DPOKE	I INVERSE
I GOVER	I ZIP	I MOTOR	I SHIFT

Cassette £9.95 inc. Disc £13.94 inc. (464, 664 and 6128). State Machine Model

TRANSMAT

Easy Tape to Disc Transfer

Offers easy transfer of **tape based software** onto **disc.** For Amstrad DDI-1 disc drive system. Nothing available on the market can match TRANSMAT'S capabilities and versatility. Beware of poor copies. Offers: Auto or non auto modes, comprehensive disc header reader, adds relocator if necessary, easy erase or rename programs. ***"This is a very easy to use utility"*** *— Personal Computing Today, July 1985.*

Cassette £7.95 in. Disc £11.95 inc. (464, 664 and 6128)

TOMCAT

Cassette Headerless Backup

Achieve the ultimate in **tape to tape backup** with this no nonsense program which handles practically everything, even headerless, usually in one go. If you require to load and list software etc. then you require SYCLONE, the versatile tape to tape utility. If all you need are backups of your software which load in up to 4 times the norm, then you must have TOMCAT. There is probably nothing on the market that can anywhere match TOMCAT'S performance and easy operatfon. ***"Second best thing to disc drive".***

Cassette £6.95 inc. Disc £10.95 inc. (464, 664 and 6128)

AMSTRAD 464/664/6128

QUALITY UTILITIES FROM OUR FAST, RELIABLE AND FRIENDLY MAIL-ORDER SERVICE. ALL ENQUIRIES WITH S.A.E. ANSWERED. TRADE AND OVERSEAS ORDERS WELCOME.

RSX SYCLONE 2

Tape to Tape Utility

A RSX Tape to Tape utility which adds brand new commands, offering many facilities, for your CPC 464. Allows you to backup your valuable software to load in up to one quarter of the time. Choice of seven saving speeds, 1000 to 4000 baud. SYCLONE not required to load back high speed copies. Comprehensive header reader. Commands available from BASIC, load and list protected BASIC programs. List your WELCOME tape. Disc compatible. ***"A very useful tape to tape tool kit".***

Cassette £6.95 inc. Disc £10.95 inc. (464 only)

RSX PRINTER PAC-1

Printer Utility for DMP1 and EPSON Compatible Printers

A resident system extension which adds SIX new commands to your CPC 464/664. A two program pack which caters for the **AMSTRAD DMP1 printer** (AMDUMP) and **EPSON compatible printers** (EPDUMP). Screen dump in all modes. Two sizes with EPDUMP. Text dump in all modes. Specify background inks. Abbreviated codes to printer. Three new type styles for the DMP1. Can be used with AMSWORD. ***"A very useful utility, frequently used".***

Cassette £5.95 inc. Disc £9.95 inc. (464 664/6128 version disc only)

SCRIPTOR

Definable Character Fonts for the DMP1

A must for **DMP-1 printer** owners. Offers six definable character fonts. Allows your DMP-1 to perform — don't upgrade. Offers: true lower case descenders, joined writting, futuristic, bold and heavy, italics, character definer program, Amsword compatibility. Professionally finish your letters. ***"Tired of the standard type-style on the DMP-1 Printer? Then Scriptor is for you."*** *– PCT July, 1985*

Cassette £6.95 inc. Disc £10.95 inc. (464 and 664)

ROMS

Now available on ROM, **TRANSMAT** the much respected and professionally used Tape-to-Disc Transfer program. Also available **SYSTEM X** — both only **£19.95 inc.** Suitable for 464, 664, 6128 – (must not be used to infringe copyright).

SPECIAL OFFER

Buy two or more titles and receive, completely free, a real time Digital Alarm Clock program on cassette

PRIDE PROMISE

We are so confident in our professionally manufactured and packaged software that we dare ask you to show us a cheaper comparable commercial program that can match the high standards of any one of our products and we will refund the difference in price.

CUSTOMER ENQUIRIES

TEL: 0582 411686 between 9.00-10.00 am please.

PRIDE UTILITIES LTD.
(Dept. CWA)
7 Chalton Heights
Chalton, Luton,
Beds. LU4 9UF
England

ALL PRICES INCLUDE VAT AND P & P.
For Europe add £1.00 per title
For Australasia add £1.75 per title
For Rest of World add £1.50 per title

THIS IS NOT JUST A MODEM, BUT A COMPLETE SYSTEM, NOTHING ELSE TO BUY

'COMMUNICATOR'

Incorporating serial and parallel interfaces, to allow software control of all functions. Each feature controlled from Basic with the bar commands. Call from M/C or on entering bar modem driven for ease of use. Bell/CCITT standards 300/300, 600, 1200, 1200/75, 75/1200 full and half duplex. Auto dial and auto answer, contact bulletin boards. Prestel compatible, software built in on its own sideways ROM, unique panel display, it displays what the modem is doing, mode of operation, and digits when auto dialing, standard B.T. plug connector.

ONLY £163

Note. This modem is not B.T. approved.

ALSO AVAILABLE AT AMAZING PRICES!

SIDEWAYS ROM

The unit holds 4 ROMS. Each can be 2,4,8 or 16k in size incorporating a device to allow +slower+ ROMS to be used less than Amstrad suggested 200ns That means cheaper ROMS. Free utility ROM with every unit.

£66.95

PARALLEL PORT

Make that robot move. Control electrical appliance. Twin 8 bit ports. Operates direct from Basic 2 x 14 way speedblock connectors

£66.95

ALL UNITS ARE CASED AND HAVE THROUGH CONNECTORS

RS 232

Communicate with your modem. Talk to other computers. Use serial printers. Split baud rates. Standard 25 way 'D' connector.

SPECIAL OFFER £64.95

8 BIT PRINTER PORT

Make use of that 8 bit printer Allows character codes Above 127 (ie 0 to 255) Plugs in between centronics Port and printer cable

£17.95

Connecting Cable

A connecting cable is required for connecting the RS232, parallel port and the sideways ROM to the CPC664. All other units connect directly to the 464 and 664. A cable kit is available and will allow any add-ons to be connected to the 664 – £5.20. Software on disk– £3.50 extra

COMPUTER HARDWARE & SOFTWARE

KDS ELECTRONICS

15 HILL STREET, HUNSTANTON, NORFOLK PE36 5BS. TEL:(04853)2076

Please add VAT to all prices.

Useful business package

WHEN Amstrad released the CPC464 they intended that it should be used in some capacity as a business machine. To this end they have begun releasing a number of programs aimed at the small business. **Invostat,** from Dialog, is one.

It will produce a range of documents including invoices, statements, and credit notes. The accompanying eight page booklet is both concise and informative. Within 30 minutes of loading the program the user should be capable of producing the whole range of available documents.

The program is menu driven, providing the non-computing businessman with easy access to all facilities. When reviewing this type of program I always look at its crashability rating. This is always important when company records are at stake. I found Invostat impossible to crash even when stooping to my dirtiest tricks.

Each customer has his own client file in which all previous transactions are stored. Having accessed a particular file you can view any of the documents in detail. They are professionally presented in 80 column mode and can be sent to the printer if required.

As part of setting up the program you enter details of all of your products, the price of each item being used by the program in invoice calculations.

Though the entry of this data may seem a tedious task, the file can be used to produce an up-to-date price list at the press of a button. New products can be added at any time and the prices of old ones amended in seconds.

Invostat is an easy to use yet very professional product. Its facilities should prove useful to any business, no matter how small.

James Riddell

Leave the arcades for the links

WITH the golf season well into its stride, it seems an appropriate time to take a look at CRL's **Handicap Golf.**

The program offers a full 18-hole golf course, with a choice of playing nine or 18 holes.

One or two players can compete, although I found that two players make the game far more interesting.

The Mode 1 graphics screen is used, which unfortunately limits the selection of colours available, but nevertheless the graphics are excellent.

The layout of each hole may be previewed. This is necessary because many of the holes are too long to fit on the screen, and it takes several views to see the entire hole from tee to green.

The holes are well laid out, with the inevitable well-placed trees and bunkers, but the layout seems just a little bit repetitive as all the holes are played directly left to right.

After viewing the hole, it's time to select a suitable club ready to tee off.

A full set of clubs is available, and these are displayed together with the approximate driving distance possible from each one – which is useful if you're not a golf expert.

Wind direction is displayed on the screen, and this needs to be taken into consideration before selecting a club.

Having chosen, the direction of hit is then adjusted with the cursor keys before you finally drive off. After this the ball is seen rising into the air, hopefully in the general direction of the green.

The player graphics are quite novel.

Both player and caddie are displayed on the tee. After a neat swing, the ball moves off and the player is seen to throw his club back to the caddie.

Once the ball stops, both men walk off in the direction of the ball ready for the next shot.

With careful selection of club and direction, you should find yourself on the green, within putting distance of the hole.

A close-up of the green is shown, and now it's a matter of directing the putter and selecting the strength of hit.

My only criticism of the game is that I found it a little too easy.

In my first 18-hole round, I scored an amazing five under par – watch out, Sevvy . . .

With a little bit of putting practice, I should hope to improve on this score.

Overall, though, this is a good implementation of the game of golf.

The graphics are well thought out, and entertaining.

It's easy to use, and I really found the game quite relaxing – a welcome change from fast-moving arcade-type games.

Geoff Turner

Sound makes the difference

LOAD in **Jammin,** from Taskset, and you end up with a veritable disco – 20 loud tunes, each accompanied by an infectious lively drum beat, plus plenty of bright, flashing lights on the screen.

You've also got quite a good game – original, challenging and above all refreshingly amusing.

But before you discover the delights of this problem you have got to wait for around nine long minutes while block after block feeds slowly into the computer.

During this agonising delay, there isn't really much to do. Luckily, the wait is worth it.

You play the part of Rankin Rodney, saddled with a daft name and a difficult task.

Rodney must make his way through passages of the instrument dumping ground, searching for instruments to create the perfect big band sound.

Conduct yourself carefully, you are warned, as bum notes and distortion will attempt to block your way.

There are 20 screens in all, each with its own tunes and problems to overcome.

On each screen there are four instruments for you to collect and bring back to the centre.

The instruments are guarded by a stationary monster which kills on contact.

There are also roving semi-quavers that will try to steal any instrument you are carrying and return it to its guardian.

You move around using a system of moving boxes, hopping from one to the other depending on where you want to go.

You have four lives with which to complete your task.

A bonus ticks down from 4,000 and if you lose a life it resets and you have to start the screen again.

Although the screens are colourful the graphics could have been of far higher quality.

Rankin Rodney and the four guardians are crudely drawn in one colour. Also, responses are sometimes annoyingly slow.

But one thing that cannot be faulted is the sound. The superb tunes add a great deal to the game and each varies during play so there should be no need to reach for the volume control in despair.

Without its infectious tunes, Jammin would not be value for money. With them, it is well worth a look.

Brian Finnerty

And now

IN a way I wish I hadn't volunteered to review **Psychedelia,** from Llamasoft – the latest offering from the fertile mind of Jeff Minter.

Oh, there's nothing wrong with the product itself – quite the opposite, it's superb.

The problem is that it's just impossible to describe . . .

When shown in the office and to friends, it provoked responses such as "Wow . . . Amazing . . . It's like a firework display. How does he do that? The ultimate Etch-a-Sketch!

Making a mint isn't all that easy...

START to load **Millionaire,** from Incentive, and make yourself a brew . . . As you are just squeezing the tea bag, you will hear "Who wants to be a millionaire?"

Anwer the phone ringing from your Amstrad as this signifies the start of the game.

The plot is easy enough.

With £500 and a computer program, you start up a business and through months of endeavour, wheeling and dealing, you attempt to become a millionaire.

Well that's how the theory goes – now on with the game.

The first question you will be asked after selecting the software field that your company will be trading in is: "What makes a good program?"

For this you are allocated 20 points to share out among four statements, marking each with up to eight points.

For each month of trading you will be shown a financial statement and a bar graph of your performance, followed by any news.

At your turn you must select from a list of options for that month, for example Programming, Loans or See Honest Harry the get-rich-quick guy.

Monthly advertising comes as next choice, followed by five once-only items, such as a TV advert costing £5,000.

All that remains now is to choose how many tapes and the cost of duplication and wait for next month.

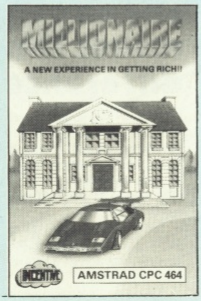
You start the game living in a humble terrace and move up through a variety of dwellings to an office block and – well, that would be telling, as this is the monthly reward for your achievements.

Care has to be taken in the early stages of the game to ensure positive assets – if you overspend you become bankrupt and this ends the game.

This is an interesting game, well presented and it will entertain young and old alike.

To reach the dizzy heights takes a long time and an option to save the game would have been a nice touch.

Lynne Savidge



It's levels and ladders time again

HARD Hat Mack, from Ariolasoft, is a level-and-ladders game.

The action takes place on a building site and there are three completely different screens.

To succeed on the first you must climb to the top by filling in gaps in the platforms using steel girders.

While doing so you must avoid rivets which are being thrown down from above by an invisible assailant.

Once you have plugged all the gaps, you have to rivet each girder in place with a special gun that zips along the levels looking rather like a spinning top.

To assist your progress there are chains at one end of each platform which you can climb up.

In addition, there is a trampoline that you can use to bounce up to the next floor and there is also a lift at your disposal which takes you up three floors.

On the second screen you have to race around collecting lunch-boxes.

In order to access the different platforms you have to hop on to a girder which is hoisted up and down by a winch.

Darting around the first two screens there is a meanie called Vandal whose sole purpose is to make life unpleasant for you. One touch from him and you're dead.

Two meanies appear on the third screen – Vandal and Osha, a government inspector.

Here you must collect boxes and put them into a chute. In the centre there are lifts that Hard Hat Mack can ride to get to the different floors.

The addition of a joystick option was the only reason I was able to play the game, because the indicated keys wouldn't work.

Apparently an error was made when the cassette was

w for something entirely different...

Far out, man . . ." and much more.

Psychedelia is subtitled A Light Synthesizer, which just about sums it up.

The program is a tool which enables you to create fantastic swirling patterns of colour that almost seem to be alive – moving round the screen and growing all the time.

The author intends it to be used to create graphic displays to accompany music. There are three ways of doing this.

The first is to sit down with a joystick, preferably in a darkened room, and create "live", in time with the music.

Alternatively you can practise your piece, record it – and it'll remember up to half an hour's-worth of display – and replay it while you're listening to the music. The third method is to put it in automatic mode and just watch.

The program uses the keyboard to set the various options and parameters and the joystick to actually create the patterns. The speed, colours, symmetry, pulse, lightform, buffer length, ooze, sloth mode, strobo zap rate, explosion mode and many more functions and parameters can be set by pressing the appropriate keys. Such flexibility is appreciated.

Then sit back, hold down the fire button on your joystick and just wave it about . . .

There's no need to worry about not being creative or not being able to draw – it's impossible to do badly. Anyone can use it to produce his own masterpieces.

You'll either love this or be totally bored.

I can sit for hours just staring at its almost-hypnotic display.

It's not a game, nor a utility, and there's nothing to win. There aren't any puzzles and there isn't anything to beat. So what is it?

It's simply the most fascinating, unusual and entertaining piece of software I've seen!

Roland Waddilove

released with Spectrum instructions, so regrettably I was unable to test the keys.

To give Ariolasoft credit, they rectified this slip-up immediately they became aware of it.

The Mode 1 graphics were fine and good use was made of sound.

The game could perhaps have been improved by the addition of some different screens, but even in its current state I found it fun and extremely addictive.

I am certain that a game played via the keyboard would be equally enjoyable.

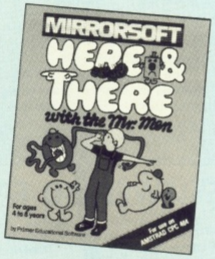
David Andrews

Getting left in the right place

LEARNING to tell the difference between left and right can be a difficult task for the average four-to-eight-year-old and the four programs in **Here and there with the Mr Men,** from Mirrorsoft, should make the task easier.

The cassette is accompanied by a small Mr Men book containing a short story to go with each program on the tape.

A menu allows you to select any of the four games,



although having made your choice there's no way back to the menu except loading the tape again.

The first game finds Mr Tickle in pieces and your task is to put him together again by moving a hole in the sides of a box until it is aligned with the piece in question.

If the gap's in the right place, the pieces shoot through the hole to restore Mr Tickle to his normal state.

Once he's back in one piece again, he's off to tickle Mr Grumpy, who locks himself in a room with doors which are too small for Mr Tickle to get through.

Using simple instructions, you must tell Mr Tickle which direction to move his arms so that he can reach Mr Grumpy.

The excellent use of graphics and sound in this game is guaranteed to keep the kids amused for hours and, should you tire of the noise, the sound can be turned off altogether.

In the third program, poor old Mr Lazy is feeling very hungry and your task is to guide a worm along the branches of a tree to throw an apple into his open mouth.

The graphics are, yet again, sure to capture the imagination and keep the children busy.

The final program is a version of the standard computer game Fox and Hounds, although the characters have been changed.

This package contains some of the best educational software for very young children I've seen for Amstrad computers and can be thoroughly recommended.

Steve Lucas

... and maths for the minors

THE House that Jack Built, by Knightsoft, has a picture on the cassette inlay of Jack, a thick-looking builder counting on his fingers.

This is the only clue, with the title, as to the contents of the cassette – an educational package aimed at 7 to 9 year-olds.

The object of the game is to answer simple mathematical questions and, if you answer correctly, watch Jack build his house.

After selecting the choice of multiplication or division, sound, music, clock and a degree of difficulty, a picture appears of Jack standing waving at you.

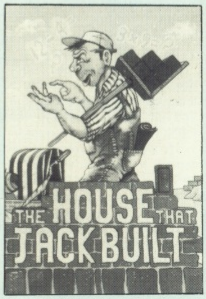
The clock, if selected, starts, and 20 seconds later you are asked the first question – for example, 8 x 3 = .

Answer the question correctly and watch him move to the other side of the screen and build a row of blocks – while the tune "Hi-ho, hi-ho, it's off to work we go" plays, if the music option is chosen.

When Jack has laid his blocks, he returns to his standing position and the second question is asked.

Answer all 15 questions and the house is completed, Jack waves goodbye and walks into his garage.

The screen clears and



gradings as to how fast you achieved the task are displayed.

As an example, for Excellent you need a time of just over eight minutes, which is 30 seconds answering the questions and nearly eight minutes watching Jack.

This program is sold under the educational banner and its aim is the practice of simple sums.

The child using this program, however, is given no help if poor at sums. In fact quite the reverse. Answer a question wrongly and three rows of bricks are removed which took one and a half minutes to build.

But beware, if you answer the same question wrongly three times a message appears: "Correct answer is 8. Now enter it and remember it".

The kind of child this slow approach would suit would not benefit from being told off.

The graphics are reasonable, but the house is always the same. The program is much too slow and educationally it's only suited to those who don't need it.

Glynne Davies

Graphics made easier

DRAUGHTSMAN, from Computer Graphics, has the makings of an extremely good graphics aid.

It comes complete with a 15-page manual, and a keyboard strip showing the action of the re-defined keys.

The standard graphics toolbox utilities are available – line, circle, ellipse, polygon, and arc drawing, colour and infilling, in all three modes.

In addition there is an option to design shapes on various sized grids. This works like a character or sprite designer, with each shape designated a number and able to be re-called when building up a drawing.

The shape to be drawn at the cursor can be moved into place before being finally set into position. This facility can also be used to good effect when placing text on the screen.

There are guides for drawing in perspective and a hatching command, vital for technical drawing where sectioning is commonplace. In fact professional packages are often judged on the speed of their auto-sectioning.

Regrettably, however, the program does have its failings.

The fill command, for example, is painfully slow.

The hatching is user-controlled but very difficult to use when shading irregular shapes.

A printer dump command is also included but it only refers to your own existing dump. A package of this potential should surely include one of its own.

On the good side, the program is easy to use, the manual easy to follow, and the presentation is professional.

Glynne Davies

Tutor with a twist

THERE are some programs which are very easy to review, because their quality is immediately apparent.

Others are equally easy to identify as poor fodder, and again their deficiencies are obvious.

However, **The Key Factor,** from Amsoft, is a real problem, because I have found it to be frustratingly addictive and good fun to play – yet I'm not at all sure what makes it such a good program . . .

The basic idea is extremely easy, and not terribly original, involving shooting down sprites which threaten to engulf the Earth.

These invaders fall in one of eight columns, at the base of which is a letter or other keyboard symbol.

The correct key must be pressed in order to shoot the invader in a rather impressive explosion.

This combination of arcade game and keyboard tutor doesn't sound particularly exciting, I can hear you say. But the symbols labelling the columns have a discouraging habit of changing, so your wits have to be sharp throughout.

The rate with which the sprites drop also increases with the level selected, and as various screens are successfully negotiated.

The programming of the invaders is very impressive, and the movement extremely smooth.

Keyboard response is quick and precise, and it is possible – indeed, vital – to have several missiles launched at once.

When one of the letter bases is sufficiently worn down, the fate of the Earth is inevitable.

As I said – rather addictive, but I am still not sure quite why.

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
Artist & Sprite Designer	C.R.L.
Artwork	Kuma
Astro Attack	Amsoft
Amstrad Tutorial	Amsoft
Atomsmasher	Romik
Beach Head	U.S. Gold
Blagger	Alligata
Brawn Free	Nemesis
Bridge Player	C.P.
Centre Court	Epicsoft
Chopper Squad	Interceptor
Code Machine	Picturesque
Colossal Adventure	Level 9
Confuzion	Incentive
Country Cottages	Sterling
Crystal Theft	Wiccasoft
Cubit	Mr Micro
Deep Thought Chess	C.P.
Defend or Die	Alligata
Detective	Argus Press
DFM Database + Labels	Dialog
Digger Barnes	Cable
Dragon's Gold	Romik
Draughtsman	Computer Graphics
Dungeon Adventure	Level 9
Ex*bert	Microbyte
Fighter Pilot	Digital Integration
Flexi-friend	Camel
Flight Path 737	Anirog
Flight Simulation	Myrrdin
Font 464	Hisoft
Football Manager	Addictive
Frank N. Stein	P.S.S.
Fruit Machine	Amsoft
Fruity Frank	Kuma
Forest at World's End	Interceptor
Galaxia	Kuma
Gate Crasher	Exopal
Gems of Stradus	Kuma
Ghouls	Micro Power
Handicap Golf	CRL
Happy Letters	B.E.S.
Hard Hat Mack	Ariolasoft
Harrier Attack	Durrell
Here and There with the Mr Men	Mirrorsoft
Happy Writing	B.E.S.
Holdfast	Kuma
Home Accounts Manager	Dialog
Home Budget	Kuma
House that Jack Built	Knightsoft
Hunchback	Ocean
Hunter Killer	Protek
Invostat	Dialog
Jammin	Taskset
Jet Boot Jack	English
Jet Set Willy	Software Projects
Key Factor	Amsoft
Killer Gorrilla/Gauntlet	Micro Power
Lords of Time	Level 9
Manic Miner	Software Projects
Map Rally	B.E.S.
Masterfile	Campbell
Message from Andromeda	Interceptor
Millionaire	Incentive
Money Manager	Connect Systems
Mountain Palace Adventure	Duckworth
Mr Freeze	Firebird
Mr Wong's Loopy Laundry	Artic Computing
Music Composer	Kuma
Mutant Monty	Artic Computing
Mystery of the Java Star	Shards
Number Painter	ASK
Osprey	B.E.S.
Pinball Wizard	C.P.
Project Planner	Amsoft
Psychedelia	Llamasoft
Punchy	Mr Micro
Pyjamarama	Microgen
Return to Eden	Level 9
Roland in Space	GEM
Roland in the Caves	Amsoft
Roland Goes Square Bashing	Durell
Rollaball	Timeslip
Royal Quest	Timeslip
Screen Designer	DJL
Sir Lancelot	Melbourne House
Smuggler's Cove	Camel
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
Superchess	C.P.
Survivor	Anirog
Tasword	Tasman
Test Match	CRL
The Moors Challenge	Timeslip
The Prize	Arcade
The Wild Bunch	Firebird
Timebomb	Black Knight
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

The elements of CP/M – in the eighth of our Amstrad quick reference charts

CP/M is a disc operating system concerned with handling files and the peripherals used to create and store them. Developed by Digital Research, CP/M stands for Control Program for Microcomputers.

```
|CPM
```

puts the micro into CP/M, pulling the data from the outer two tracks of a system disc and storing it in memory. The prompt:

```
A>
```

or

```
B>
```

appears showing that the micro is in CP/M's console mode and indicating the default drive. Typing A: or B: in response to the prompt will change the default drive.

In this state CP/M accepts four commands directly from the keyboard without looking to the master disc for further software. Other "commands", known as transient commands, are available as utilities on the master disc. CP/M can call these by name.

Entering:

```
AMSDOS
```

in response to the prompt pulls the transient program AMSDOS.COM off the master disc, ending CP/M.

ERA erases uinprotected files from the directory.

```
ERA FRED
```

gets rid of the file named FRED. Notice that CP/M doesn't require the inverted commas around the filename, except when using the CLOAD and CSAVE utilities.

Wildcards can be used:

```
ERA *.AUG
```

gets rid of all files ending in AUG. Beware! There is no warning or second chance. The file is erased without a message. Only when:

```
ERA *.*
```

is used to get rid of all the files on a disc is there a prompt to give a second chance.

ERRORS: CP/M isn't very friendly when you make a mistake. Type in:

```
dri
```

in response to the prompt and all you get is:

```
DRI?
```

to inform you of the error of your ways. Notice that it accepts lower case letters but forces them to upper case. The same applies to filenames.

KEYBOARD CONTROL: The following key combinations can be used in CP/M:

CTRL+C returns to console mode when used at beginning of a line. Some utilities use it as an abort signal.

CTRL+P toggles the printer on and off.

CTRL+S stops screen output, such as produced by TYPE.

CTRL+Z signals end of text – useful with the PIP utility.

TYPE has the specified file displayed on the screen.

```
TYPE text
```

has the contents of *text* appearing on the monitor. This command only works with Ascii files. With other types of file there are unpredictable results.

REN renames files.

```
REN newname=oldname
```

re-christens *oldname* as *newname*. So:

```
REN certain=possible
```

changes *possible* into *certain*.

If the new name already exists or the file is read-only then the command fails.

FILENAMES follow the pattern used for AMSDOS. A: and B: can be used as prefixes to temporarily overide, but not change, the default drive.

DIR displays the names held in the directory of the disc in historical order. Wildcards can be used.

DIR by itself gives the whole directory.

DIR*.BAK gives all the files with filetype BAK.

DIR B:*.BAS gives all the filenames with filetype BAS on the disc in drive B.

RE FOR THE AMSTRAD NOW PRESENT -

UTOPIA £29.95

BASIC UTILITIES ROM
(Prod Code EU)

40K USER RAM

Beebug's 'TOOLKIT' is the standard utilities ROM for the BBC Micro and has sold thousands of copies. Now the author has written an Amstrad version. Available only on ROM the program contains numerous Basic Programming AIDS including search/replace within Basic program, listing basic variables, moving basic lines, load, save, verify, type, dump, format, copy and much more.

ALL INCL PRICE £29.95

AND GREAT NEWS FOR MACHINE-CODE ENTHUSIASTS:

MAXAM

IN CARTRIDGE **NOW ONLY** **£49.95 (incl VAT, p&p)**
NOW ALSO AVAILABLE IN **ROM ALONE** FOR ONLY £39.95
ALL VERSIONS NOW CONTAIN **FULL** SPECIFICATION
DISC £26.95 TAPE £19.95

PRESS COMMENT

"Innovative device this article finished on the ARNOR editor..... well worth the money!" **- AMSTRAD COMPUTER USER**

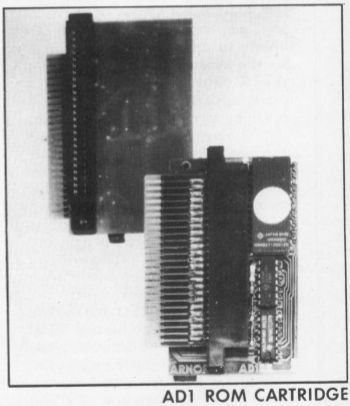
"Assemblers.....look no further ARNOR is the best I have seen" **- COMPUTING W.T. AMSTRAD**

"Absolute magic! - ARNOR must be the market leaders" **- POPULAR COMPUTING WEEKLY**

"Quite Special.....difficult to match" **- COMPUTING TODAY**

"ARNOR are to be congratulated on a superb job.....definitely the best" **- HOME COMPUTING WEEKLY**

"A product no serious AMSTRAD user can afford to be without" **- YOUR COMPUTER**



AD1 ROM CARTRIDGE

IN GOOD COMPUTER STORES EVERYWHERE - OR DIRECT.....

PLEASE SEND ME
(PRODUCT CODE) (PRICE)

_____ _____
_____ _____
_____ _____

Total £_____

I enclose Cheque/PO for £_____
(PAYABLE ARNOR LTD)

OR
Please debit my Access/Visa

Card No [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

NAME:_____ SIGNATURE:_____

ADDRESS_____

CWA 10

ALL TRADE/CREDIT CARD SALES, ENQUIRIES ETC 01-688-6223
SEND TO ARNOR Ltd Dept.□, THE STUDIO, LEDBURY PLACE, CROYDON, SURREY CRO 1ET

GEOF TURNER and MICHAEL NOELS
conclude their series on basic graphics techniques

Wrapping it all up with a neat bit of rubber banding

WE'VE already seen how useful the EOR function can be when applied to our graphics routines. With its ability to draw and erase shapes on the screen without disturbing the background, it's useful in many applications.

Let's just recap on how it works:

In Program I we're drawing a solid red rectangle. When a key is presseo, a blue line cuts through the centre of the rectangle to divide it equally into two pieces.

Press a key again, the line is removed and we end up with our original rectangle. This will be repeated over and over again if we continue to press a key.

All we are doing is EORing the line on and off again. Line 80 has asked the micro to draw in EOR mode while line 90 waits for a key to be pressed. Then the line is drawn over again.

As you'll remember, EORing something twice returns things to the original state.

Suppose we wish to cut the rectangle into two smaller rectangles, but we're not quite sure where we wish to place the cut.

Program II is similar to the previous program, but this time we're given the opportunity to alter the position of the dividing line.

The line is originally placed in the centre of the rectangle as before, but by pressing either of the up or down cursor keys, the lines may be moved into another position.

The program works like this. Lines 10 to 60 are used to draw the rectangle while line 70 fixes the original value of *y* for the line. Line 80 calls the subroutine used to draw the blue line.

```
10 REM PROGRAM I
20 MODE 0
30 FOR strip=100 TO 500 STEP 4
40 MOVE strip,0
50 DRAW strip,300,3
60 NEXT
70 WHILE -1
80 PRINT CHR$(23);CHR$(1);
90  WHILE INKEY$="":WEND
100 MOVE 100,150
110 DRAW 500,150,3
120 WEND
```

Program I

```
10 REM PROGRAM II
20 MODE 0
30 FOR strip=100 TO 500 STEP 4
40 MOVE strip,0
50 DRAW strip,300,3
60 NEXT
70 y=150
80 GOSUB 130
90 WHILE -1
100 IF INKEY(0)=0 THEN GOSUB 130:y=y+
2:GOSUB 130
110 IF INKEY(2)=0 THEN GOSUB 130:y=y-
2:GOSUB 130
120 WEND
130 IF y<0 THEN y=0 :RETURN
140 IF y>300 THEN y=300 :RETURN
150 PRINT CHR$(23);CHR$(1);
160 MOVE 100,y
170 DRAW 500,y,3
180 RETURN
```

Program II

When it detects that one of the cursor keys has been pressed at lines 100 or 110, the program jumps to the subroutine at line 130, and draws the line again, which has the effect of erasing it.

The value of the variable *y* is either increased or decreased, depending on which key has been pressed, and then the subroutine is used yet again to redraw the line in its new position.

The process is repeated every time a cursor key is pressed. Lines 130 and 140 simply prevent the line going past the top or bottom edges of the rectangle.

In Program II we've used the computer to help us to place the dividing line in the position that we want it. This is in fact the first step to computer aided design – usually referred to as CAD.

We can take this a stage further now. Imagine you are an architect and you wish to decide on the position of the front door of a house.

Program III draws your house, represented by the red rectangle, and a green door is placed on the screen to the left of it. Using the left and right cursor keys, the door can be moved across the house until a suitable position is found.

Be wary of one thing when you run this month's programs. The graphics logic used in previous programs may affect their working.

You might have to resort to pressing Ctrl, Shift and Escape to reset the micro. It may be heavy handed, but it works.

Program III is very similar to the previous program, but notice that we are now moving a solid object rather than just a single line.

However it's not necessary to repeatedly draw the complete door. Instead, we are simply EORing the leftmost and rightmost edges of the door. This makes it look like we are moving the complete door across the screen.

```
10 REM PROGRAM III
20 MODE 0
30 INK 15,18
40 FOR strip=100 TO 500 STEP 4
50 MOVE strip,0
60 DRAW strip,300,3
70 NEXT
80 PRINT CHR$(23);CHR$(1);
90 FOR strip=0 TO 48 STEP 4
100 MOVE strip,0
110 DRAW strip,150,12
120 NEXT
130 x=0
140 WHILE -1
150 IF INKEY(1)=0 THEN GOSUB 180:x=x+
4
160 IF INKEY(8)=0 THEN x=x-4:GOSUB 18
0
170 WEND
180 MOVE x,0
190 DRAW x,150,12
200 MOVE x+52,0
210 DRAW x+52,150,12
220 RETURN
```

Program III

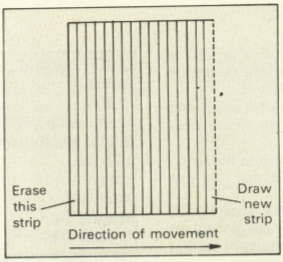


Figure I: How the door moves

Figure I shows how the door consists of a number of vertical strips. To move it to the right, all we need to do is to erase the leftmost strip and draw a new strip on the right edge of the door.

Notice that lines 150 and 160, which are used to detect left or right movement, are different.

One of them changes the variable *x* before calling the subroutine, the other alters it after calling the subroutine. This allows us to use the one subroutine for both left and right movement.

Although these programs are very simple examples, the same basic principles can be used in more complex design programs.

Program IV is a rather more sophisticated house design program.

Here we're dispensing with the use of colour and simply using line or wire frame graphics.

The outline of a house is drawn on the screen, and we are given the opportunity to add the doors and windows in any position we like. After the house is drawn – lines 40 to 130 – the program waits for either key D or W to be pressed. Pressing D will result in a door being drawn, W a window.

When the shape appears, it can be moved around the screen by using the cursor keys.

The technique used is similar to the previous programs. When a cursor key is detected (lines

```
10 REM PROGRAM IV
20 MODE 0
30 REM draw house
40 MOVE 0,50
50 DRAW 639,50
60 MOVE 200,50
70 DRAW 200,300
80 DRAW 600,300
90 DRAW 600,50
100 MOVE 200,300
110 DRAW 250,350
120 DRAW 550,350
130 DRAW 600,300
140 WHILE -1
150 WHILE shape=0
160 IF INKEY(59)=0 THEN shape=1
170 IF INKEY(61)=0 THEN shape=2
180 WEND
190 s=100
200 code=1
210 x=0:y=399
220 GOSUB 330
230 WHILE code=1
240 IF INKEY(39)=0 THEN GOSUB 330:s=s
-2:GOSUB 330
250 IF INKEY(31)=0 THEN GOSUB 330:s=s
+2:GOSUB 330
260 IF INKEY(0)=0 THEN GOSUB 330:y=y+
2:GOSUB 330
270 IF INKEY(2)=0 THEN GOSUB 330:y=y-
2:GOSUB 330
280 IF INKEY(8)=0 THEN GOSUB 330:x=x-
4:GOSUB 330
290 IF INKEY(1)=0 THEN GOSUB 330:x=x+
4:GOSUB 330
300 IF INKEY(9)=0 THEN code=0:GOSUB 3
30:shape=0
310 WEND
320 WEND
330 PRINT CHR$(23);CHR$(code);
340 MOVE x,y
350 IF shape=2 THEN 450
360 DRAWR s,0
370 DRAWR 0,-s
380 DRAWR -s,0
390 DRAWR 0,s
400 MOVER 0.5*s,0
410 DRAWR 0,-s
420 MOVER -0.5*s,0.5*s
430 DRAWR s,0
440 RETURN
450 DRAWR 0.75*s,0
460 DRAWR 0,-1.5*s
470 DRAWR -0.75*s,0
480 DRAWR 0,1.5*s
490 MOVER 0.15*s,-0.2*s
500 DRAWR 0.45*s,0
510 DRAWR 0,-0.45*s
520 DRAWR -0.45*s,0
530 DRAWR 0,0.45*s
540 RETURN
```

Program IV

240-300), the subroutine at 330 is called to erase the shape before its new position is calculated.

The subroutine is called again, and the shape is redrawn in its new position.

As well as being able to move the objects around the screen, we can also vary their size.

Pressing the < or > keys will make the shape smaller or larger. We do this by altering the variable *s* so that this time the shape is redrawn in a different size.

Having placed our door or window in position it can then be fixed by pressing the Copy key.

This changes the value of the variable *code* to zero, which results in the shape being redrawn with normal logic. This gives us a permanent drawing.

After using the Copy key, the cursor returns to the top left hand corner of the screen ready to collect another shape when W or D is pressed.

Perhaps you might like to add other shapes of your own design to the program. These should be added to the subroutine from line 540, and the RETURN statement at line 540 will need to be repositioned at the new end of the subroutine.

Your new object should make the value of the variable *shape* equal to 3 when the appropriate key is pressed.

Finally you will need to redirect the program to the last part of the subroutine at line 350.

The EOR function is quite useful in making objects appear to move around the screen. In reality we are of course simply erasing the object in its old position and redrawing it in a new position.

This technique is often used in games applications. Program V draws a rotating dial which could be used as a scanner or radar screen in your space games.

Lines 40 to 60 draw a circle. Then a line representing the pointer is drawn from the centre to the perimeter of the circle.

The pointer will now rotate under the control of the left and right cursor keys.

Using the familiar method, the subroutine at line 130 repeatedly EORs the pointer off and on. Lines 100 and 110 decide which way the pointer should go by altering the value of *angle*.

Although in this program the movement is controlled by the cursor keys, we could use variables within the program to move the pointer.

If you've ever used a flight simulator program, you will probably have seen this technique used to display some of the aircraft's instruments.

In Program VI we have made use of the Amstrad's built in timers to produce a real time clock. The second hand is moved every second using timer number 1.

Line 100 tells the program to jump to the subroutine at line 130 every second and go on to move the blue seconds hand.

Similarly line 110 tells the program to divert to the subroutine at line 150 once every minute and so move the red minute hand. Timer number 2 is used to move the minute hand.

I haven't included an hour hand – perhaps you would like to add this, using timer number 3. The only problem is, you'll have to hang around

```
10 REM PROGRAM V
20 MODE 1
30 DEG
40 FOR angle=0 TO 360
50 PLOT 100+SIN(angle)+320,100+COS(an
gle)+200
60 NEXT
70 angle=1
80 GOSUB 130
90 WHILE -1
100 IF INKEY(1)=0 THEN GOSUB 130:angl
e=angle+1:GOSUB 130
110 IF INKEY(8)=0 THEN GOSUB 130:angl
e=angle-1:GOSUB 130
120 WEND
130 PRINT CHR$(23);CHR$(1);
140 IF angle=361 THEN angle=1
150 IF angle=0 THEN angle=360
160 MOVE 320,200
170 DRAW 100+SIN(angle)+320,100+COS(a
ngle)+200
180 RETURN
```

Program V

```
10 REM PROGRAM VI
20 MODE 1
30 DEG
40 FOR angle=0 TO 360
50 PLOT 100+SIN(angle)+320,100+COS(an
gle)+200,1
60 NEXT
70 a=1:s=1
80 GOSUB 170
90 GOSUB 210
100 EVERY 50,0 GOSUB 130
110 EVERY 3000,1 GOSUB 150
120 WHILE INKEY$="":WEND
130 GOSUB 170:s=s+6:GOSUB 170
140 RETURN
150 GOSUB 210:a=a+6:GOSUB 210
160 RETURN
170 PRINT CHR$(23);CHR$(1);
180 MOVE 320,200
190 DRAW 100+SIN(s)+320,100+COS(s)+20
0,2
200 RETURN
210 MOVE 320,200
220 DRAW 75+SIN(a)+320,75+COS(a)+200,
3
230 RETURN
```

Program VI

```
10 REM PROGRAM VII
20 MODE 1
30 INK 3,24
40 code=1
50 x=320
60 y=200
70 a=400
80 b=300
90 GOSUB 170
100 WHILE -1
110 IF INKEY(0)=0 THEN GOSUB 170:b=b+
4:GOSUB 170
120 IF INKEY(2)=0 THEN GOSUB 170:b=b-
4:GOSUB 170
130 IF INKEY(1)=0 THEN GOSUB 170:a=a+
4:GOSUB 170
140 IF INKEY(8)=0 THEN GOSUB 170:a=a-
4:GOSUB 170
160 WEND
170 PRINT CHR$(23);CHR$(code);
180 MOVE x,y
190 DRAW a,b,code+1
210 MOVE a-10,b
220 DRAW a+10,b
230 MOVE a,b-10
240 DRAW a,b+10
250 RETURN
```

Program VII

for one hour to see if it works.

We'll move on now to the technique known as rubber banding. With this it is possible to produce line drawings on the screen using the cursor keys – a sort of computerised Etch-a-Sketch.

Program VII draws a line with one end fixed to a point at the centre of the screen. The other end of the line is the free end and is marked with a crosshair cursor.

This cursor can be moved around the screen using, unsurprisingly, the Amstrad's cursor keys.

You will see how the free end of the line appears to stretch and shrink as it attempts to follow the cursor. It appears to act like a piece of elastic, hence the name rubber banding.

The line is movable because we are once again using the EOR function which repeatedly erases and redraws the line in its new position.

The program uses two sets of variables to mark the ends of the line. *x* and *y* are the coordinates of the fixed end, while *a* and *b* are the coordinates of the free end. The subroutine at line 170 simply draws a line between *x,y* and *a,b* using EOR logic.

We can arrange to fix the line in any position by adding the following lines to Program VII:

```
150 IF INKEY(9)=0 THEN GOSUB 170
:code=0:GOSUB 170:code=1:GOSUB 170
200 IF code=0 THEN RETURN
```

Now when we press the Copy key, the line is first erased then redrawn in normal logic, and becomes permanently fixed on the screen.

Line 200 prevents the cursor from being permanently fixed. The line is finally redrawn in EOR logic so that it will continue to respond to the cursor keys.

We can therefore fix as many lines as we like with the Copy key. Table I shows the values used in graphics logic selection.

Parameter	Function
0	Normal
1	EOR
2	AND
3	OR

Table I

With Program VII, all the lines are drawn from a fixed point at the centre of the screen. However we may wish to move the fixed end of the line as well as the free end.

We can modify it once again by changing line 150 to:

```
150 IF INKEY(9)=0 THEN code=0 :GOSUB
170 :x=a :y=b :code=1
```

Now, when the line is fixed, the position of the fixed end is moved to the last position of the free end. In this way continuous lines can be drawn,

and we can build up pictures.

In an earlier program we drew the outline of a house using data in the program. We can now draw the same house or, indeed, any other shape simply by positioning the cursor and drawing the appropriate lines.

The drawback with Program VII is that we have to draw a continuous line. There are no facilities to lift the pen from the paper and move it to another position. Also we can't erase a line once it has been fixed.

Program VIII is similar to the previous program, but we have now added the missing facilities which provide us with a complete rubber banding design program. There are several keys used in this program and these are shown in Table II.

KEY	ACTION
CURSORS	MOVE PEN
U	PEN UP
D	PEN DOWN
DEL	DELETE LINE
COPY	FIX LINE

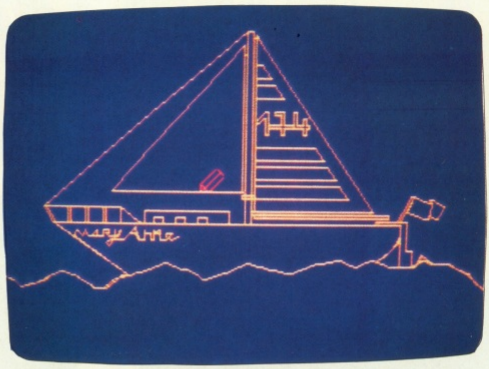
Table I

This program is a little more complex than the previous one. You might like to spend some time working through it to understand fully how it works.

You should have plenty of time as this is the last in the series on basic graphics techniques.

We hope that you've enjoyed it and learnt a lot. If you want to learn more, then you'll have to delve into the world of machine code. The excellent articles by Mike Bibby and Roland Waddilove should help here.

And, in the meantime, if you come up with any graphics masterpieces let us have a look at them here at *Computing with the Amstrad.*



Rubber banding in action using Program VIII

```
10 REM PROGRAM VIII
20 REM INITIALISE VARIABLES
30 MODE 1
40 INK 3,24
50 code=1
60 nib=1
70 x=320
80 y=200
90 a=320
100 b=200
110 REM DRAW CURSOR
120 GOSUB 240
130 REM DETECT KEYS
140 WHILE-1
150 IF INKEY(0)=0 THEN GOSUB 370:b=b+
4:GOSUB 370
160 IF INKEY(2)=0 THEN GOSUB 370:b=b-
4:GOSUB 370
170 IF INKEY(1)=0 THEN GOSUB 370:a=a+
4:GOSUB 370
180 IF INKEY(8)=0 THEN GOSUB 370:a=a-
4:GOSUB 370
190 IF INKEY(9)=0 AND nib=1 THEN GOSU
B 440
200 IF INKEY(42)=0 AND nib=1 THEN GOS
UB 500
210 IF INKEY(61)=0 AND nib=0 THEN GOS
UB 500
220 IF INKEY(79)=0 THEN GOSUB 560
230 WEND
240 REM DRAW CURSOR
250 PRINT CHR$(23);CHR$(1);
260 MOVE A,B
270 DRAWR 0,10,2
280 DRAWR 20,20
290 DRAWR 10,-10
300 DRAWR -20,-20
310 DRAWR -10,0
320 MOVER 0,10
330 DRAWR 10,-10
340 MOVER -5,5
350 DRAWR 20,20
360 RETURN
370 REM DRAW LINE
380 IF nib=0 THEN GOTO 420
390 PRINT CHR$(23);CHR$(1);
400 MOVE x,y
410 DRAW a,b
420 GOSUB 240
430 RETURN
440 REM FIX LINE
450 PRINT CHR$(23);CHR$(0);
460 MOVE x,y
470 DRAW a,b,1
480 x=a:y=b
490 RETURN
500 REM RAISE / LOWER PEN
510 GOSUB 370
520 IF nib=1 THEN nib=0 ELSE nib=1
530 x=a:y=b
540 GOSUB 370
550 RETURN
560 REM DELETE LINE
570 PRINT CHR$(23);CHR$(0);
580 MOVE a,b
590 DRAW x,y,0
600 x=a:y=b
610 RETURN
```

Program VIII

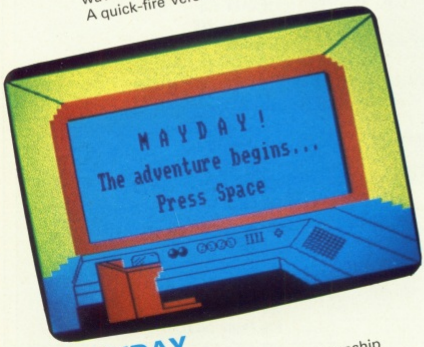
4 GREAT GAMES FOR THE PRICE OF ONE!

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!

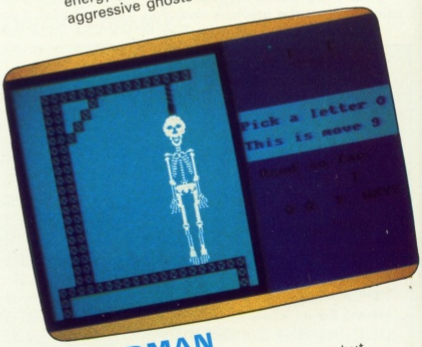
SNAPMAN

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 £6.95 Overseas/Europe *6048* ☐
3" Disc £9.95 £10.95 Overseas/Europe *6049* ☐

Payment: please indicate method (✓)

☐ Access/Mastercard/Eurocard

No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Barclaycard/Visa

No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Expiry date: /

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____ Signed _____

Address _____

_____

_____

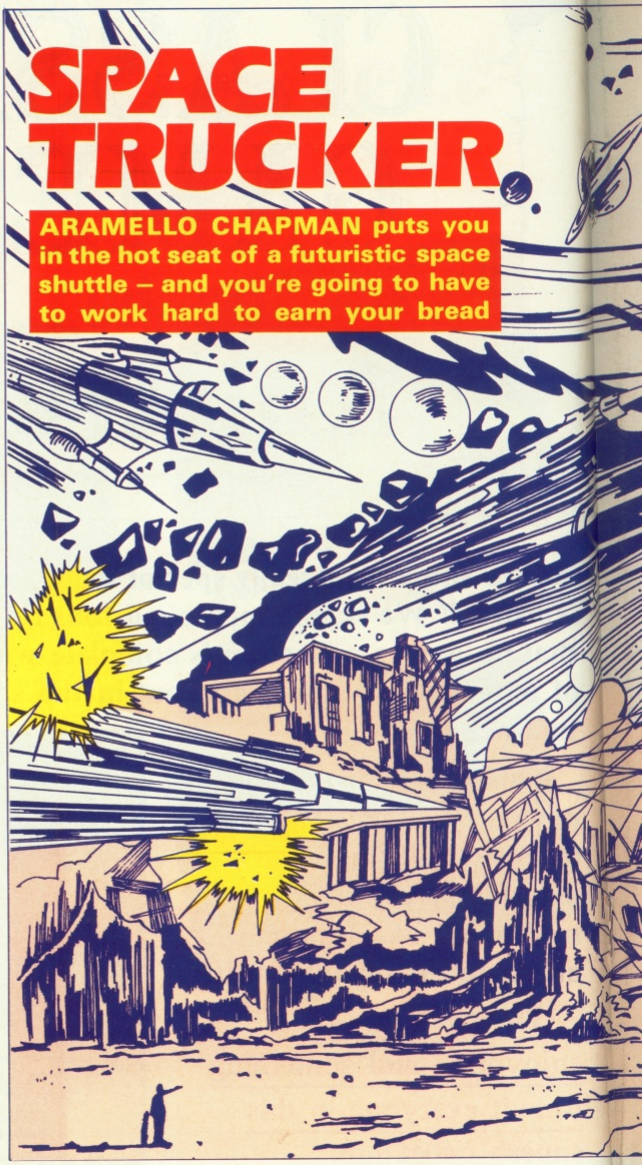
Send to:
***Computing with the Amstrad*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

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

Don't forget to quote your credit card number and full address. CA11

SPACE TRUCKER

ARAMELLO CHAPMAN puts you in the hot seat of a futuristic space shuttle – and you're going to have to work hard to earn your bread



AFTER the last Great War in 2095 in which the whole of the planet Earth's northern hemisphere was wiped out, a Brazilian-based authority took control of all the space colonies.

The resulting company was named after the large colony of Porto Quento.

You are a Space Trucker in the Porto Quento Freight Company. As a newcomer to the firm, not particularly good at your job, you are usually given the outlying planets in the galaxy for your regular round.

Using the interstellar shuttle, you must ferry cargo from the orbiting Mothership to the planet's landing-pad.

This may seem at first to be quite a mundane task, but I haven't yet mentioned the alien traffic. If you think it's hectic on our roads, wait till you meet these space-hogs.

To add to your problems there is also an awful lot of debris floating around up there which was dumped last time Mac-Alpines paid a visit.

You've got only three shuttles at your disposal, so you must be extremely careful to avoid contact with anything likely to cause damage and put you out of commission.

Gravity will pull you down automatically once you leave the Mothership, and to counter this effect you must use the shuttle's thruster units.

You only have a limited amount of fuel, so use it thoughtfully.

Each time you successfully complete a return trip you score according to the amount of fuel remaining in the shuttle. Every five successful trips your fuel is increased by 20 units.

Good luck – you're going to need it.

KEYS

Z	Move left
X	Move right
Space	Thrust
H	Hold
R	Restart

ROUTINES

120	Main program loop.
200	Moves shuttle and checks for collisions.
390	Moves aliens, rocks and Mothership.
480	Decreases leaves and prints explosions.
630	Increases score and changes flight direction.
690	Increases score and changes flight direction.
770	Prints the screen.
930	Loads arrays with character patterns.
1220	Sets up characters.
1560	Sets up variables.
1690	Instructions.
1960	Pokes machine code into memory.
2130	Prints high-score table.

VARIABLES

ale$	Array holding traffic (rocks).
na$	Array holding high-score names.
hs	Array holding high score.
basex	X coordinate of landing-pad.
score	Score.
level	Current screen.
sped	Speed of traffic.
mothx	X coordinate of Mothership.
shipx	X coordinate of shuttle.
shipy	Y coordinate of shuttle.
fuel	Fuel remaining.
lives	Number of shuttles left.
dir	Direction of shuttle.
newx,newy	Shuttle's new coordinates.
left	Address of left scroll.
right	Address of right scroll.
char	Address of read screen routine.
th	Thrust used (1=yes 0=no).

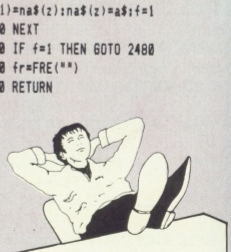
```
1 REM******* Space Trucker ********
2 REM
3 REM(c)Computing with the Amstrad
4 REM
5 REM**** by Aramello Chapman ****
10 fr=FRE(""):CLEAR:DIM ale$(16),na$(
8),hs(8)
20 FOR f=1 TO 8:hs(f)=(1000-((f*100)+
100)):NEXT :FOR f=1 TO 8 STEP 2:na$(f
)="SPACE":na$(f+1)="Trucker":NEXT
30 GOSUB 1220:REM U.D.G
40 GOSUB 1960:REM Machine Code
50 INK 14,24:INK 1,26:INK 0,0:INK 15,
13:MODE 1:PEN 2:LOCATE 15,10:PRINT"Pl
ease Wait!!"
```

```
60 GOSUB 930:REM Create Traffic
70 GOSUB 1690:REM Instructions
80 GOSUB 1560:REM Variables
90 basex=INT(RND*15)+1
100 GOSUB 770:REM Set Up Screen
110 LOCATE 1,2:PEN 13:PRINT score
120 REM******************************
130 REM******* Main Loop *********
140 IF dir=1 AND shipx=basex+2 AND sh
ipy=21 THEN GOTO 630
150 IF dir=-1 AND shipx=mothx AND shi
py=5 THEN GOTO 690
160 IF INKEY(44)=0 THEN GOTO 180
170 GOTO 200
180 IF INKEY(60)<>0 THEN GOTO 180
190 GOTO 120
200 REM******************************
210 REM***** Move Spacecraft *****
220 PEN 1:FOR f=1 TO 100-sped:NEXT f
230 IF INKEY(47)=0 AND fuel>0 THEN GO
SUB 350:th=1:GOTO 260
240 newx=newx+(INKEY(71)=0 AND shipx>
2)-(INKEY(63)=0 AND shipx(19):newy=ne
wy+(dir AND shipy)*4)
250 SOUND 1,20+newy,5,10,3,1
260 IF dir=1 AND (newy=10 OR newy=17
OR newy=13 OR newy=20) OR dir=-1 AND
(newy=8 OR newy=15 OR newy=11 OR newy
=18) THEN 330
270 IF th=1 AND (shipy=9 OR shipy=12
OR shipy=16 OR shipy=19) OR shipy=4 T
HEN GOTO 300
280 GOSUB 410:LOCATE shipx,shipy:PRIN
T" ":LOCATE newx,newy:CALL char:PRINT
 CHR$(232):shipx=newx:shipy=newy:cr=P
EEK(&A08D):IF cr()32 THEN GOTO 500
290 GOTO 190
300 IF cr()32 THEN GOTO 500
310 LOCATE shipx,shipy:PRINT" ":GOSUB
 410:LOCATE newx,newy:CALL char:cr=PE
EK(&A08D):IF cr()32 THEN 500 ELSE shi
px=newx:shipy=newy:LOCATE shipx,shipy
:PRINT CHR$(232);
320 th=0:GOTO 190
330 LOCATE shipx,shipy:PRINT" ":shipx
=newx:shipy=newy:LOCATE shipx,shipy:G
OSUB 410:PRINT CHR$(232)
340 GOTO 190
350 FOR f=1 TO 10:SOUND 1,0,5,12,0,0,
5:NEXT
360 fuel=fuel-1
370 LOCATE 9,2:PEN 13:PRINT fuel:PEN
1
380 RETURN
390 REM******************************
400 REM****** Move Traffic *******
410 IF shipy()9 OR shipy()12 OR shipy
()16 OR shipy()19 OR shipy()4 THEN GO
TO 430
420 LOCATE newx,newy:PRINT" ":GOSUB 4
40:LOCATE newx,newy:CALL char:cr=PEEK
(&A08D):IF cr()32 THEN GOTO 500 ELSE
LOCATE newx,newy:PRINT CHR$(232):GOTO
 450
430 GOSUB 440:GOTO 450
440 FOR f=1 TO 4:CALL left,8:CALL rig
ht,11:CALL left,15:CALL right,18:CALL
 left,3:NEXT:RETURN
450 mothx=mothx-1
460 IF mothx(1 THEN mothx=20
470 RETURN
480 REM******************************
490 REM****** Death Routine ******
500 LET shipx=newx:LET shipy=newy:PEN
 3:LOCATE shipx,shipy:PRINT CHR$(250)

510 SOUND 1,0,0,15,3,2,1:SOUND 2,0,0,
15,3,1,2
520 FOR f=1 TO 100:NEXT
530 PEN 3:LOCATE shipx-1,shipy:PRINT
CHR$(252);CHR$(250);CHR$(253)
540 SOUND 1,0,0,15,3,3,2:SOUND 2,0,0,
15,1,2,4
550 FOR f=1 TO 200:NEXT
560 PEN 3:LOCATE shipx-1,shipy:PRINT
" "CHR$(251)" "
570 SOUND 1,0,0,15,3,4,1
580 FOR f=1 TO 2000:NEXT
590 lives=lives-1
600 IF lives=0 THEN FOR f=10 TO 14:LO
CATE 6,f:PRINT STRING$(9," "):NEXT:PE
N 3:LOCATE 7,11:PRINT"G A M E":PEN 12
:LOCATE 7,13:PRINT"O V E R":FOR f=1 T
O 3000:NEXT:GOTO 2130
610 shipx=10:shipy=5:dir=1:mothx=10:c
r=32:newx=10:newy=5
620 GOTO 90
630 REM******************************
640 REM****** Landed Routine *****
650 score=score+fuel:SOUND 1,50,0,15,
3,1,3:SOUND 1,300,0,15,3,1,4
660 dir=-1
670 FOR f=1 TO 1000:NEXT
680 GOTO 110
690 REM******************************
700 REM******** Docked ***********
710 shipx=10:shipy=5:level=level+1:ne
wx=10:newy=5
720 LET dir=1:mothx=10
730 FOR f=1 TO fuel
740 score=score+1:LOCATE 1,2:PEN 13:P
RINT score:SOUND 1,(fuel+10)-f,1,15:N
EXT f
750 IF level=5 THEN fuel=fuel+20:sped
=sped+30:IF sped)70 THEN sped=70
760 GOTO 90
770 REM******************************
780 REM****** Set Up Screen *****
790 MODE 0:PAPER 0:CLS:PEN 1:BORDER 0
800 LOCATE 1,23:PEN 9:PRINT STRING$(2
0,CHR$(234))
810 LOCATE 1,24:PRINT STRING$(20,CHR$
(143))
820 LOCATE 1,1:PEN 12:PRINT"SCORE  F
UEL  SHIPS"
830 LOCATE 1,3:PAPER 15:PEN 6:PRINT S
TRING$(20,CHR$(229))
840 LOCATE basex,22:PAPER 0:PEN 3:PRI
NT CHR$(235);:PEN 12:PRINT CHR$(244);
CHR$(245);:PRINT CHR$(246);:PEN 3:PRI
NT CHR$(235):LOCATE basex,23:PRINT CH
R$(236);:PEN 12:PRINT CHR$(247);CHR$(
248);CHR$(249);:PEN 3:PRINT CHR$(236)
850 IF level=1 THEN lev=1 ELSE IF lev
el=2 THEN lev=5 ELSE IF level=3 THEN
lev=9 ELSE IF level=4 THEN lev=13 ELS
E IF level=5 THEN level=1:GOTO 850
860 LOCATE 1,9:PEN 11:PRINT ale$(lev)
870 LOCATE 1,12:PEN 8:PRINT ale$(lev+
1)
880 LOCATE 1,16:PEN 7:PRINT ale$(lev+
2)
890 LOCATE 1,19:PEN 10:PRINT ale$(lev
+3)
900 LOCATE 9,4:PEN 14:PRINT CHR$(225)
CHR$(226)CHR$(227)
910 PEN 13:LOCATE 9,2:PRINT fuel:LOCA
TE 17,2:PRINT lives
920 RETURN
930 REM******************************
940 REM***** Create Traffic *******
950 RESTORE 1050
960 FOR f=1 TO 16
970 ale$(f)=""
980 FOR g=1 TO 20
990 READ n
1000 IF n=0 THEN n=32
1010 ale$(f)=ale$(f)+CHR$(n)
1020 fr=FRE("")
1030 NEXT g,f
1040 RETURN
1050 DATA 242,243,0,0,0,241,0,0,0,0,0
,239,240,0,0,241,0,0,241,0
1060 DATA 241,0,0,0,0,237,238,0,0,241
,0,0,242,243,0,0,0,0,0,0
```

```
1070 DATA 0,0,0,0,241,0,0,0,242,243,0
,0,0,0,0,0,0,241,0
1080 DATA 0,0,0,0,241,0,0,0,237,238,0
,0,0,241,0,0,0,242,243,0
1090 DATA 0,0,0,0,0,0,225,227,0,0,0,2
28,0,0,0,225,227,0,0,0
1100 DATA 0,228,0,0,0,0,230,231,0,0,0
,228,0,228,225,227,0,0,0,0
1110 DATA 0,0,0,0,0,0,230,231,0,0,228
,0,0,225,227,0,0,0,228,0
1120 DATA 0,228,0,0,0,230,231,0,0,0,2
25,227,0,0,0,0,228,0,0,0
1130 DATA 0,242,243,0,0,0,241,0,0,0,2
39,240,0,0,242,243,0,0,0,0
1140 DATA 0,0,0,0,237,238,0,241,0,0,2
39,240,0,0,0,242,243,0,241,0
1150 DATA 0,241,0,241,0,0,242,243,0,0
,0,239,240,0,0,241,0,0,237,238
1160 DATA 0,0,0,0,241,242,243,0,239,2
40,0,0,237,238,241,0,0,0,0,241
1170 DATA 225,227,0,0,228,0,0,230,231
,0,0,228,0,0,0,230,231,0,0,0
1180 DATA 0,0,0,228,0,0,230,231,0,0,2
25,227,0,0,230,231,0,0,228,0
1190 DATA 228,0,0,0,225,227,0,0,230,2
31,0,0,0,228,0,0,0,225,227,0
1200 DATA 225,227,0,0,0,228,0,0,230,2
31,0,0,0,225,227,0,0,0,228,0
1210 GOTO 1210
1220 REM*****************************
1230 REM******* U.D.Gs ************
1240 SYMBOL AFTER 225
1250 SYMBOL 225,0,7,15,57,121,255,3,0
1260 SYMBOL 226,60,255,195,255,255,19
5,153,0
1270 SYMBOL 227,0,224,240,156,158,255
,192,0
1280 SYMBOL 228,195,153,189,231,255,1
89,153,195
1290 SYMBOL 229,34,119,34,0,0,34,119,
34
1300 SYMBOL 230,7,6,31,62,119,248,255
,15
1310 SYMBOL 231,224,96,248,124,238,31
,255,240
1320 SYMBOL 232,24,60,102,255,219,255
,66,231
1330 SYMBOL 233,24,24,60,60,170,36,14
6,168
1340 SYMBOL 234,0,0,0,34,119,255,255,
255
1350 SYMBOL 235,0,0,16,16,16,16,24,24
1360 SYMBOL 236,24,126,44,126,126,254
,246,246
1370 SYMBOL 237,28,239,255,126,108,63
,6
1380 SYMBOL 238,24,126,231,183,254,18
4,248,96
1390 SYMBOL 239,28,98,239,255,126,108
,63,6
1400 SYMBOL 240,24,126,231,183,254,18
4,248,96
1410 SYMBOL 241,60,126,219,175,110,12
2,28,0
1420 SYMBOL 242,8,55,93,247,222,117,6
3,29
1430 SYMBOL 243,216,116,202,119,191,2
54,124,200
1440 SYMBOL 244,0,0,0,0,1,2,7,10
1450 SYMBOL 245,60,126,126,255,255,0,
255,170
1460 SYMBOL 246,0,0,0,0,128,64,224,17
6
1470 SYMBOL 247,15,31,52,52,255,63,25
5,255
1480 SYMBOL 248,255,255,146,146,255,2
55,223,223
1490 SYMBOL 249,240,248,76,76,255,252
,255,253
1500 SYMBOL 250,145,102,62,164,18,73,
102,153
1510 SYMBOL 251,0,72,0,4,64,4,0,66
1520 SYMBOL 252,7,15,126,248,248,126,
15,7
1530 SYMBOL 253,224,240,126,31,31,126
,240,224
1540 SYMBOL 254,198,165,198,165,6,40,
40,16
1550 RETURN
1560 REM*****************************
1570 REM******** Variables ********
1580 score=0
1590 level=1:sped=10
1600 mothx=10
1610 shipx=10:newx=10
1620 shipy=5:newy=5
1630 fuel=50
1640 lives=3
1650 dir=1:cr=32
1660 ENV 1,15,-2,10:ENV 2,15,-1,2:ENV
 3,15,-1,10:ENV 4,15,1,2:ENT 1,15,5,2
1670 fr=FRE("")
1680 RETURN
1690 REM*****************************
1700 REM****** Instructions *******
1710 MODE 1:BORDER 0:GOSUB 1890
1720 LOCATE 1,7:PEN 2:PRINT" The year
 is 2160 and as a Space Truckerin the
 Porto Quento Freight Company,it  is
your job to transport supplies from
orbiting Motherships to the landing p
ad on the planet's surface."
1730 PRINT" The Company has kindly sa
id that pilotswill be paid by the amo
unt of fuel they save on each leg of
 the trip (This also helps to stop the
 pilots joy riding on  company time).
"
1740 PRINT" However the planet which
you have been sent to is just not the
 kind of place tojoy ride  -  as you
will soon find out!"
1750 WHILE INKEY(47)<>0:WEND
1760 LOCATE 1,1:PRINT STRING$(26,11)
1770 GOSUB 1890
1780 LOCATE 1,6:PEN 2:PRINT" There ar
e several hazards which befall a budd
```

```
ing pilot.These include alien starshi
ps and large asteroids.In the search
for higher profits the Company has no
t fitted any weaponry,therefore you
will have to avoid everything."
1790 PRINT" Your small shuttle has a
limited amountof thruster fuel - used
 to counter the effects of gravity.
However it is only replenished every
 five screens so bewareyou only get a
nother 20 units."
1800 PRINT" You can only land on the
top of the pador dock exactly with th
e middle of the Mothership."
1810 WHILE INKEY(47)<>0:WEND
1820 LOCATE 1,1:PRINT STRING$(26,11)
1830 GOSUB 1890
1840 LOCATE 17,7:PEN 3:PRINT"THE KEYS
":LOCATE 17,8:PEN 1:PRINT STRING$(8,C
HR$(131))
1850 LOCATE 14,9:PEN 2:PRINT"'Z'-Move
 Left":LOCATE 14,11:PRINT"'X'-Move Ri
ght":LOCATE 12,13:PRINT"<SPACE>-Thrus
t":LOCATE 14,15:PRINT"'H'-Hold":LOCAT
E 14,17:PRINT"'S'-Restart"
1860 WHILE INKEY(47)<>0:WEND
1870 LOCATE 1,1:PRINT STRING$(26,11)
1880 GOTO 2130
1890 LOCATE 1,1:PAPER 1:PEN 3:PRINT S
TRING$(40,CHR$(229))
1900 FOR f=2 TO 4:LOCATE 1,f:PRINT CH
R$(229);CHR$(229):LOCATE 39,f:PRINT C
HR$(229);CHR$(229):NEXT
1910 LOCATE 1,5:PRINT STRING$(40,CHR$
(229))
1920 LOCATE 15,3:PEN 2:PAPER 0:PRINT"
Space Trucker"
1930 LOCATE 1,20:PAPER 1:PEN 3:PRINT
STRING$(40,CHR$(229));FOR f=21 TO 23:
LOCATE 1,f:PRINT CHR$(229);CHR$(229);
LOCATE 39,f:PRINT CHR$(229);CHR$(229)
:NEXT:LOCATE 1,24:PRINT STRING$(40,CH
R$(229))
1940 LOCATE 10,22:PAPER 0:PRINT"Press
 <SPACE> to continue."
1950 RETURN
1960 REM****************************
*
1970 REM*********Machine Code********
*
1980 hig=&9FFF:address=hig+1:high=hig
+1:RESTORE 2050:left=high:right=high+
50:char=high+107
1990 READ n
2000 IF n=999 THEN READ n:IF n<>sc TH
EN PRINT CHR$(7):PRINT STRING$(6,CHR$
(7)):CLS:LOCATE 11,10:PEN 3:PRINT"ERR
OR IN MACHINE CODE":END ELSE IF n=161
62 THEN sc=0:RETURN
2010 sc=sc+n
2020 POKE address,n
2030 address=address+1
2040 GOTO 1990
2050 REM**********LEFT SCROLL*******
2060 DATA &dd,&6e,&00,&26,&00,&cd,&1a
,&bc,&e5,&7e,&f5,&54,&5d,&eb,&23,&cb,
&f4,&cb,&fc,&0e,&4f,&7e,&12,&23,&cb,&
f4,&cb,&fc,&eb,&23,&cb,&f4,&cb,&fc,&e
b,&d
2070 DATA &20,&ef,&f1,&12,&e1,&7c,&c6
,&08,&67,&e6,&38,&20,&d7,&c9
2080 REM**********RIGHT SCROLL*******
2090 DATA &dd,&6e,&00,&26,&00,&cd,&1a
,&bc,&01,&4f,&00,&09,&cb,&4f,&cb,&fc,
&e5,&7e,&f5,&54,&5d,&2b,&cb,&f4,&cb,&
fc,&0e,&4f,&7e,&12,&2b,&cb,&f4,&cb,&f
c,&eb,&2b,&cb,&f4
2100 DATA &cb,&fc,&eb,&0d,&20,&ef,&f1
,&12,&e1,&7c,&c6,&08,&67,&e6,&38,&20,
&d8,&c9
2110 DATA &cd,&60,&bb,&32,&8d,&A0,&c9
2120 DATA 999,16162
2130 REM**********High Score*********
2140 REM*****************************
2150 MODE 1
2160 LOCATE 4,5:PEN 3:PRINT CHR$(150)
;STRING$(32,CHR$(154));CHR$(156)
2170 LOCATE 16,3:PEN 2:PRINT"HIGH SCO
RE":PEN 3
2180 FOR f=6 TO 15:LOCATE 4,f:PRINT C
HR$(149):LOCATE 37,f:PRINT CHR$(149):
NEXT
2190 LOCATE 4,16:PRINT CHR$(147);STRI
NG$(32,CHR$(154));CHR$(153)
2200 FOR f=1 TO 8
2210 IF score>hs(f) THEN GOSUB 2310:f
=10
2220 NEXT
2230 FOR f=1 TO 8:PEN 1:LOCATE 8,f+6:
PRINT na$(f):LOCATE 18,f+6:PEN 3:PRIN
T".....";hs(f):NEXT
2240 LOCATE 1,17:PRINT STRING$(220,"
")
2250 IF INKEY$<>"" THEN GOTO 2250
2260 PEN 2:LOCATE 1,20:PRINT STRING$(
40,CHR$(154));LOCATE 1,22:PRINT STRIN
G$(40,CHR$(154)):PEN 1:
2270 LOCATE 1,23:PRINT STRING$(40," "
)
2280 LOCATE 11,21:PEN 3:PRINT"PRESS";
:PEN 1:PRINT"< SPACE >";:PEN 3:PRINT"
TO PLAY.":PEN 1
2290 IF INKEY(47)<>0 THEN GOTO 2290
2300 LOCATE 1,1:PRINT STRING$(26,11):
GOTO 80
2310 a$="ABCDEFGHIJKLMNOPQRSTUVWXYZ
#%&()!{}?*+"+CHR$(254)
2320 c=19:LOCATE 1,20:PEN 1:PRINT a$
2330 LOCATE 3,17:PEN 1:PRINT"USE CURS
OR KEY'S LEFT,RIGHT AND COPY":LOCATE
4,18:PRINT"TO SELECT LETTERS.(MAXIMUM
 OF 10.)":PEN 2:LOCATE 1,19:PRINT STR
ING$(40,CHR$(154)):LOCATE 1,22:PRINT
STRING$(40,CHR$(154))
2340 LOCATE 12,23:PEN 3:PRINT"PRESS";
:PEN 1:PRINT"< 'X' >";:PEN 3:PRINT"TO
 EXIT.":PEN 1
2350 x$=""
2360 FOR z=1 TO 10
2370 LOCATE c,21:PEN 2:PRINT" "
2380 IF INKEY(1)=0 AND c<40 THEN c=c+
1
2390 IF INKEY(8)=0 AND c>1 THEN c=c-1
2400 IF INKEY(9)=0 AND c=40 THEN LOCA
TE 7,f+6:PRINT"          ":z=100:GOT
O 2450
2410 IF INKEY(63)=0 THEN z=11:GOTO 24
50
2420 IF INKEY(9)<>0 THEN LOCATE c,21:
PRINT"*":FOR a=1 TO 50:NEXT:GOTO 2370
2430 x$=x$+MID$(a$,c,1):LOCATE 7+z,f+
6:PEN 1:PRINT MID$(a$,c,1)
2440 FOR a=1 TO 200:NEXT
2450 NEXT
2460 IF z=101 THEN GOTO 2350
2470 hs(8)=score:na$(8)=x$
2480 f=0
2490 FOR z=1 TO 7
2500 IF hs(z)<hs(z+1) THEN t=hs(z+1):
hs(z+1)=hs(z):hs(z)=t:a$=na$(z+1):na$
(z+1)=na$(z):na$(z)=a$:f=1
2510 NEXT
2520 IF f=1 THEN GOTO 2480
2530 fr=FRE("")
2540 RETURN
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette. See our special offer on Page 101.

There's no need to peek to keep your sprites doing what they oughta

Part IV of ROLAND WADDILOVE's machine code graphics series

SO far in this short series we've looked at how to print multicoloured sprites, read the keyboard and move them around the screen.

This month we're going to look at simple collision detection. Also I'll show you how to keep track of where the sprite is and keep it within any preset boundaries.

Collision detection is fairly easy if you go about it in the right way – there's no need to start peeking the screen memory.

To keep track of where sprites are on the screen or off, it is best to store their position as a pair of x,y coordinates.

The sprite's coordinates aren't the same as the normal coordinates that we PLOT at though. As the screen is 80 bytes wide and 200 bytes high these are more convenient to use. The top left corner of the screen is 0,0.

When it comes to printing a sprite there is a problem – the coordinates aren't much use for the print routine as it needs a screen address.

There are two ways round this. The hard way would be to calculate the screen address from the coordinates. I've never been much of a mathematician, so my method is simply to keep a record of both the screen address and the coordinates of each sprite.

If you have a look at this month's listing you'll see that it is very similar to the one we used last time.

Two sprites are placed on the screen. One can be moved by pressing the cursor keys and the other is fixed. Program I needs to be run first. This pokes some data to &9000 that's needed for the sprites.

```
10 REM PROGRAM I
20 POKE &9000,8:POKE &9001,4
30 FOR i=0 TO 31
40 READ j:POKE &9002+i,j
50 NEXT
60 REM alien
70 REM Rows=8/Columns=4
80 DATA 4,12,12,8,72,148,104,132,72
90 DATA 156,108,132,28,60,60,44,28
100 DATA 32,16,44,4,48,48,8,8,0,0,4
110 DATA 4,0,0,8
```

Program I

Try moving the sprites around the screen. There are two important points to note. First, the sprite cannot be moved off the screen, either left, right, top or bottom. The second point is that whenever the two sprites collide the Amstrad will beep.

The coordinates of the fixed sprite are stored in *alienxy* and the coordinates of the moving sprite in *yourxy*. The initialisation part of the program starting at &8000 sets the start coordinates and places the sprites on the screen.

In order to keep track of the screen coordinates *up, down, left* and *right* have been modified. When the sprite is to be moved the relevant coordinate is checked to see if the sprite is at the edge of the screen.

Take a look at *up*. The address is stored on the stack and the x,y coordinates in the E and D registers.

After checking that the cursor up key is pressed the y coordinate is transferred to the A register and ANDed with itself to see if it is zero. If this is true the sprite must be at the top of the screen already, so can't be moved any higher. A jump is made to *escape* if it is at the top.

If y is not zero then y is decremented by 2 and &1000 subtracted from the address. The sprite is moved up two pixels at a time.

The other sections – *down, left* and *right* work in exactly the same way, checking for the key then checking the coordinates to see whether it can be moved.

Now that we have the coordinates of each sprite we can easily test for collisions. The collision detection is carried out by *bumped* which returns with the Carry flag set if the collision is true.

The routine needs the coordinates and the sizes of the two sprites, *Collision* sorts this out. The coordinates are placed in *spr1x* and *spr2x*, a four byte block of workspace.

The sizes are passed in the DE and BC registers.

To test whether two sprites are overlapping, first the x coordinates are tested then the y coordinates. Here's the algorithm used, 1 refers to sprite 1, 2 to sprite 2 and h and w are the heights and widths:

IF x1 is less than x2 THEN add w1 to x1, see if this is greater than x2 and RETURN if false ELSE add w2 to x2, see if this is greater than x1 and RETURN if false.

IF y1 is less than y2 THEN add h1 to y1, see if this is greater than y2 and RETURN ELSE add h2 to y2, see if this is greater than y1 and RETURN.

If you're not sure how this works Figure I shows two overlapping sprites. In this case x1 is less than x2 so w1 is added to x1. This is greater than x2 so sprite 1 must be overlapping sprite 2's left hand edge.

The y coordinates are now checked as sprite 1 could be at the top of the screen and sprite 2 at the bottom. In this case y1 is less than y2, so h1 is added to y1. This is greater than y2, so sprite 1 has definitely collided with sprite 2. &8132 to &815D shows how this is done in machine code.

That's all for this month. By now you should now have some powerful machine code routines for writing arcade games. All you need is that elusive original idea.

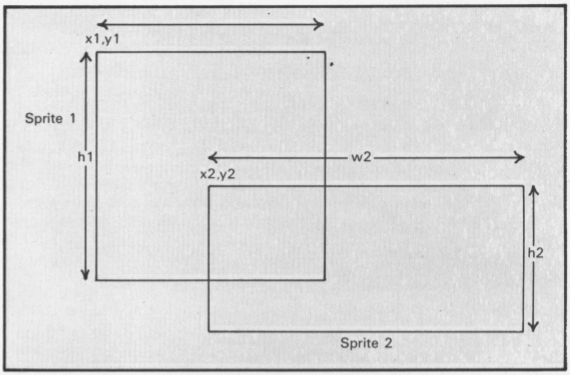


Figure I: Detecting collisions between sprites

Program II

```
RAW Assembler V.3

Pass... 2           ORG &8000

8000:               .new=&80EF
8000:               .old=&80CD
8000:               .newdata=&80F2
8000:               .olddata=&80D0

                    initialise
8000:21 10 08       LD HL,&0810
8003:22 64 81       LD (alienxy),HL
8006:21 60 C0       LD HL,&C060
8009:22 EF 80       LD (new),HL
800C:21 02 90       LD HL,&9002
800F:22 F2 80       LD (newdata),HL
8012:2A 00 90       LD HL,(&9000)
8015:22 5E 81       LD (rows),HL
8018:CD EE 80       CALL put

801B:21 00 C0       LD HL,&C000
801E:E5             PUSH HL
801F:22 CD 80       LD (old),HL
8022:22 EF 80       LD (new),HL
8025:2A 00 90       LD HL,(&9000)
8028:22 5E 81       LD (rows),HL
802B:21 02 90       LD HL,&9002
802E:22 F2 80       LD (newdata),HL
8031:CD EE 80       CALL put
8034:E1             POP HL
8035:11 00 00       LD DE,0

8038:               .start
8038:ED 53 62 81    LD (yourxy),DE
803C:22 60 81       LD (address),HL
803F:22 EF 80       LD (new),HL
8042:21 02 90       LD HL,&9002
8045:22 F2 80       LD (newdata),HL
8048:22 D0 80       LD (olddata),HL
804B:2A 5E 81       LD HL,(rows)
804E:CD 19 BD       CALL &BD19
8051:CD C8 80       CALL print
8054:CD 15 81       CALL collision
8057:2A 60 81       LD HL,(address)
805A:22 CD 80       LD (old),HL
805D:E5             PUSH HL
805E:ED 5B 62 81    LD DE,(yourxy)

                    up
8062:AF             XOR A
8063:CD 1E BB       CALL &BB1E
8066:28 16          JR Z,down
8068:7A             LD A,D
8069:A7             AND A
806A:28 52          JR Z,escape
806C:15             DEC D
806D:15             DEC D
806E:E1             POP HL
806F:7C             LD A,H
8070:D6 10          SUB &10
8072:67             LD H,A
8073:D6 C0          SUB &C0
8075:30 C1          JR NC,start
8077:01 B0 3F       LD BC,&3FB0
807A:09             ADD HL,BC
807B:C3 38 80       JP start

807E:               .down
807E:3E 02          LD A,2
8080:CD 1E BB       CALL &BB1E
8083:28 16          JR Z,left
8085:7A             LD A,D
8086:FE C0          CP 192
8088:28 34          JR Z,escape
808A:14             INC D
808B:14             INC D
808C:E1             POP HL
808D:7C             LD A,H
808E:C6 10          ADD A,&10
8090:67             LD H,A
8091:D2 38 80       JP NC,start
8094:01 50 C0       LD BC,&C050
8097:09             ADD HL,BC
8098:C3 38 80       JP start
```

```
809B:                .left
809B:3E 08           LD A,8
809D:CD 1E BB        CALL &BB1E
80A0:28 0A           JR Z,right
80A2:7B              LD A,E
80A3:A7              AND A
80A4:28 18           JR Z,escape
80A6:1D              DEC E
80A7:E1              POP HL
80A8:2B              DEC HL
80A9:C3 38 80        JP start

80AC:                .right
80AC:3E 01           LD A,1
80AE:CD 1E BB        CALL &BB1E
80B1:28 0B           JR Z,escape
80B3:7B              LD A,E
80B4:FE 4B           CP  75
80B6:28 06           JR Z,escape
80B8:1C              INC E
80B9:E1              POP HL
80BA:23              INC HL
80BB:C3 38 80        JP start

80BE:                .escape
80BE:3E 42           LD A,66
80C0:CD 1E BB        CALL &BB1E
80C3:E1              POP HL
80C4:CA 38 80        JP Z,start
80C7:C9              RET

80C8:                .print
80C8:22 5E 81        LD (rows),HL
80CB:45              LD B,L
80CC:21 00 00        LD HL,0
80CF:11 00 00        LD DE,0
80D2:F3              DI
80D3:                .loop1
80D3:C5              PUSH BC

80D4:3A 5F 81        LD A,(columns)
80D7:47              LD B,A
80D8:E5              PUSH HL
80D9:                .loop2
80D9:1A              LD A,(DE)
80DA:AE              XOR (HL)
80DB:77              LD (HL),A
80DC:23              INC HL
80DD:13              INC DE
80DE:10 F9           DJNZ loop2
80E0:E1              POP HL
80E1:7C              LD A,H
80E2:C6 08           ADD A,&8
80E4:67              LD H,A
80E5:30 04           JR NC,nextrow1
80E7:01 50 C0        LD BC,&C050
80EA:09              ADD HL,BC
80EB:                .nextrow1
80EB:C1              POP BC
80EC:10 E5           DJNZ loop1
80EE:                .put
80EE:21 00 00        LD HL,0
80F1:11 00 00        LD DE,0
80F4:3A 5E 81        LD A,(rows)
80F7:47              LD B,A
80F8:                .loop3
80F8:C5              PUSH BC
80F9:3A 5F 81        LD A,(columns)
80FC:47              LD B,A
80FD:E5              PUSH HL
80FE:                .loop4
80FE:1A              LD A,(DE)
80FF:AE              XOR (HL)
8100:77              LD (HL),A
8101:23              INC HL
8102:13              INC DE
8103:10 F9           DJNZ loop4
8105:E1              POP HL
8106:7C              LD A,H
8107:C6 08           ADD A,&8
8109:67              LD H,A
810A:30 04           JR NC,nextrow2
810C:01 50 C0        LD BC,&C050
810F:09              ADD HL,BC
8110:                .nextrow2
8110:C1              POP BC
8111:10 E5           DJNZ loop3
8113:FB              EI
8114:C9              RET

8115:                .collision
8115:2A 62 81        LD HL,(yourxy)
8118:22 66 81        LD (sprix),HL
811B:2A 64 81        LD HL,(alienxy)
811E:22 68 81        LD (spr2x),HL
8121:ED 5B 00 90     LD DE,(&9000)
8125:ED 4B 00 90     LD BC,(&9000)
8129:CD 32 81        CALL bumped
812C:D0              RET NC
812D:3E 07           LD A,7
812F:C3 5A BB        JP &BB5A

8132:                .bumped
8132:21 68 81        LD HL,spr2x
8135:3A 66 81        LD A,(sprix)
8138:BE              CP  (HL)
8139:38 0A           JR C,c1
813B:7E              LD A,(HL)
813C:80              ADD A,B
813D:47              LD B,A
813E:3A 66 81        LD A,(sprix)
8141:90              SUB B
8142:D0              RET NC
8143:18 05           JR checky
8145:                .c1
8145:82              ADD A,D
8146:57              LD D,A
8147:7E              LD A,(HL)
8148:92              SUB D
8149:D0              RET NC
814A:                .checky
814A:23              INC HL
814B:3A 67 81        LD A,(spr1y)
814E:BE              CP  (HL)
814F:38 08           JR C,c2
8151:7E              LD A,(HL)
8152:81              ADD A,C
8153:4F              LD C,A
8154:3A 67 81        LD A,(spr1y)
8157:91              SUB C
8158:C9              RET
8159:                .c2
8159:83              ADD A,E
815A:5F              LD E,A
815B:7E              LD A,(HL)
815C:93              SUB E
815D:C9              RET

815E:                .rows
815E:00              DEFB 0
815F:                .columns
815F:00              DEFB 0
8160:                .address
8160:00 00           DEFW 0
8162:                .yourxy
8162:00 00           DEFW 0
8164:                .alienxy
8164:00 00           DEFW 0
8166:                .sprix
8166:00              DEFB 0
8167:                .spr1y
8167:00              DEFB 0
8168:                .spr2x
8168:00              DEFB 0
8169:                .spr2y
8169:00              DEFB 0

816A:                END
```

Give your fingers a rest . . .

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

See our special offer on Page 101.

JOHN HUGHES explains the inner workings of Amstrad Basic in this, the first of a two part series

FANTASTIC, isn't it? You plug in your Amstrad, flick a couple of switches, and up comes that reassuring message telling you that Basic is at your command.

In this two part series we'll be taking a look at how the Basic works. The general principles are pretty much the same for most other micros as well, but there are just enough differences to make it very interesting – or frustrating – if you try to transfer what you learn here directly to other machines.

Perhaps the most important difference between Basic and other high level computing languages such as Fortran and Cobol is that Basic is an interpreted language, whereas the latter two are compiled.

That is to say, a program written in Cobol or Fortran is turned into machine code en bloc and then executed. whereas one written in Basic is turned into machine code statement by statement, each statement being executed as it is interpreted. There are compiled Basics, but they are outside the scope of these articles.

This has various effects, both good and bad. The ones that most concern us here are that it makes Basic rather a slow-running language – although this isn't too apparent on most modern machines unless you go in for a lot of screen animation.

It also means that the source program remains in the computer's memory all the time it is being executed, and that in turn means that it is very easy for us to watch it working.

Now it's time to actually poke around in your Amstrad's innards – or PEEK around, to be more precise. Reset your computer with Ctrl+Shift+Esc then enter the following Basic program:

```
10 REM This is a remark
20 PRINT "This is a PRINT statement"
```

Then type in the following line in direct mode. You will probably find it easier to follow this if you are in Mode 2, as this allows so much more on the screen at a time:

```
FOR J=368 TO 428:PRINT PEEK(J);:NEXT
```

and the computer will respond with:

```
23  0  10  0  197  32  84  104  105  115  32
105  115  32  97  32  114  101  109  97  114
107  0  34  0  20  0  191  32  34  84  104
105  115  32  105  115  32  97  32  80  82  73
78  84  32  115  116  97  116  101  109  101
110  116  34  0  0  0  0  0
```

What you are seeing here is what the two lines of Basic program that you have typed in are stored in the computer's memory. Each number represents what is held in one byte of memory, starting from the point at which your program begins.

An important concept to grasp is that, for hardware reasons, computers store many numbers in what humans would regard as the wrong order – the low number first, and the high number second.

It's a bit like 21 (twenty-one) being written down as 12! In this case though, rather than tens and units the hi and lo "columns" are 256 and units respectively. See our machine code series or Bits and Bytes to see why.

You can see this so-called "lo-byte/hi-byte" situation in the first two numbers – 23, 0. These mean that this line of program is 23 bytes long (including the two bytes used for this information.)

The next two numbers, 10, 0, are also lo-byte/hi-byte, and tell us the number of the first line of the program.

After this comes the code 197, which stands for the Basic keyword REM – all Basic keywords are represented in a similar way, and the codes are called tokens. Note, by the way, that the space which you typed in after the line number is not represented in the memory.

Then it gets easy. If you look up the next 17 numbers, as far as 107, in the

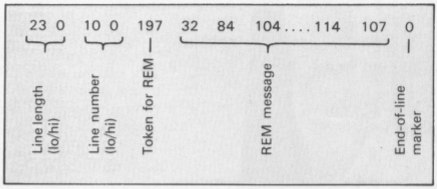


Figure I: How ***10 REM This is a remark*** *is stored*

User Instructions, you will see that they are the Ascii codes for the remaining letters in the statement – 32 is a space, 84 is a capital T and so on.

Finally, the sequence finishes with zero as an end-of-line marker.

All Basic command statements are represented in the same general way. If you're still confused, Figure I may help to make things a little clearer.

The second statement occupies 34 bytes, as you will see from perusing the next two numbers. Then comes the second line number, 20, 0, and the token 191, which stands for PRINT, then a space, 32, a double quote, 34, and so on.

But note that here the word PRINT, which formed part of the message included in the quotation marks, is listed as a series of Ascii values – 80, 82, 73, 78, 84 – rather than as a token, because tokens only stand for instructions that the computer has to obey, and not for similar words which occur inside quotation marks.

As this point you would probably assume that each Basic keyword has its own unique token to go with it, but this is not quite true. If the second line of your original program had been written as:

```
20 ? "This is a PRINT statement"
```

using ? as a substitute for PRINT, you would find that the token for ? is 191, the same as for the keyword spelled out in full.

This is why when you list a program written using ? the computer always replaces it with PRINT – it has no way of knowning which form of the command you originally used.

But the other obvious equivalence, between REM and ', doesn't work in the same way. We've already seen that the token for REM is 197, and you might guess that when you use ' as a keyword it too would have the same token.

But in fact the token is 1, 192 – so, far from saving memory with the shorter form, it actually takes up another byte!

But because the tokens are different, the Basic interpreter is able to distinguish between them – ' isn't turned into REM in the same way as ? is turned into PRINT when you list a program.

Before you start to see how many tokens you can work out for yourself, there are a couple of curiosities which are worth noticing.

The first is merely a historical oddity, but the second may help you to write programs that are easier to follow when debugging time rolls round.

You probably know that when Basic was first developed it required all assignments to be seen coming – that is, instead of writing A=10, you had to write LET A=10.

Until quite recently some computers still used interpreters which inserted the token for LET (165) even if it didn't appear in the original program.

The Amstrad doesn't put it in unless you have used it yourself, so that you really do save memory space by not using it – which you don't if you use ' and ? instead of REM and PRINT.

The other oddity, which the Amstrad shares with very many other micros, is that anything added to the right-hand end of certain types of statement is ignored by the interpreter *if it is enclosed in brackets.*

This is especially helpful in conditional jumps, where you can insert little mini-REMs after each condition instead of at the end of the whole statement. The following, for example, is certainly not legal in terms of what the User Instructions tell you, but on the other hand it works and is often very useful:

```
10 IF j=1 THEN 40 (initialise
variables) ELSE IF j>1 THEN 100 (main
menu)
```

Next we'll be taking a look at how variables are stored in program lines, and in particular how the Amstrad distinguishes between string and numeric variable types.

'There are a couple of curiosities that are worth noticing'

Don't just vanish without Trace!

KEITH DENHAM brings you two new helpful routines

THOSE of you who have tried to de-bug complex Basic programs using the Trace (TRON/TROFF) function will have found it less help than you might have hoped.

Because the program line numbers are printed wherever the cursor happens to be and many statements can be executed before the program pauses, it's easy to end up with a screen of legitimate output badly disrupted by the line numbers and with no idea of the order in which the latter appeared.

Worse still, on some occasions all sorts of processing goes on without pause before a CLS command removes all the Trace output. This article provides two routines which enhance the operation of this function and will also be of interest to machine code beginners.

Quite how to improve matters depends on whether or not a printer is available. If it is, the best solution is to allow the program to operate as normal and divert the trace output to the printer channel.

Program I does this, so the screen is undisrupted and you have a continuous printed record of the line numbers used. These are, of course, dispersed throughout any output normally directed to the printer but this is far less common than output to the screen and, in any case, is much easier to follow.

For those not having a printer a considerable improvement is always to print the trace output in the same place on the screen – minimising disruption of the display – and to incorporate "single stepping".

Program II does this, causing the program to stop before performing each new line, printing the line number at the bottom left of WINDOW 7, and waiting for the Tab key to be pressed before carrying out that line and progressing to the next.

A single routine would be possible but little of the code is common to both options, so this would be much longer. And since you're unlikely to want both at once it's better to have separate routines using the least memory space.

If you have an assembler program the machine code can be created using Listings I and II and saved using the assembler tape commands. Alternatively, Programs I and II will create and save the code from Basic.

Before the routines are re-loaded at any time – using LOAD "TRACE1" or LOAD "TRACE2" – the necessary memory space should be protected from overwriting by MEMORY 42449.

Both routines locate in the same memory addresses and are set on by TRON:CALL &A5D2 and off by TROFF:CALL &A5DE. Additionally

```
RAM Assembler V.3

Pass... 2           ORG 42450

A5D2:               .enable
A5D2:3E C3          LD A,&C3
A5D4:32 5A BB       LD (&BB5A),A
A5D7:21 E4 A5       LD HL,divert
A5DA:22 5B BB       LD (&BB5B),HL
A5DD:C9             RET
A5DE:               .disable
A5DE:F5             PUSH AF
A5DF:CD 37 BD       CALL &BD37
A5E2:F1             POP AF
A5E3:C9             RET
A5E4:               .divert
A5E4:E5             PUSH HL
A5E5:D5             PUSH DE
A5E6:C5             PUSH BC
A5E7:47             LD B,A
A5E8:21 1D A6       LD HL,store
A5EB:7E             LD A,(HL)
A5EC:FE 00          CP  0
A5EE:20 14          JR NZ,printer
A5F0:78             LD A,B
A5F1:FE 5B          CP  91
A5F3:28 0B          JR Z,flagon
A5F5:CD DE A5       CALL disable
A5F8:CD 5A BB       CALL &BB5A
A5FB:CD D2 A5       CALL enable
A5FE:18 19          JR reset
A600:               .flagon
A600:21 1D A6       LD HL,store
A603:34             INC (HL)
A604:               .printer
A604:78             LD A,B
A605:               .busy
A605:CD 2E BD       CALL &BD2E
A608:38 FB          JR C,busy
A60A:CD 2B BD       CALL &BD2B
A60D:78             LD A,B
A60E:FE 5D          CP  93
A610:20 07          JR NZ,reset
A612:3E 0A          LD A,&0A
A614:CD 2B BD       CALL &BD2B
A617:36 00          LD (HL),0
A619:               .reset
A619:C1             POP BC
A61A:D1             POP DE
A61B:E1             POP HL
A61C:C9             RET
A61D:               .store
A61D:               DEFS 1
A61E:               END
```

Listing I

for the second routine the instruction KEY DEF 68,0,0 will prevent the keyboard buffer being filled with the right arrow sign usually produced by the Tab key.

When either routine is being used the program involved MUST avoid the [or] brackets in screen output and when using the second routine the best results are obtained if WINDOW 7 is left defined as the whole screen and not used for any other output.

For the technically minded, the assembly listings work as follows. First, the initial call places a new address in the PRINT_CHAR "jumpblock".

This means that when the system expects to print to the screen using the printing routine in the ROM it is directed to our routine instead and we can decide what to do with the print instruction.

We look for the [bracket which opens any trace print. When this is found that bracket and following characters are sent to the printer (Program I) or the revised screen location at the bottom of WINDOW 7 (Program II) until the next] bracket.

Having intercepted the Print routine, if we actually want to print something to the screen we temporarily reset the jumpblock back to normal to avoid going round and round our own routine and disappearing up a blind alley!

```
RAW Assembler V.3

Pass... 2                ORG 42450

A5D2:                    .enable
A5D2:3E C3               LD A,&C3
A5D4:32 5A BB            LD (&BB5A),A
A5D7:21 E4 A5            LD HL,divert
A5DA:22 5B BB            LD (&BB5B),HL
A5DD:C9                  RET
A5DE:                    .disable
A5DE:F5                  PUSH AF
A5DF:CD 37 BD            CALL &BD37
A5E2:F1                  POP AF
A5E3:C9                  RET
A5E4:                    .divert
A5E4:E5                  PUSH HL
A5E5:D5                  PUSH DE
A5E6:C5                  PUSH BC
A5E7:CD DE A5            CALL disable
A5EA:47                  LD B,A
A5EB:21 48 A6            LD HL,store
A5EE:7E                  LD A,(HL)
A5EF:FE 00               CP  0
A5F1:28 23               JR NZ,trace
A5F3:78                  LD A,B
A5F4:FE 5B               CP  91
A5F6:28 05               JR Z,flagon
A5F8:CD 5A BB            CALL &BB5A
A5FB:18 44               JR reset
A5FD:                    .flagon
A5FD:21 48 A6            LD HL,store
A600:34                  INC (HL)
A601:CD 78 BB            CALL &BB78
A604:22 49 A6            LD (stor1),HL
A607:3E 07               LD A,7
A609:CD B4 BB            CALL &BBB4
A60C:32 4B A6            LD (stor2),A
A60F:26 01               LD H,1
A611:2E 19               LD L,25
A613:CD 75 BB            CALL &BB75
A616:                    .trace
A616:78                  LD A,B
A617:CD 5A BB            CALL &BB5A
A61A:78                  LD A,B
A61B:FE 5D               CP  &5D
A61D:20 22               JR NZ,reset
A61F:3A 4B A6            LD A,(stor2)
A622:CD B4 BB            CALL &BBB4
A625:2A 49 A6            LD HL,(stor1
A628:CD 75 BB            CALL &BB75
A62B:21 48 A6            LD HL,store
A62E:36 00               LD (HL),0
A630:06 64               LD B,100
A632:                    .loop
A632:C5                  PUSH BC
A633:06 00               LD B,0
A635:                    .loop1
A635:10 FE               DJNZ loop1
A637:C1                  POP BC
A638:10 F8               DJNZ loop
A63A:                    .key
A63A:3E 44               LD A,68
A63C:CD 1E BB            CALL &BB1E
A63F:28 F9               JR Z,key
A641:                    .reset
A641:CD D2 A5            CALL enable
A644:C1                  POP BC
A645:D1                  POP DE
A646:E1                  POP HL
A647:C9                  RET
A648:                    .store
A648:                    DEFS 1
A649:                    .stor1
A649:                    DEFS 2
A64B:                    .stor2
A64B:                    DEFS 1
A64C:                    END
```

Listing II

```
10 REM PROGRAM I
15 REM TRACE TO PRINTER
20 FOR x=&A5D2 TO &A61D:READ a$:POKE
x,VAL("&"+a$):NEXT
30 DATA 3e,c3,32,5a,bb,21,e4,a5,22,5b
,bb,c9,f5,cd,37,bd,f1,c9
40 DATA e5,d5,c5,cd,47,21,1d,a6,7e,fe,00
,20,14,78,fe,5b,28,0b,cd
50 DATA de,a5,cd,5a,bb,cd,d2,a5,18,19
,21,1d,a6,34,78,cd,2e,bd
60 DATA 38,fb,cd,2b,bd,78,fe,5d,20,07
,3e,0a,cd,2b,bd,36,00,c1,d1,e1,c9,00
70 REM example of routine 1
80 TRON:CALL &A5D2
90 CLS:PRINT"Testing 1,2,3,4"
100 PRINT"This is line 100"
110 TROFF:CALL &A5DE
```

Program I

```
10 REM PROGRAM II
15 REM TRACE TO WINDOW#7
20 FOR x=&A5D2 TO &A64B:READ a$:POKE
x,VAL("&"+a$):NEXT
30 DATA 3e,c3,32,5a,bb,21,e4,a5,22,5b
,bb,c9,f5,cd,37,bd,f1,c9
40 DATA e5,d5,c5,cd,de,a5,47,21,48,a6
,7e,fe,00,20,23,78,fe,5b,28,05
50 DATA cd,5a,bb,18,44,21,48,a6,34,cd
,78,bb,22,49,a6,3e,07,cd,b4,bb
60 DATA 32,4b,a6,26,01,2e,19,cd,75,bb
,78,cd,5a,bb,78,fe,5d,20,22,3a,4b,a6
70 DATA cd,b4,bb,2a,49,a6,cd,75,bb,21
,48,a6,36,00,06,64,c5,06,00,10,fe
80 DATA c1,10,f8,3e,44,cd,1e,bb,28,f9
,cd,d2,a5,c1,d1,e1,c9,00,00,00,00
90 REM example of routine 2
100 CLS:PRINT"Remember to use TAB key
 to single step, or hold down and it
will repeat!
110 TRON:KEY DEF 68,0,0:CALL &A5D2
120 PRINT"Testing 1,2,3,4
130 PRINT"This is line 130
140 GOTO 160
150 REM this line skipped
160 STOP
```

Program II

SPECIAL reader offer!

-and it's only £24.95!

IT'S the most imaginative add-on yet developed for the Amstrad – a highly sophisticated light pen that allows you to use the machine's superb graphics to the full.

And the first models to come off the production line are being offered exclusively to readers of Computing with the Amstrad!

We're making them available to readers at an unbeatable £24.95 – and that includes packing and despatching to your door by recorded delivery.

The complete kit consists of the light pen, Amstrad interface and a graphics package that will soon have you designing your own screen masterpieces.

The interface, which plugs into the expansion port on the back of the Amstrad, is compatable with both the CPC464 and CPC664 and the disc drive unit.

The light pen can be used with your own Basic or machine code programs quite easily. Several example programs in the manual demonstate its use.

The powerful graphics package supplied with the light pen will allow you to create colourful pictures which can be saved to tape or disc. It's entirely menu driven using colourful icons, so even the youngest child can use it.

The superb software allows you to draw in the sort of fine detail that is not normally possible with a light pen. By magnifying a section of the screen or using a combination of light pen and cursor keys single pixels can be plotted.

There's a choice of 10 colours, four pen sizes or airbrush. Rubber banding is also catered for and text can be written horizontally or vertically.

All in all, it makes it the most versatile package ever offered at the price.

Send for it NOW – use the official order form on Page 101

Learning to spell can be fun when you load up the...

YOUNGSTERS just learning to read will have great fun with this educational game. Without realising it, they'll be improving their shape recognition and spelling while pretending to be a crane driver loading lorries.

A word is displayed at the bottom of the screen and just above there is a train loaded with all the letters of the alphabet. At the top of the screen is a crane and on the right is an empty lorry.

The object of the game is to use the crane to pick the letters of the word, in the right order, off the train and to transfer them to the waiting lorry.

When the last letter has been placed on the lorry it drives off. The old word is then replaced by a new one, and the lorry reverses on to the screen after dumping its load.

The crane controls are very simple, so even toddlers can play. The train can be moved left and right using the left and right cursor keys and the letter directly below the crane can be picked up by pressing the cursor down key. If you have a joystick then this can be used instead.

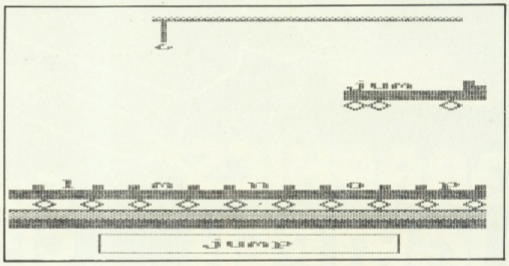
The most important part of the program is the subroutine to select a word and the list of words themselves. This is at the end of the program. The first data statement contains the number of words. You can have as many as you want, just place the number there.

The rest of the data statements contain the words. These can be any words up to five letters long. If they are longer they won't fit on the back of the truck. If you don't like our selection simply type in your own.

There is no need to have lower case letters if you don't want them. Line 410 initialises the string. As you can see from the listing, characters 97 to 122 are used. These are the lower case alphabetic characters.

In fact any 26 consecutive characters could be substituted. You could even define your own.

If you are typing the program into a CPC664 you'll need to increase the volume parameter of the sound commands or it will be too quiet.



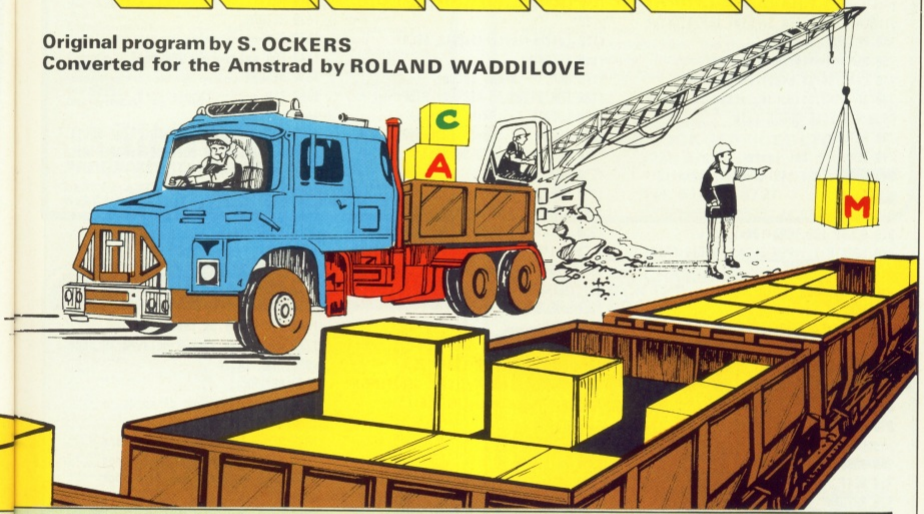
VARIABLES

p	Part of alphabet to be displayed.
a$	Alphabet string.
wagon$	Train.
word$	Word chosen.

```
10 REM Alphabet Train
20 REM By S.Ockers
30 REM Converted For The Amstrad
40 REM By R.A.Waddilove
50 MODE 0
60 GOSUB 340:REM initialise
70 GOSUB 1340:REM titles
80 GOSUB 200:REM screen
90 WHILE NOT fed.up
100 GOSUB 1660:REM choose word
110 GOSUB 530:REM lorry left
120 WHILE letter<LEN(word$)+1
130 IF (INKEY(1)>-1 OR INKEY(75)>-1)
AND p>2 THEN p=p-2:GOSUB 470:p=p-2:GO
SUB 470
140 IF (INKEY(8)>-1 OR INKEY(74)>-1)
AND p<102 THEN p=p+2:GOSUB 470:p=p+2:
GOSUB 470
150 IF INKEY(2)>-1 OR INKEY(73)>-1 TH
EN GOSUB 750
160 WEND
170 GOSUB 610:REM move lorry
180 WEND
190 END
200 REM ----- Screen -----
210 MODE 0:INK 1,0:INK 2,2:INK 3,6:IN
K 4,6:INK 5,6:INK 6,8:INK 7,18
220 INK 1,0:INK 2,2:INK 3,6:INK 4,6:I
NK 5,6:INK 6,8:INK 7,18
230 p=6:f=-1
240 LOCATE 7,1:PEN 3:PRINT CHR$(22)+C
HR$(1);STRING$(13,218):LOCATE 7,1:PRI
NT STRING$(13,210);CHR$(22);CHR$(0);
250 LOCATE 7,2:PRINT CHR$(149):LOCATE
 7,3:PRINT CHR$(149):LOCATE 7,4:PRINT
 CHR$(174)
260 LOCATE 1,18:PAPER 0:PEN 7:PRINT M
ID$(a$,p,20);
270 FOR i=1 TO 5:PRINT USING "&";wago
n$;:NEXT
280 PAPER 0:PEN 6
290 FOR i=1 TO 5:PRINT " ";CHR$(202);
" ";CHR$(202);:NEXT
300 PEN 10:PRINT CHR$(22);CHR$(1);STR
ING$(20,208);CHR$(11);:PAPER 8:PEN 9:
```

ALPHABET TRAIN

Original program by S. OCKERS
Converted for the Amstrad by ROLAND WADDILOVE




```
PRINT STRING$(20,207);CHR$(22);CHR$(0
);STRING$(20,207);
310 MOVE 120,8:DRAW 120,38,15:DRAW 52
0,38:DRAW 520,8:DRAW 120,8
320 RETURN
330 REM ----------------------------
340 REM Initialise
350 DEFINT a-z
360 ENT 1,50,-10,1,50,10,2
370 ENT -2,10,-1,1,10,1,1
380 c$=" "+STRING$(6,143):d$=" "+CHR$
(202)+CHR$(202)+"  "+CHR$(202)+" "
390 INK 15,26:INK 14,24:INK 11,20:INK
 12,15
400 a$=SPACE$(128):a$=SPACE$(5)
410 FOR i=97 TO 122:a$=a$+" "+CHR$(13
2)+CHR$(i)+CHR$(136):NEXT
420 a$=a$+SPACE$(5)
430 DATA 14,1,15,2,198,15,3,143,14,4,
15,5,198,15,3,143
440 wagon$="":RESTORE 430:FOR i=1 TO
16:READ j:wagon$=wagon$+CHR$(j):NEXT
450 RETURN
460 REM ----------------------------
470 REM Move train
480 LOCATE 1,18:PEN 7
490 SOUND 1,1000,15,5,0,0,15:GOSUB 16
30
500 IF f THEN CALL &BD19:INK 6,4:INK
1,6:INK 2,6:INK 4,0:INK 5,2:PRINT MID
$(a$,p,20) ELSE CALL &BD19:INK 6,8:IN
K 1,0:INK 2,2:INK 4,6:INK 5,6:PRINT M
ID$(a$,p,20)
510 f=NOT f
520 RETURN
530 REM --- Move lorry left ---
540 b$=SPACE$(6)+CHR$(141)
550 SOUND 129,1000,5000,4,0,2:SOUND 1
30,800,5000,4,0,2
560 FOR i=1 TO 7
570 GOSUB 710
580 NEXT
590 CALL &BCA7
600 RETURN
610 REM --- Lorry right ---
```

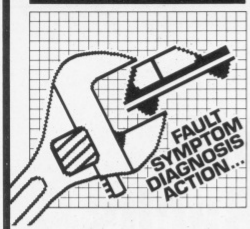
```
620 b$=" "+word$+MID$(b$,LEN(word$)+2
)
630 FOR i=1 TO 2:SOUND 1,1000,100,4,0
,1:SOUND 2,2000,100,4,0,1:NEXT
640 FOR delay=0 TO 2000:NEXT
650 FOR i=7 TO 1 STEP -1
660 SOUND 129,1000,200,4,0,2:SOUND 13
0,800,200,4,0,2
670 GOSUB 710
680 NEXT
690 CALL &BCA7
700 RETURN
710 REM move
720 j=21-i:FOR delay=1 TO 500:NEXT
730 LOCATE j,8:PEN 11:PRINT LEFT$(b$,
i):LOCATE j,9:PEN 12:PRINT LEFT$(c$,i
):LOCATE j,10:PEN 3:PRINT LEFT$(d$,i)
740 RETURN
750 REM ---- letter -----
760 L$=MID$(a$,p+6,1):GOSUB 790
770 IF L$=MID$(word$,letter,1) THEN G
OSUB 930 ELSE GOSUB 1170
780 RETURN
790 REM ---- Get letter -----
800 hook$=CHR$(149)+CHR$(8)+CHR$(10)+
CHR$(174)
810 FOR i=4 TO 16
820 SOUND 132,1000+100*i,25,6
830 LOCATE 7,i:PEN 3:CALL &BD19:PRINT
 hook$
840 GOSUB 1630
850 NEXT
860 hook$=CHR$(174)+CHR$(8)+CHR$(10)+
CHR$(15)+CHR$(7)+L$+CHR$(8)+CHR$(10)+
" "
870 FOR i=16 TO 4 STEP -1
880 SOUND 132,1000+100*i,25,6
890 LOCATE 7,i:PEN 3:CALL &BD19:PRINT
 hook$
900 GOSUB 1630
910 NEXT
920 RETURN
930 REM --- Right letter ---
940 FOR i=6 TO 13+letter
950 SOUND 129,800,5000,4,0,2
960 LOCATE i,2:PEN 3:CALL &BD19:PRINT
 " ":CHR$(149):LOCATE i,3:PRINT " ";C
HR$(149):LOCATE i,4:PRINT " ";CHR$(17
4):LOCATE i,5:PEN 7:PRINT " ";L$
970 GOSUB 1630
980 NEXT
990 FOR i=4 TO 6
1000 SOUND 132,1000+100*i,25,6
1010 LOCATE 14+letter,i:PEN 3:CALL &B
D19:PRINT CHR$(149);CHR$(8);CHR$(10);
CHR$(174);CHR$(8);CHR$(10);CHR$(15);C
HR$(7);L$
1020 GOSUB 1630
1030 NEXT
1040 FOR i=6 TO 4 STEP -1
1050 SOUND 132,1000+100*i,25,6
1060 LOCATE 14+letter,i:PEN 3:CALL &B
D19:PRINT CHR$(174);CHR$(8);CHR$(10);
" "
1070 GOSUB 1630
1080 NEXT
1090 FOR i=13+letter TO 7 STEP -1
1100 SOUND 129,800,5000,4,0,2
1110 LOCATE i,2:PEN 3:CALL &BD19:PRIN
T CHR$(149);" ":LOCATE i,3:PRINT CHR$
(149);" ":LOCATE i,4:PRINT CHR$(174);
" "
1120 GOSUB 1630
1130 NEXT
1140 LOCATE 1,18:PEN 7:PRINT MID$(a$,
p,20)
1150 letter=letter+1:CALL &BCA7
1160 RETURN
1170 REM --- Wrong letter ---
1180 FOR i=200 TO ASC(L$) STEP -1
1190 LOCATE 7,5:PRINT CHR$(i)
1200 SOUND 132,500+INT(RND*500),20,4
1210 FOR delay=1 TO 50:NEXT
1220 NEXT
1230 FOR i=4 TO 16
1240 SOUND 132,1000+100*i,25,6
1250 LOCATE 7,i:PEN 3:CALL &BD19:PRIN
T CHR$(149);CHR$(8);CHR$(10);CHR$(174
);CHR$(8);CHR$(10);CHR$(15);CHR$(7);L
$
1260 GOSUB 1630
1270 NEXT
1280 FOR i=16 TO 4 STEP -1
1290 SOUND 132,1000+100*i,25,6
1300 LOCATE 7,i:PEN 3:CALL &BD19:PRIN
T CHR$(174);CHR$(8);CHR$(10);" "
1310 GOSUB 1630
1320 NEXT
1330 RETURN
1340 REM ----- Titles -----
1350 BORDER 0:INK 0,0:INK 1,24:INK 2,
26:INK 3,18:INK 4,20:INK 5,2
1360 RESTORE 1380
1370 MOVE 30,200:DRAW 30,398,4:DRAW 5
80,398:DRAW 580,200:DRAW 30,200
1380 DATA HFFAHFFFHFHFHBLB
1390 DATA NFFANBNFNFPAHAKA
1400 DATA FFNEFAFFFFNFNEKA
1410 RESTORE 1380
1420 PEN 2:x=3:y=3:GOSUB 1540
1430 DATA LBHFHFFNK
1440 DATA KAPANFFHO
1450 DATA KAFFFFFFL
1460 PEN 1:x=6:y=8:GOSUB 1540
1470 PEN 3:LOCATE 7,18:PRINT "k e y s
"
1480 PEN 5:LOCATE 8,22:PRINT CHR$(242
);"  ";CHR$(243)
1490 LOCATE 10,23:PRINT CHR$(241)
1500 WHILE INKEY$<>"":WEND
1510 SOUND 1,239,300,2:SOUND 2,239,10
0,3:SOUND 4,239,200,4:SOUND  2,95,200
,3:SOUND 4,80,100,4
1520 WHILE INKEY$="":WEND
1530 RETURN
1540 REM --- Print title ---
1550 FOR i=1 TO 3
1560 LOCATE x,y+i-1
1570 READ title$
1580 FOR j=1 TO LEN(title$)
1590 PRINT CHR$(ASC(MID$(title$,j,1))
+63);
1600 NEXT
1610 NEXT
1620 RETURN
1630 REM ----- delay -----
1640 FOR delay=1 TO 200:NEXT
1650 RETURN
1660 REM ------ choose word ------
1670 letter=1
1680 PAPER 10:LOCATE 5,24:PRINT SPC(1
2):FOR delay=0 TO 1000:NEXT:PAPER 0:L
OCATE 5,24:PRINT SPC(12)
1690 RESTORE 1800:READ number
1700 FOR i=0 TO INT(RND*number)
1710 READ word$
1720 NEXT
1730 PAPER 0:PEN 14
1740 LOCATE 11-LEN(word$)\2,24
1750 PRINT word$;
1760 SOUND 129,50,100,4
1770 INK 14,24,12:FOR delay=0 TO 5000
:NEXT:INK 14,24
1780 RETURN
1790 REM ---- Number of words ----
1800 DATA  35
1810 REM - Words up to 5 letters -
1820 DATA cat,dog,cow,sheep,frog
1830 DATA tree,mat,cup,bell,book
1840 DATA key,horse,fish,sky,rain
1850 DATA sun,wind,air,jump,quack
1860 DATA zebra,up,down,left,right
1870 DATA hand,foot,head,ear,eye
1880 DATA nose,mouth,leg,arm,bird
```

Give your fingers a rest . . .

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

See our special offer on Page 101.

CAR CURE



CAR CURE will transform your micro into an **expert mechanic** to help you **find faults** on your car and then **determine the cure.** Technical Colleges buy **CAR CURE** to help students **learn** about the faults cars suffer, as well as their symptoms and cures. Now you too can benefit!

Complete with comprehensive **5000-word User Guide** and packaged in a long-life plastic wallet. Export orders get American-English cross reference. All for less than many garages charge for an hour's work!

Cassette for **Amstrad CPC464** and **48K Spectrum.** *(microdrive transfer instructions included free, on request)* at only £9.95.

Disc for **BBCB** *(state 40 or 80 track)* at £11.95

★ **REVIEWS**

"... works extremely well, a great deal of effort has gone into making it user friendly as well as functional ... a very worthwhile package"
–CRASH

"... an expert system ... incredibly easy to use"
–Popular Computing Weekly

"... well produced ... a worthwhile addition to your toolkit"
–ZX Computing

"... one of the most useful programs I have ever used. Makes 'xyz' look very childish and useless. A software company to keep an eye on"
–ZX Appeal (Canada)

★ **COMPREHENSIVE**
+ 900 faults
+ 300 symptoms

★ **POWERFUL**
+ Discover faulty parts for every symptom
+ Display faults for every part
+ Recommended cure for every fault
+ Trace symptoms of any fault
+ Identify faults causing two symptoms
+ Mark items for easy reference
+ Eliminate faults checked OK
+ Review progress on summary screen
+ Call help if in difficulty
+ Save diagnosis in data file

SIMTRON
PROGRAMS TO HELP YOU

DEPT CA, 4 CLARENCE DRIVE, EAST GRINSTEAD, WEST SUSSEX RH19 4RX **Telephone (0342) 28188**

UNLOCK YOUR
AMSTRAD 464/664/6128

AMSKEY ● Tape to tape backup utility **(Must not be used to infringe copyright laws)**

Multi colour display and so very user friendly.
Gives full header details and choice of saving speed.
Full on screen instructipns and prompts.
FREE Peep a code program, for finding the hidden words in adventure games, etc.

Tape £6.99

AMSHELP ● Transfer tape software to disc

Better than most, AMSHELP will even transfer ASCII files. Also automatically adds a relocator for those programs that overwrite the disc memory area.
FREE Disc utility load and list protected programs etc.

Tape £6.99 Disc £10.99

AMSDISC1 ● The most powerful Amstrad disc utility to date

Powerful CPM style utility with many features making easy work of disc file handling, including the unerasing of erased files. "No disc user should be without this utility" (Independent Amstrad review).
FREE Disc utility load and list protected programs etc.

Disc £10.99

Prices include packing and postage
Overseas orders please add £1

INTERLOCK SERVICES LTD
123b Brent Street
London NW4
Tel: 01-202 8727
Personal callers welcome

ACCESS VISA

SHEKHANA COMPUTER SERVICES

	RRP	Our Price
Way of Exploding Fist	9.95	7.50
Rocky Horror Show	8.95	6.75
Every Ones a Wally	9.95	7.50
Marsport	9.95	7.50
Frank Brunos Boxing	8.95	6.75
Dark Star	7.95	6.40
Jump Jet	9.95	7.95
Raid (Over Moscow)	9.95	7.50
B. Jacks Challenge	8.95	7.50
Boulder Dash	9.95	7.50
Fighter Pilot	8.95	6.95
Jet Set Willy	8.95	6.75
Dragontorc	8.95	6.95
The Hobbit	14.95	11.95
Handicap Golf	6.95	5.95
3D Voice Chess	12.95	10.95
Robin of Sherwood	9.95	7.95
Beachhead	9.95	7.35
Rocco (Rocky)	8.95	7.50
Covenent	6.95	5.50
Project Future	8.95	7.50
Sourcery	8.95	6.95
Alien 8	9.95	7.50
Airwolf	8.95	7.50
Arnehem	9.95	7.95
Dun Darach	9.95	7.50
Hard Hat Mac	8.95	7.50
Archon	10.95	9.99
Bridge Player (Kuma)	6.95	5.95
*Shadowfire	9.95	7.50
3D Star Strike	6.95	5.50
Hypersport	8.95	6.75
Emerald Isle	6.95	5.75
Mini Office	5.95	4.70
Scrabble	9.95	7.95
Monopoly	9.95	7.95
*Lords of Midnight	8.95	7.45
Sorcery + (Disk)	13.95	12.50
Hurricane Higgins Snooker	8.95	7.50
H. Higgs Snooker (disk)	12.95	11.95
Fighter Pilot (disk)	12.95	11.95
Roland in Space (disk)	12.95	11.95
Beachhead (disk)	13.95	12.50
Master Chess	8.95	6.75
*F.G.T. Hollywood	9.95	7.50
International Karate	6.50	4.99
Daley Toms Decathalon	8.95	6.75
Scooby Doo	8.95	6.75
Steve Davis Snooker	7.95	6.45
*Spy v Spy	9.95	7.50
Wizards Lair	8.95	7.50
Combat Lynx	8.95	6.95
Grand Prix Rally 2	8.95	6.95
*Glen Hoddle Football	8.95	7.50
Master of the Lamp	9.95	7.95
Gremlins	9.95	7.50
Arabian Knights	8.95	7.50
Azimuth (Head Aligment)	8.95	7.50
Hard Hat Mac	8.95	7.50
Warlords	8.95	7.50
On The Run	7.95	6.45
One On One	8.95	7.50
Starion	9.95	7.50
Superpipeline II	8.95	7.50
Ian Bothams Cricket	9.95	7.50
Realm of Impossibility	8.95	7.50
Red Moon	8.95	5.95
Android One	7.95	5.95
Android Two	7.95	5.95
Tornado Low Level	7.95	5.95
Highway Encounter	7.95	5.95
Forth (Interceptor)	12.99	9.99
International Basketball	5.99	4.99
Slap Shot	8.95	7.50
3D Stunt Rider	8.95	7.50
Knight Lore	9.95	7.50
Red Arrows	9.95	7.50
Five A Side Football	8.95	7.50
*Yie Ar Kung Fu	8.95	6.75
Battle for Midway	9.95	8.00
Tank Busters	7.95	6.50
Code Name Mat II	8.95	6.75
Killer/Gorilla	9.95	7.99
Animated Strip Poker	8.95	7.25
Any Level 9 Game	9.99	7.50
Roland in Space	9.95	6.95
*Matchday	9.95	7.50
*World Series Baseball	8.95	6.75
Quickshot II Joystick	12.95	8.99
The Neverending Story	9.95	7.50

Cheques/P.O. To: **SCS (DEPT CWA11), P.O. BOX 394 LONDON N.15 6JL. (MAIL ORDER ADDRESS ONLY). Tel: 01-800 3156 (S.A.E. FOR LIST)**

Please bring copy of advert to our shop for above discounts to Marbles Shopping Centre Unit 11. 527-531 Oxford St. W1R 1DD. (1 min. from Marble Arch Tube Station). Open 7 days a week from 10am-7pm. Sunday 10am-6pm.

* Please ring to confirm availability.
Barclaycard/Access orders welcome

VISA ACCESS

ADVANCED PROGRAMMING TECHNIQUES ON THE AMSTRAD CPC 464

FREE WITH EVERY ORDER – AN AMSTRAD CRIB CARD

"bridging the chasm between BASIC & MACHINE CODE programming is totally achieved in this book.

Many subtle and complex insights are revealed ... gratifying to find lucid explanation of tailoring MACHINE CODE programs into a BASIC program.

Exceptionally clear type-face ...

Sensibly priced"

"HOME COMPUTING WEEKLY"

CHEQUE OR POSTAL ORDER FOR £8.50 (inc p & p) TO

PHOENIX PUBLISHING ASSOCIATES, 14 VERNON ROAD, BUSHEY, HERTS, WD2 2JL. Tel WATFORD 0923 32109

NAME .....

ADDRESS .....

.....

AP1 ..... **CODE** .....

NIGEL PETERS brings his series on Amstrad sound to a grand finale with an examination of the SQ function

SQ will help budding Bachs get their cantatas together

IT seems a long time since we started out together on this exploration of the Amstrad's Basic sound commands. In the past few months we've covered the channel, pitch, duration and volume parameters and licked both volume and pitch envelopes.

After all that we had another look at the channel parameter and saw how it was bit significant, learning how to hold, release and rendezvous notes on different channels.

And now, in this final article, we'll be taking a look at the SQ function.

At first sight the SQ command looks like it should be a mathematical function, squaring a number. However, what it does is test the state of the Sound Queue, hence SQ. If you're not too sure about what a channel's sound queue is, take a look at the articles in the July and August issues of *Computing with the Amstrad*.

SQ is used to gather information about the state of a channel. It can tell you if a note is playing – not always so obvious if you're using three channels at once – how many free places there are on the queue and if the note at the front of the queue has been held or rendezvoused with another note.

This is extremely useful when you're trying to figure out why your masterpiece doesn't work like it was supposed to.

When you want to know about a channel you just use SQ followed by that channel's parameter in brackets. So if you wanted to know about Channel A you'd use:

```
PRINT SQ(1)
```

and, of course, press the Enter key. Provided that you haven't got a note playing, you should get the figure 4 returned. You'll find that:

```
PRINT SQ(2)
```

and:

```
PRINT SQ(4)
```

will give the same result. Well, it's consistent, but what does it mean? The answer is that it means:

- That there's no note playing.
- That there's no note held at the start of that channel's sound queue.
- The leading note in the queue, if any, is not rendezvoused with a note on any other channel.
- There are four free places in the sound queue.

That's a lot of information to get from one number, isn't it? The reason a single figure can throw so much light on the state of the channel queue is because the number returned from SQ is bit significant. Table I shows the significance of each bit.

number	bit	if set it shows:
1	0	spaces in queue
2	1	spaces in queue
4	2	spaces in queue
8	3	rendezvous with A
16	4	rendezvous with B
32	5	rendezvous with C
64	6	first note held
128	7	channel playing

Table I: Bit significant values of SQ

So, let's have a bit of a look at the significance of the 4 we get from:

```
PRINT SQ(1)
```

when there's no note playing on channel A. However, since we're dealing with bit significance it makes sense to use BIN$ to put the number into 8 bit binary form. So, with no note playing on channel A when we use:

```
PRINT BIN$(SQ(1),8)
```

we should get:

```
00000100
```

which is the binary version of decimal 4, returned. Figure I shows decimal 4 as a binary number along with the numbers of the bit positions.

bit number	7	6	5	4	3	2	1	0
binary byte	0	0	0	0	0	1	0	0

Figure I: Bit positions of binary 4

Now by combining the data in Figure I with the information in Table I we should be able to see how we learnt so much about the state of channel A from a single 4.

Let's look at bit 7. This is a 0, so we know that there is no note playing. If there was the binary bit would be set and the decimal value would be at least 128, and maybe more depending on the state of the other bits in the byte.

Bit 6 is also 0, and this shows that the front note on the queue is not held. If that bit was set it would mean that the first note on the queue is

waiting for a RELEASE command before it sounds.

There are still more 0s in bit positions 5, 4 and 3. These show that the first note in the sound queue, always supposing there is one, is not rendezvoused with any other.

If bit 5 was set then it would mean that the front note on that channel was rendezvoused with one on channel C. Bit 4 being set means that the note is rendezvoused with one on channel B.

As you might guess, a 1 in bit 3 show that the leading note is waiting for one on channel A.

The final three bits are just used to show the number of three places in the sound queue. In this case bits 2, 1 and 0 hold the binary number 100, which is decimal 4. This shows, as you might expect since there is no note playing, that there are four places free in that particular channel's sound queue.

If the last three bits are 011 it shows that there are 3 places free in the queue, if they are 010, 2. I leave it to you to figure out how many unused places there are when the last three bits are 001 and 000.

As you can see, the figure returned by SQ already contains a powerful lot of information even though the Amstrad hasn't made a sound yet.

Now we'll move on to showing what SQ can do when the micro is actually making noises, but first you might like to set up the small Enter key with:

```
KEY 139,"SOUND 135,0,0,0"+CHR$(13)
```

Use this to clear the channels when you get stuck!

And now to making noises. Let's play a long note on channel A with:

```
SOUND 1,200,2000,7
```

and see what SQ can tell us about it.

Provided you do it while the note is playing, you'll find that:

```
PRINT SQ(1)
```

will give you the number 132 for your pains. I think you'll agree that the binary form gives more information. You'll find a quick:

```
PRINT BIN$(SQ(1),8)
```

will do the job, returning the binary value:

```
10000100
```

Figure II shows this number along with its bit numbers.

bit number	7	6	5	4	3	2	1	0
binary byte	1	0	0	0	0	1	0	0

Figure II: Bit positions of binary 132

From this and Table I you should be able to see that since bit 7 is set the channel is active. Also as bits 2, 1 and 0 make the binary number 100, it should come as no surprise that there are four spaces left in the sound queue.

Now let's have two notes in succession on channel A and see what SQ can tell us about them. Enter:

```
SOUND 1,200,2000,7
```

followed by:

```
SOUND 1,300,1000,7
```

which gives us two long notes, the second one lower in pitch.

While the first note is playing:

```
PRINT SQ(1)
```

gives us the figure 131. Using:

```
PRINT BIN$(SQ(1),8)
```

gives the more useful binary form of 131, returning:

```
10000011
```

This shows that the channel is active (bit 7 is set) and that there are only three places left in the sound queue (bits 2-0). This tallies with what we know as we can hear the note and, since the second one hasn't started yet it must be on the queue. There are only three of the original four spaces left.

As soon as the second note has started playing, enter:

```
PRINT BIN$(SQ(1),8)
```

again and you'll see that:

```
10000100
```

is the result.

Again bit 7, being set to a 1, shows that the channel is still active. However now that the first note is finished and the second one started, there are four places left in the queue (bits 2-0). I leave it to you to put two, three and four notes on the queue and see what happens to the results of:

```
PRINT BIN$(SQ(1),8)
```

as the notes come off the queue and sound.

What about bits 3, 4 and 5? These show whether or not the leading note of the queue is rendezvoused with one on another channel. Clear the channel A queue by pressing the small Enter key and type in:

```
SOUND 17,200,1000,7
```

This puts a note of pitch 200,

duration 10 seconds and volume 7 on th channel A queue, making it rendezvous with one on channel B. Table II should refresh your memory about channel parameters.

number	bit set	result
1	0	uses channel A
2	1	uses channel B
4	2	uses channel C
8	3	rendezvous with A
16	4	rendezvous with B
32	5	rendezvous with C
64	6	hold until RELEASEd
128	7	flush the channel

Table II: Channel parameter values and actions

Now using:

```
PRINT BIN$(SQ(1),8)
```

produces:

```
00010011
```

Bit 7 is 0, there is no note playing. Nor is the leading note held, so bit 6 is also clear. Bit 5 is 0 as the note isn't rendezvoused with channel C. However it is rendezvoused with one on channel B, so bit 4 is set. For obvious reasons bit 3, which marks a rendezvous with a note on channel A is clear.

The binary number formed by bits 2, 1 and 0 is 011, which means there are three spare places on the queue. The fourth space is held up by our note waiting forlornly for a date with one from channel B. Put it out of its misery with:

```
SOUND 10,1,1,1
```

and let's see about putting a note on channel B to rendezvous with one on channel A. We do this with:

```
SOUND 10,200,1000,7
```

Now:

```
PRINT BIN$(SQ(2),8)
```

gives us:

```
00001011
```

Notice the 2 in the brackets after the SQ. We're dealing with channel B.

By now you should have no trouble reconciling the fact that because the note is rendezvoused with one on channel A, bit 3 is set. As before, the last three bits show that there are three places left in the queue.

```
SOUND 17,1,1,1
```

provides a suitable partner for that note.

In order to show the significance of bit 5, enter:

```
SOUND 34,200,1000,7
```

which puts a note on the channel B queue, rendezvoused with one on channel C. Now:

```
PRINT BIN$(SQ(2),8)
```

gives:

```
00100011
```

showing that when a note is rendezvoused with one on channel C, bit 5 is set to 1.

Now let's look at bit 6. This is set when the leading note on the queue is held. Let's put a note on the channel A queue and hold it with:

```
SOUND 65,200,1000,7
```

and interrogate it with:

```
PRINT BIN$(SQ(1),8)
```

giving:

```
01000011
```

As you can see, bit 6 is set, showing that the leading note on channel A is held. If you now enter:

```
RELEASE 1
```

you'll see that:

```
PRINT BIN$(SQ(1),8)
```

returns:

```
10000100
```

while the note is playing and:

```
00000100
```

when it's finished.

We've already seen that bit 7 is set when the channel is active, that is, the note is playing. If you really want to test your understanding of SQ, try using it on channels whose leading notes are not only held but also rendezvoused. Then see what happens as you free these notes from their double restraints.

SQ isn't just used for fact-finding and debugging channel queues. It can also be used to sort out some of the drawbacks we've come across when using sound with programs. Program I shows what I mean.

As you'll see when you run the

```
10 REM Program I
20 FOR noise=1 TO 10
30 PRINT "Loop number ";noise
40 SOUND 1,10*noise,100,7
50 NEXT noise
```

program, the PRINT and SOUND commands are out of step. The numbers rush through to six, then appear one at intervals of a second. Meanwhile the notes are playing and carry on after the "Ready" message has appeared. You'll remember from before that this is a result of the sound queue getting full and holding up the Basic.

Well SQ can remedy this, by making the micro enter a delaying WHILE . . . WEND which holds up the loop while a note is playing. You do this with a line such as:

```
45 WHILE SQ(1)>=127:WEND
```

While a note is playing the WHILE . . . . WEND loop cycles uselessly, holding up the FOR . . . NEXT loop. Of course when that note finishes so does the delay loop, and the program carries on. This keeps the messages and the notes in step, but it does slow the Basic program down.

Program II shows another problem.

```
10 REM Program II
20 FOR noise=1 TO 10
30 READ pitch
40 SOUND 1,pitch,100,7
50 NEXT noise
60 DATA 100,200,300,400,500
70 DATA 600,400,300,200,100
```

Here you'll hear 10 notes, one after the other. Now suppose I wanted this wonderful melody to be playing while the micro was doing some maths? The aim might be to keep you entertained during a boring set of sums. How can I do it? If it was the two times table I wanted to accompany with my tune I might try a line like:

```
25 PRINT 2*noise
```

but it doesn't do much good. What we want is a method that has the

'You'll learn a lot more by trying the sound commands out on your micro'

Amstrad doing two things at once, playing a tune and running a Basic program. This isn't as impossible as it seems. All it needs is the ON SQ( ) GOSUB command. Program III shows how it is done.

```
10 REM Program III
20 ON SQ(1) GOSUB 70
30 WHILE NOT true
40 PRINT "The Amstrad is doing two th
ings at once!"
50 WEND
60 REM Sound producing routine
70 READ pitch
80 IF pitch=0 THEN RESTORE:GOTO 70
90 SOUND 1,pitch,100,7
100 ON SQ(1) GOSUB 70:RETURN
110 DATA 10,30,50,70,90
120 DATA 80,70,60,50,40,0
```

Here the messages are being printed out while a "tune" is being played and it's all due to the ON SQ GOSUB. Let's take a closer look at the program.

Line 20 is the first occurrence of the new command. All it does is to tell the Amstrad that when there is a free place on the channel A sound queue it is to momentarily interrupt what it is doing and go to the subroutine at the specified line number.

The figure in brackets after the SQ determines which channel is used to determine the interruption. As soon as it has performed the subroutine it goes back to where it left off.

In other words, the ON SQ GOSUB sets an interrupt. When the conditions are ripe – a space on the appropriate channel queue – the micro stops what it is doing while it goes to the specified subroutine, obeys the code it finds there, and goes back to where it left off. So long as the subroutine is short and the interruptions few and far between, the user hardly notices the interrupt.

So line 20 has set up an interrupt and, since there is nothing in the channel A sound queue at present, it immediately goes to the subroutine at line 70. This just reads a value for *pitch* from the data of lines 110 and 120 and uses the SOUND command of line 90 to play a note.

Then, while the note is playing the RETURN of line 100 sends the micro back to the main program, but only after setting up another interrupt with ON SQ GOSUB. This now waits for the next space to be free. In fact the first few times round the subroutine the interrupt will work straightaway as, at the beginning, there are four empty places to be filled. After that it has to wait until one of the notes has finished and a space appeared on the queue.

When an interrupt is over the micro goes back to the endless WHILE . . . WEND formed by lines 30 and 50. This prints out the message while a note is playing. As soon as one stops and a place becomes free on the queue the interrupt is triggered and the subroutine at line 70 is performed again, providing another note.

One important thing to notice is that as soon as an ON SQ GOSUB has sent the micro to the subroutine the interrupt is disabled. It only works once.

If you want to use it again you've got to issue another ON SQ GOSUB to set it up again. This is the reason why line 100 has an ON SQ GOSUB enabling the interrupt again. Every time the subroutine has been called the interrupt is switched off. Line 100 just makes sure that it is switched back on again before it returns to the main program.

Incidentally, if you use a SOUND command or a SQ function in your programs you will have to reset the interrupt after them, as these also disable it.

Now try leaving out line 80 of Program III and see if you can explain what happens. The micro tells you that the data is exhausted long before all the 11 *pitch* values have been used. Why?

The answer lies in line 100. This keeps on looking for more notes to put on the queue when spaces become vacant. After it's shoved 11 notes on the queue it still looks for more. Of course there aren't any now that the RESTORE of line 80 has gone, so the:

```
DATA exhausted in 70
```

message appears. The result is that the program crashes before all the notes have reached the front of the queue and been played.

And that's where I run out of data as well. I've finished sounding off, and there's nothing left in my queue except Program IV which just shows SQ and ON SQ GOSUB in action again.

```
10 REM Program IV
20 ON SQ(1) GOSUB 70
30 WHILE NOT true
40 PRINT "The Amstrad is doing two th
ings at once!"
50 WEND
60 REM Sound producing routine
70 READ pitch
80 count=count+1
90 SOUND 1,pitch,100,7
100 IF count <10 THEN ON SQ(1) GOSUB
70
110 RETURN
120 DATA 10,30,50,70,90
130 DATA 80,70,60,50,40
```

Try figuring out how that works, and then try using all we've covered to write your own music.

You'll learn a lot more by trying the sound commands out at your micro than just by reading about them. And you'll have a lot of fun, which is what computing is all about.

If you come up with anything nice, let's see it at *Computing with the Amstrad*. And on that note I bid farewell.

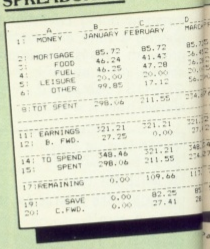
QUICK TO LEAR

THAT'S...

MINI OFFICE

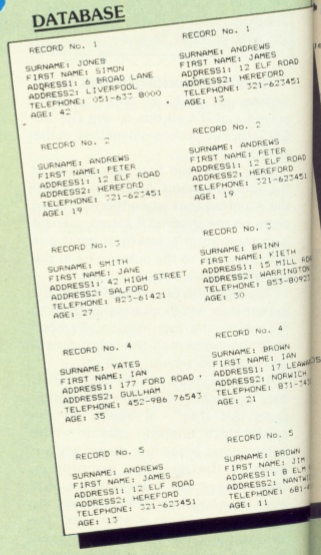
FINALIST BRITISH MICRO COMPUTING AWARDS 1985

SPREADSHEET



JUST LOOK WHAT THIS PACKAGE CAN DO!

DATABASE



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

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

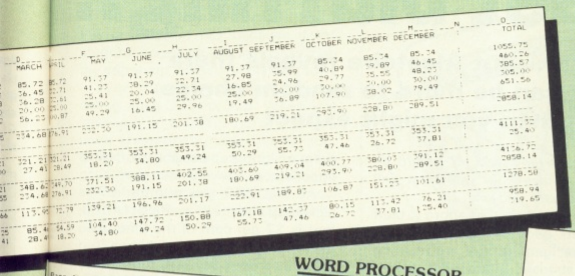
GRAPHICS – Turn those numbers into an exciting visual display! *Features:* 3D bar chart ● Pie chart ● Graph.

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

...and it's all at t price of just £

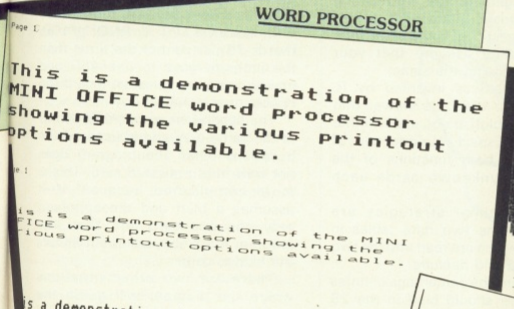
RN, EASY TO USE

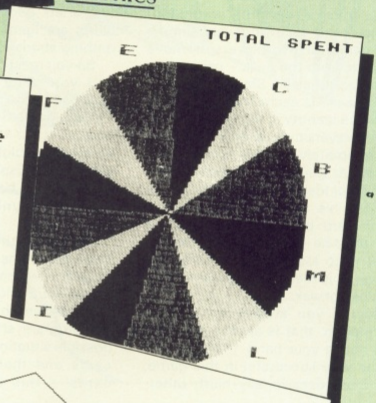
Specially written for your AMSTRAD CPC

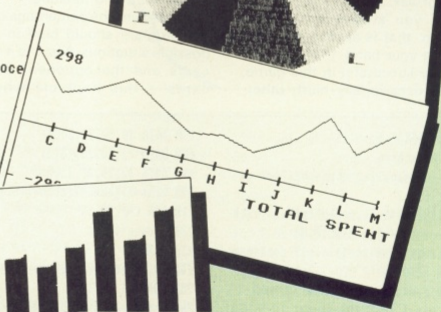


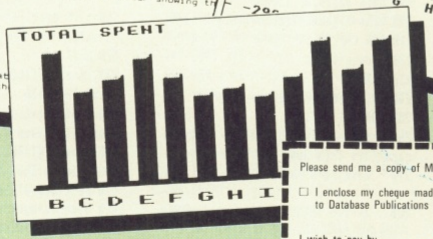
GRAPHICS

WORD PROCESSOR









at the unbelievable

st £5.95
CASSETTE
3" DISC £9.95

DATABASE SOFTWARE

Please send me a copy of Mini Office

☐ I enclose my cheque made payable to Database Publications Ltd.

Amstrad 464 Tape	£5.95
Amstrad 464 3" Disc	£9.95
Amstrad 664 3" Disc	£9.95

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.

CA11

Just bluffing..

. . . but Aleatoire reveals there can be more to random choice than you might expect

MANY books have been written about bluff and some attempts have been made to treat the subject mathematically – this approach comes under the General Theory of Games and reveals a number of apparent paradoxes.

For example, given limited time or resources, it is often more effective to confuse your opponent rather than try to improve your own position, and that the *best* way to confuse him is to make random decisions.

To give a working example, consider the following very simple card game for two players. Remove 11 cards from the pack – usually the ace to jack of a suit. Shuffle them and deal five cards each. The object is to guess the identity of the undealt card.

The players pick up their cards and play alternately. At each turn a player can either guess the hidden card – winning or losing immediately – or he can ask his opponent if he has a certain card. If the opponent has that card then he must tell the truth. To cheat is to lose. Irrespective of the answer, it is then the opponent's turn to guess or ask.

Now if you always try to gain information, that is always ask for a card not in your hand, then you will not be very successful in this game. You must occasionally bluff, otherwise whenever asked for a card he does not have your opponent will know the card and win.

However you cannot afford to bluff too often because, providing your bluffs are ignored, you are learning nothing about your opponent's hand.

So you must bluff some of the time and you must assume that your opponent is doing the same.

This game was invented by R. Isaacs back in 1967. He showed that whether you bluff or not, and whether you should suspect your opponent of bluffing, are both functions of the number of unknown cards each player holds.

These optimal strategies are encoded in the two data tables in Program I which are read into the two arrays *ibluff* and *hebluffs*.

ibluff gives the percentage of times the program should bluff in the 25 possible situations. Assume it has five cards and the opponent has four cards – this happens when the program ignores a possible bluff by its opponent at his very first turn.

At line 380 the number 27 is picked up and if 27 >= RND(1)*100, that is 27 per cent of the time, then the program will bluff – ask for a card in its hand not previously mentioned.

If, however, 27 < RND(1)*100, that is 73 per cent of the time, then the program goes to line 430 and makes a genuine guess at a card the opponent may have.

The second array, *hebluffs*, works in the same way and is used on the occasions when the program does not have the requested card. This is more complicated because, if it assumes a bluff and the opponent only has one card left, then the program must also assume it knows the hidden card.

There are two other situations where the program will guess the card.

- It didn't bluff and got "no". It waits for its turn and then says it must be

```
10 REM PROGRAM I
20 REM Aleatoire
30 REM (c)Computing with the Amstrad
40 MODE 1
50 DIM ibluff(5,5),hebluffs(5,5),deck
(11):true=-1:false=0:GOTO 80
60 guess=INT(RND(1)*11+1):IF deck(gue
ss)<his GOTO 60
70 deck(guess)=-deck(guess):RETURN
80 FOR his=1 TO 5:FOR mine=1 TO 5:REA
D ibluff(his,mine):NEXT mine:NEXT his
90 DATA 33,50,50,56,58
100 DATA 26,33,37,40,43
110 DATA 19,26,29,31,33
120 DATA 16,21,23,25,27
130 DATA 12,17,19,21,22
140 FOR his=1 TO 5:FOR mine=1 TO 5:RE
AD hebluffs(his,mine):NEXT mine:NEXT
his
150 DATA 50,50,41,38,33
160 DATA 33,33,28,25,22
170 DATA 38,32,27,24,21
180 DATA 32,30,26,23,21
190 DATA 32,29,25,23,21
200 wins=0:games=0:mine=1:his=2:it=3
210 FOR i=1 TO 11
220   deck(i)=it
230 NEXT i
240 PRINT"You have cards   ";
250 FOR i=1 TO 10
260   card=INT(RND(1)*11+1): IF deck(
card)<>it THEN 260
270   deck(card)=mine

280   IF i>5 THEN deck(card)=his:PRIN
T card;" ";
290 NEXT i
300 PRINT
310 win=false:bluffing=false:hehadit=
true:mycards=5:hiscards=5
320 IF RND(1)<0.5 THEN 470
330 PRINT
340 FOR i=1 TO 11
350   IF ABS(deck(i))=his THEN PRINT
i
360   IF deck(i)=-mine    THEN PRINT
"    ";i
370 NEXT i
380 IF ibluff(hiscards,mycards)<(RND(1
)*100 THEN 430
390 bluffing=true          :REM Ask fo
r one of my own cards
400 guess=INT(RND(1)*11+1): IF deck(g
uess)<>mine THEN 400
410 deck(guess)=-mine
420 mycards=mycards-1: GOTO 440
430 bluffing=false: GOSUB 60: hiscard
s=hiscards-1
440 PRINT "Have you got card ";guess;
450 INPUT; answer$:PRINT
460 IF LEFT$(answer$,1)="y" THEN heha
dit=true ELSE hehadit=false
470 INPUT"Your turn ";request
```

the last card it asked for. Note that it doesn't remember what it was.

● The program has no cards left – this is a forced risk because if the program doesn't make the guess then the opponent will win at his next turn.

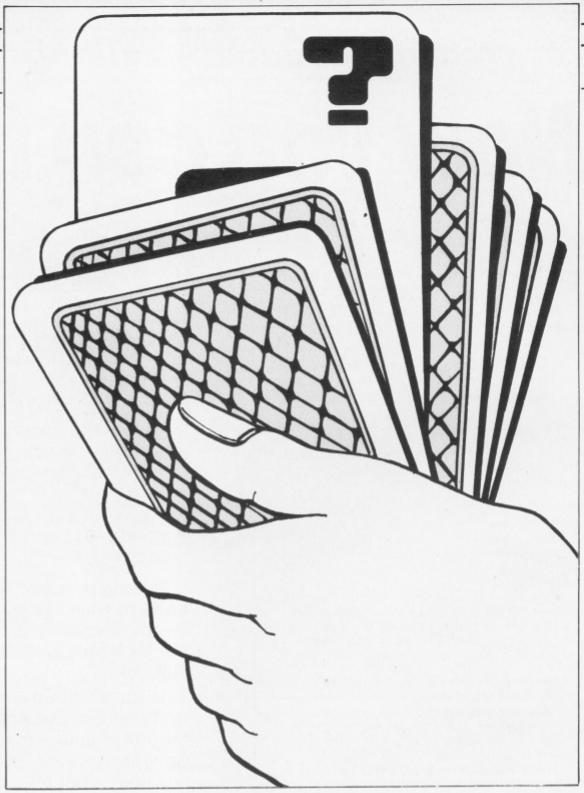
Type in the program including the debug lines 340 to 370. When you have got it working note that you enter a *negative* number to make your guess, and, if the program always goes first, it should win about 55 per cent of the games providing that you mix your bluffs in the same way as the program.

If, however, you are *always* honest you should beat the program 60 per cent of the time. This is because it is set to assume that you bluff occasionally. An interesting improvement is to make the program observe how often you bluff and let it modify the *hebluffs* array accordingly.

If you never bluff then this array should eventually become full of 100s – if you always bluff then the *hebluffs* array contents should all drop to zero.

Exactly how you observe and modify is up to you, but it is a nice example of a learning program that not only alters its own behaviour but should also make you change yours.

N.B: Enter a lower case "y" for "yes", anything else means you don't have the requested card.



```
480 IF request)0 THEN 510
490 IF ABS(deck(-request))<>it THEN w
in=true:REM He guessed wrong
500 GOTO 740
510 IF NOT bluffing AND NOT hehadit T
HEN 700
520 IF ABS(deck(request))<>mine THEN
600
530 PRINT"Yes I have card "¡request
540 IF deck(request)>0 THEN mycards=m
ycards-1
550 deck(request)=-mine
560 IF mycards>0 THEN 690
570 GOSUB 60: PRINT"I must guess "¡gu
ess
580 IF ABS(deck(guess))=it THEN win=t
rue:REM I guessed right
590 GOTO 740
600 REM Is he bluffing?
610 IF hebluffs(hiscards,mycards)<RND
(1)*100 THEN 650: REM Assume bluff el
se
620 PRINT"You are NOT bluffing. I gue
ss that is the card"
630 IF ABS(deck(request))=it THEN win
=true:REM He wasnt bluffing
640 GOTO 740
650 REM I think he is bluffing
660 PRINT"No, I do not have card "¡re
quest
670 deck(request)=-ABS(deck(request))
680 hiscards=hiscards-1
690 IF bluffing OR hehadit THEN 710
700 PRINT"It should be the last one I
 asked for":win=true:GOTO 740
710 IF hiscards)0 THEN 330: REM Dont
know yet
720 GOSUB 60:PRINT"The card  must be
"¡guess
730 IF ABS(deck(guess))=it THEN win=t
rue ELSE PRINT"You just bluffed me!"
740 i=0
750 i=i+1: IF ABS(deck(i))<>it THEN 7
50
760 PRINT"The card  was the "¡i¡games
=games+1
770 IF win THEN wins=wins+1:PRINT"I w
in" ELSE PRINT"I lose"
780 PRINT"My average is "¡wins/games*
100;"%":PRINT:GOTO 210
```

Give your fingers a rest . . .

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

See our special offer on Page 101.

Let your fingers do the talking...with this speci

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.

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. _____ Expiry date / /

☐ Visa Card No. _____

Signed _____

Name _____

Address _____

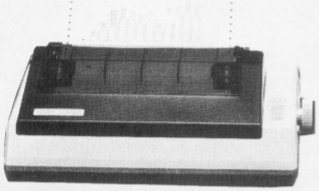
POST TO: Speech Synthesiser Offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

Allow 28 days for delivery

CA11

OPEN SUNDAYS 10 - 1

more unbeatable deals from Data star Systems!



THE *INCREDIBLE* NEW STAR SG-10 NEAR LETTER QUALITY PRINTER.

Star SG-10 £259 + VAT	£297.85
Parallel cable for any Micro (max. price)	£20
2 Spare ink ribbons	£5
2000 sheets of continuous listing paper	£15
Next day doorstep delivery service	£10
	£347.85

Datastar's all in price £297.85!!!

Just a few of the many features:—

- Easily switchable between Epson & IBM graphic/print models.
- Will print all ASCII codes from computers that can only send 7 bits on their parallel interface such as Amstrad CPC 464/664 and Apple II.
- Compatible with all word processing programs.
- 50 CPS NLQ mode available from switch on.
- 120 CPS draft mode.
- 2K print buffer — expandable to 10K

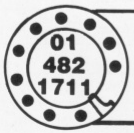
AMAZING AMSTRAD WORD PROCESSING OFFER ONLY FROM DataStar Systems

Amstrad CPC 6128 green screen computer	£299.00
Tasword/Mailmerge W/P package	£24.95
Star SG-10 printer package as above	£347.85
Next day doorstep delivery service	£20.00
Total value:	£691.80

DataStar's Superdeal Price £619.95!!!

Many more package deals available on other Amstrad models and makes of micro's.

All goods despatched FREE OF CHARGE by next day doorstep courier service



24 Hour Credit Card Hot Line

Post your cheques to

Datastar Systems UK

Unicom House, 182 Royal College Street, London NW1 9NN.
Telephone: 01-482 1711 Telex 295931 UNICOM G

PERSONAL CALLERS WELCOME — We are situated by the junction of Camden Road, near the railway bridge
MONDAY-FRIDAY 9-6 SUNDAY 10-1 *EXPORT ENQUIRIES WELCOME*

DEPTH CHARGES AWAY!

ALAN McLACHLAN puts the finishing touches to his Sub Hunt program

EVENIN' all. We ended last month's session by typing in the routine to hide 20 submarines in our undersea array. We also added a short line to reveal their "hidden" positions.

If you have done everything correctly so far, when you run the program you should see 20 plus signs indicating the whereabouts of the enemy.

We are now ready to enter the routines which take our input from the keyboard or joystick. But we must not forget to remove line 65 – otherwise the program won't progress any further.

The first routine we'll look at is the one to move the boat into the position where we want to drop the depth charge.

This and the next three subroutines are sandwiched in the WHILE... WEND loop I mentioned last month that dictates the length of each game – lines 70 and 120.

Before we discuss its contents it's important to understand how the Amstrad's INKEY function works. Although this is briefly explained in the User Instructions, a simple refresher here will do no harm.

Each key on the micro is given its own individual number and the firmware continuously interrogates the keyboard to report which keys, if any, are being pressed.

```
610 REM *** move boat ***
620 PEN #1,5
630 WHILE INKEY(18)=-1 AND INKEY(76)=
-1
640 newboatx=oldboatx+((INKEY(71)>-1)
-(INKEY(63)>-1))+((INKEY(74)>-1)-(INK
EY(75)>-1))
650 IF newboatx>15 THEN newboatx=15
660 IF newboatx<5 THEN newboatx=5
670 FOR delay=1 TO 250:NEXT delay
680 LOCATE #1,oldboatx,8:CALL &BD19:P
RINT #1," ":LOCATE #1,newboatx,8:PRI
NT #1,boat$
690 oldboatx=newboatx
700 WEND
710 RETURN
```

The INKEY function tests which keys the firmware has registered. Ignoring the more complicated issues of keys being paired with Shift or Control, they will return a value of either –1, not pressed, or 0 pressed.

Line 630 therefore simply states: *"While the Return key (18) and the Fire button (76) are not being pressed (–1) carry out the instructions up to the WEND at line 700".*

Line 640 uses the function inside some fairly complicated, but neat, logic statements in order to move the boat.

In simple terms what is happening is this: To the variable *newboatx,* we assign the value of *oldboatx* plus the sum of the truth values returned from the INKEY statement.

Let's have a closer look at line 640' and see how it works.

The INKEY numbers allocated to the Z and X keys, and the joystick left and right positions are 71, 63, 74 and 75 respectively. We use these numbers with INKEY to test whether the key is being pressed, when INKEY returns a value greater than –1.

To simplify things, let's consider a similar statement using the Z key as an example:

```
INKEY(71)>-1
```

What's happening? It looks as though we're trying to allocate the value –1 to the Z key – but have got our = and > signs in a twist. Try this:

```
PRINT 1>1
```

If you've typed it in correctly you'll get the response 0. Your micro is stating that it is of the "opinion" that the statement you have just made is FALSE, but since it can't spell, it uses the number 0. 1 isn't greater than 1 is it? Now try:

```
PRINT 2>1
```

Now see the difference. Your micro likes that. The statement is TRUE and the response is also TRUE, or –1 to our illiterate number-cruncher.

So with the above INKEY statement we're actually asking the micro to give an "opinion", based on our input, as to whether the Z key has been pressed. It will respond TRUE

(–1) or FALSE (0) depending on whether it was pressed or not. In strict computerese, on whether it "returns a value greater than –1".

Of course, we really want to test for several keys at once. No problem, we just use the same trick with different values, depending on the key or joystick responses we're looking for, stringing them all together.

That's what we've done in line 640. We make use of those values by adding them all together and adding or subtracting the result to or from the last location of the ship to move it backwards or forwards as appropriate.

Lines 650 and 660 make sure that our ship stays within the sea boundary then line 680 prints it at *newboatx* after printing two spaces at *oldboatx* to delete it from its last position.

The CALL to &BD19 was an attempt to reduce the flicker but it doesn't seem to have had much effect. Any ideas, anyone?

Finally line 690 assigns the value now in *newboatx* to the variable *oldboatx* before going back to detect another keypress.

If you have typed the subroutine in correctly you should now be able to move the boat left or right using either the Z and X keys, or the joystick.

```
720 REM *** keyboard input ***
730 WHILE INKEY$<>"":WEND
740 PEN #1,15:LOCATE #1,1,2:PRINT #1,
"What depth Capt'n ? "
750 LOCATE #1,1,5:PRINT #1,"Choose A
- M:";SPC(10)
760 k$="":WHILE k$<"A" OR k$>"M":k$=I
NKEY$:k$=UPPER$(k$):WEND
770 LOCATE #1,15,5:PRINT #1, k$
780 guess=ASC(k$)
790 FOR delay=1 TO 1000:NEXT delay
800 IF sea(newboatx,guess-64)=2 OR se
a(newboatx,guess-64)=3 THEN LOCATE #1
,1,2:PEN #1,5:PRINT #1,"You've tried
there !":FOR delay=1 TO 1000:NEXT del
ay:GOTO 740
810 RETURN
```

Listing II

Now we're ready for the keyboard input routine called at line 90.

This is almost identical to the routine used in Smiley Hunt, the only major difference being the addition of line 800.

Our array element can contain any one of four numbers that register its state at the time it is selected.

The meanings of the numbers are:

0	Empty
1	Hidden submarine
2	Already selected/empty
3	Already selected/hit

Line 800 tests to see whether a 2 or a 3 is contained in our selected location. If the result is positive, the appropriate message is displayed and the program is sent back immediately to the input line. This avoids the dropping of an unnecessary depth charge.

Now we are ready for the dropping routine itself. Type in Listing III.

This routine takes effect immediately you have input the depth for the charge. It was rather more complicated than I first anticipated because of the way the depth charge wiped out everything in its path as it descended.

My original idea was that if you made a wrong guess this was indicated by an X on the screen. Alternatively, a direct hit on a sub would be indicated first of all by a graphic explosion, then by a submarine printed on the screen.

This, you will find later, is exactly what happens, but before I found the answer, whatever was on screen was wiped off by the dropping charge. I overcame the problem as previous ones by shouting "Help!".

```
820 REM *** drop mine ***
830 PEN #1,3
840 LOCATE #1,1,5:PRINT #1,"Fire ";mi
nes+1;SPC(13)
850 SOUND 129,10,300,15,1,1,1
860 FOR newbomby= 1 TO guess-64
870 FOR count = 1 TO 50:NEXT count
880 IF newbomby>1 AND (sea(newboatx,n
ewbomby-1)=0 OR sea(newboatx,newbomby
-1)=1) THEN LOCATE #0,newboatx,newbom
by-1:PRINT #0," "
890 IF newbomby>1 AND sea(newboatx,ne
wbomby-1)=2 THEN LOCATE #0,newboatx,n
ewbomby-1:PEN #0,4:PRINT #0,CHR$(42)
900 IF newbomby>1 AND sea(newboatx,ne
wbomby-1)=3 THEN LOCATE #0,newboatx,n
ewbomby-1:PRINT #0,CHR$(243)
910 LOCATE #0,newboatx,newbomby:PEN #
0,5:PRINT #0,CHR$(242)
920 NEXT newbomby
930 mines=mines+1:FOR delay=1 TO 200:
NEXT delay:FOR vol=2 TO 15:SOUND 129,
1500,100,vol,2,0,15:NEXT vol
940 RETURN
```

Listing III

The solution was to use lines 880-900 which test the location that the charge has just left *(newbomby – 1)*, to see what character was there originally, in order to put that character back again once the charge had gone past.

Under normal circumstances, as the charge drops it is followed by a sea-coloured space to wipe out the original shape. This only happens – line 880 – when the array element contains either a 0 or a 1. If the element contains a 2 – line 890 – the space is replaced by the X character, and if a 3 – line 900 – by CHR$(243) the submarine.

You will no doubt find during testing that your plus signs, used to indicate the hidden subs, are wiped out if you drop through any. The trick is not to drop through, but to select the shallowest location in the grid

first. This problem won't arise once we have removed line 585, but for the time being leave it intact.

The next routine, to check for hits, is again the same routine we used in Smiley Hunt, as is the one to indicate a direct hit.

Enter Listings IV and V. Line 970 ensures that if you have a direct hit, the subroutine at line 1200 is called to print the explosion character (244) followed by the submarine (243).

It also puts the number 3 in the array element and increments the variable *subs*. If you draw a blank, an X is placed at the selected location, and a series of checks is carried out in order to give clues as to the location of any submarine that may be at a different depth, a different spot, or both.

Listings VI and VII are also virtually identical to the "Game Over" and "Results" subroutines from Smiley Hunt. Once you have typed these in, the game is complete.

You should now be able to run it and, selecting mostly locations indicated by the plus signs, test that all the routines we've added this month work correctly. Once you're happy that this is so, you can remove line 585.

We have made quite a lot of alterations by putting lines in and pulling them out again. So, just in case you, or indeed I, have left an odd line in or out, the complete program should match Listing VIII.

Now you'll find you have quite an entertaining strategy game that needs quite a lot of thought and planning to avoid wasting a lot of depth charges on locations that couldn't possibly have held submarines. Better still, you understand how it works.

Anyway that's enough for another month. I hope you've enjoyed getting involved with my program and I'm now planning the next. Wow that's Christmas already . . .

Who knows, I may even make a living out of this . . . See you soon.

```
950 REM *** check hits ***
960 flagx=0:flagy=0
970 IF sea(newboatx,guess-64)=1 THEN
GOSUB 1200:RETURN ELSE LOCATE #0,newb
oatx,guess-64:PEN #0,4:PRINT #0,CHR$(
42):SOUND 129,1500,100,10,2,0,15
980 FOR locx=1 TO 11:IF sea(locx,gues
s-64)=1 THEN flagx=1
990 NEXT locx
1000 FOR depth=1 TO 14:IF sea(newboat
x,depth)=1 THEN flagy=1
1010 NEXT depth
1020 IF flagx=1 AND flagy=0 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Depth O
.K, - move !";
1030 IF flagy=1 AND flagx=0 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Wrong d
epth!";
1040 IF flagy=1 AND flagx=1 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Depth/l
ocation O.K!";
1050 FOR delay=1 TO 2000:NEXT delay:s
ea(newboatx,guess-64)=2:LOCATE #0,1,1
6:PRINT #0,SPC(19);:LOCATE #1,1,2:PRI
NT #1,SPC(19);:LOCATE #1,1,5:PRINT #1
,SPC(19);
1060 RETURN
```

Listing IV

```
1200 REM *** found sub ***
1210 LOCATE #0,newboatx,guess-64:PEN
#0,5:PRINT #0, CHR$(244):FOR delay =
1 TO 2000:NEXT delay:LOCATE #0,newboa
tx,guess-64:PEN #0,5:PRINT #0, CHR$(2
43):sea(newboatx,guess-64)=3:subs=sub
s+1:LOCATE #1,1,2:PRINT #1,SPC(19);:L
OCATE #1,1,5:PRINT #1,SPC(19);
1220 RETURN
```

Listing V

```
1070 REM *** Game Over ***
1080 IF mines >29 AND subs <5 THEN LO
CATE #1,1,2:PRINT #1,"You're out of a
mmo !"
1090 IF subs >4 AND mines <30 THEN LO
CATE #1,1,2:PRINT #1,"That's the lot
!"
1100 IF subs >4 AND mines >29 THEN LO
CATE #1,1,2:PRINT #1,"That's the lot
!"
1110 FOR delay = 1 TO 3000: NEXT dela
y
1120 RETURN
```

Listing VI

```
1130 REM *** results ***
1140 CLS: LOCATE 1,10:PRINT "You used
";mines;"mines"
1150 LOCATE 1,12:PRINT "and destroyed
"subs"subs"
1160 LOCATE 1,14:PRINT "Try again Y/N
 ?"
1170 k$="":WHILE k$<>"Y" AND k$<>"N":
k$=INKEY$:k$=UPPER$(k$):WEND
1180 IF k$<>"Y" THEN END
1190 RETURN
```

Listing VII

```
10 REM Al's sub hunt
20 GOSUB 160:REM title screen
30 GOSUB 280:REM initialise
40 WHILE -1
50 GOSUB 430:REM draw scene
60 GOSUB 540:REM hide subs
70 WHILE subs<5 AND mines<30
80 GOSUB 610:REM move boat
90 GOSUB 720:REM keyboard input
100 GOSUB 820:REM drop mine
110 GOSUB 950:REM check hits
120 WEND
130 GOSUB 1070:REM game over
140 GOSUB 1130:REM results
150 WEND
160 REM *** title ***
170 MODE 0:DIM sea(19,21):BORDER 0:PA
PER 5:CLS
180 PEN 3:FOR x=3 TO 17:LOCATE x,4:PR
INT "*":NEXT x
190 FOR x=3 TO 17:LOCATE x,17:PRINT "
*":NEXT x
200 LOCATE 9,8:PRINT "Al's"
210 a$="Sub Hunt"
220 FOR x= 1 TO LEN(a$):b$=MID$(a$,x,
1):PEN (RND)*3:LOCATE 2*x+1,12:PRINT
b$:READ pitch:SOUND 1,pitch
230 FOR delay =1 TO 500:NEXT delay
240 NEXT x
250 FOR delay = 1 TO 1500:NEXT delay
260 CLS:FOR delay = 1 TO 1500:NEXT de
lay
270 RETURN
280 REM *** initialise ***
290 SYMBOL AFTER 240
300 SYMBOL 240,0,0,0,0,0,0,7,255,127
310 SYMBOL 241,0,0,0,0,0,208,240,255,25
2
320 SYMBOL 242,0,0,0,0,24,24,0,0,0
330 SYMBOL 243,0,0,16,24,126,255,126,
```

Listing VIII

```
0
340 SYMBOL 244,17,128,4,88,8,34,40,13
2
350 boat$=CHR$(240)+CHR$(241)
360 rightland$=CHR$(95)+CHR$(210)+CHR
$(140)+CHR$(140)
370 leftland$=CHR$(140)+CHR$(210)+CHR
$(95)+CHR$(95)
380 ENT 1,200,2,1:ENV 1,15,-1,25:ENV
2,15,-1,10
390 BORDER 10
400 WINDOW #0,1,20,9,25:PAPER #0,0:CL
S #0
410 WINDOW #1,1,20,1,8:INK 8,11:PAPER
#1,0:INK 15,24:CLS #1
420 RETURN
430 REM *** draw scene ***
440 mines=0:subs=0:oldboatx=10
450 CLS #0:CLS #1
460 PEN #1,5:LOCATE #1,oldboatx,8:PRI
NT #1,boat$
470 PEN #1,12:LOCATE #1,1,8:PRINT #1,
leftland$
480 LOCATE #1,17,8:PRINT #1,rightland
$;
490 FOR numy=1 TO 13:LOCATE #0,1,numy
:PEN #0,4:PRINT #0,CHR$(numy+96):NEXT
numy
500 FOR locx=6 TO 18:FOR locy=1 TO 13
:sea(locx,locy)=0:NEXT locy:NEXT locx
510 PEN #1,15:LOCATE #1,1,2:PRINT #1,
"Z, X or Joystick"
520 LOCATE #1,1,5:PRINT #1,"then Ente
r or Fire"
530 RETURN
540 REM *** hide subs ***
550 FOR count=1 TO 20
560 subx=INT(RND*11)+5:suby=INT(RND*1
3)+1
570 IF sea(subx,suby)=1 THEN GOTO 560
580 sea(subx,suby)=1
590 NEXT count
600 RETURN
610 REM *** move boat ***
620 PEN #1,5
630 WHILE INKEY(18)=-1 AND INKEY(76)=
-1
640 newboatx=oldboatx+((INKEY(71)>-1)
-(INKEY(63)>-1))+((INKEY(74)>-1)-(INK
EY(75)>-1))
650 IF newboatx>15 THEN newboatx=15
660 IF newboatx<5 THEN newboatx=5
670 FOR delay=1 TO 250:NEXT delay
680 LOCATE #1,oldboatx,8:CALL &BD19:P
RINT #1," ":LOCATE #1,newboatx,8:PRI
NT #1,boat$
690 oldboatx=newboatx
700 WEND
710 RETURN
720 REM *** keyboard input ***
730 WHILE INKEY$()"":WEND
740 PEN #1,15:LOCATE #1,1,2:PRINT #1,
"What depth Capt'n ? "
750 LOCATE #1,1,5:PRINT #1,"Choose A
- M:";SPC(10)
760 k$="":WHILE k$("A" OR k$)"M":k$=I
NKEY$:k$=UPPER$(k$):WEND
770 LOCATE #1,15,5:PRINT #1, k$
780 guess=ASC(k$)
790 FOR delay=1 TO 1000:NEXT delay
800 IF sea(newboatx,guess-64)=2 OR se
a(newboatx,guess-64)=3 THEN LOCATE #1
,1,2:PEN #1,5:PRINT #1,"You've tried
there !":FOR delay=1 TO 1000:NEXT del
ay:GOTO 740
810 RETURN
820 REM *** drop mine ***
830 PEN #1,3
840 LOCATE #1,1,5:PRINT #1,"Fire ":mi
nes+1;SPC(13)
850 SOUND 129,10,300,15,1,1,1
860 FOR newboomby= 1 TO guess-64
870 FOR count = 1 TO 50:NEXT count
880 IF newboomby>1 AND (sea(newboatx,n
ewboomby-1)=0 OR sea(newboatx,newboomby
-1)=1) THEN LOCATE #0,newboatx,newbom
by-1:PRINT #0," "
890 IF newboomby>1 AND sea(newboatx,ne
wboomby-1)=2 THEN LOCATE #0,newboatx,n
ewboomby-1:PEN #0,4:PRINT #0,CHR$(42)
900 IF newboomby>1 AND sea(newboatx,ne
wboomby-1)=3 THEN LOCATE #0,newboatx,n
ewboomby-1:PRINT #0,CHR$(243)
910 LOCATE #0,newboatx,newboomby:PEN #
0,5:PRINT #0,CHR$(242)
920 NEXT newboomby
930 mines=mines+1:FOR delay=1 TO 200:
NEXT delay:FOR vol=2 TO 15:SOUND 129,
1500,100,vol,2,0,15:NEXT vol
940 RETURN
950 REM *** check hits ***
960 flagx=0:flagy=0
970 IF sea(newboatx,guess-64)=1 THEN
GOSUB 1200:RETURN ELSE LOCATE #0,newb
oatx,guess-64:PEN #0,4:PRINT #0,CHR$(
42):SOUND 129,1500,100,10,2,0,15
980 FOR locx=1 TO 11:IF sea(locx,gues
s-64)=1 THEN flagx=1
990 NEXT locx
1000 FOR depth=1 TO 14:IF sea(newboat
x,depth)=1 THEN flagy=1
1010 NEXT depth
1020 IF flagx=1 AND flagy=0 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Depth O
.K, - move !";
1030 IF flagy=1 AND flagx=0 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Wrong d
epth!";
1040 IF flagy=1 AND flagx=1 THEN LOCA
TE #0,1,16:PEN #0,5:PRINT #0,"Depth/l
ocation O.K!";
1050 FOR delay=1 TO 2000:NEXT delay:s
ea(newboatx,guess-64)=2:LOCATE #0,1,1
6:PRINT #0,SPC(19);:LOCATE #1,1,2:PRI
NT #1,SPC(19);:LOCATE #1,1,5:PRINT #1
,SPC(19);
1060 RETURN
1070 REM *** Game Over ***
1080 IF mines >29 AND subs <5 THEN LO
CATE #1,1,2:PRINT #1,"You're out of a
mmo !"
1090 IF subs >4 AND mines <30 THEN LO
CATE #1,1,2:PRINT #1,"That's the lot
!"
1100 IF subs >4 AND mines >29 THEN LO
CATE #1,1,2:PRINT #1,"That's the lot
!"
1110 FOR delay = 1 TO 3000: NEXT dela
y
1120 RETURN
1130 REM *** results ***
1140 CLS: LOCATE 1,10:PRINT "You used
";mines;"mines"
1150 LOCATE 1,12:PRINT "and destroyed
"subs"subs"
1160 LOCATE 1,14:PRINT "Try again Y/N
?"
1170 k$="":WHILE k$<>"Y" AND k$<>"N":
k$=INKEY$:k$=UPPER$(k$):WEND
1180 IF k$<>"Y" THEN END
1190 RETURN
1200 REM *** found sub ***
1210 LOCATE #0,newboatx,guess-64:PEN
#0,5:PRINT #0, CHR$(244):FOR delay =
1 TO 2000:NEXT delay:LOCATE #0,newboa
tx,guess-64:PEN #0,5:PRINT #0, CHR$(2
43):sea(newboatx,guess-64)=3:subs=sub
s+1:LOCATE #1,1,2:PRINT #1,SPC(19);:L
OCATE #1,1,5:PRINT #1,SPC(19);
1220 RETURN
1230 REM *** music data ***
1240 DATA 119,106,95,89,80,71,63,60
```

Evesham Micros

DOUBLER

HAVING PROBLEMS BACKING UP THE LATEST LOADING TECHNIQUES?

DOUBLER+ Will solve ALL your problems

After **enormous success** on the CBM 64 and Spectrum we now have "Doubler" available for the CPC464. Designed by DRAYSOFT, inventors of the **infamous** "Interface III", this superb unit will operate with **100% success guaranteed**. It requires the use of an auxiliary cassette deck.

Simply plug in the "Doubler" on the expansion port, connect a tape player to it with the lead supplied, LOAD the software supplied in the normal way and you are **ready to start**.

Using digital techniques "Doubler" will create a brand new machine back up of **ALL** software **100% success GUARANTEED**

As a bonus **"Speedcop" is included** in the package. This is a superb conventional back up copier that will handle **all normal AMSTRAD loaders**. It has all the usual options like choice of save speed deprotect basic etc.

Solve all your back up problems in one move, remember we guarantee success.

ONLY £16.95

3" DISCS

Fed up with paying silly prices?

1–9 £4.25 each
10 for £40.00

Buy 10 and we will put "Excat" on one of the discs for you free of charge.

Disc User? EXCAT

Is a **very useful** utility that will give you the start and end addresses of all files on your discs, including binary. At little more than the cost of the disc that it is supplied on this is a worthwhile purchase. (The rest of the disc can be used as normal)

ONLY £6.95

All prices include P&P and VAT. Send cheque, Postal Order of Credit Card No. Credit Card order accepted by phone. Callers welcome. Wide range of goods stocked. Trade enquiries welcome. European orders send price as advertised. Outside Europe £2.00 for airmail. Mail order to Evesham Shop please.

EVESHAM MICROS
BRIDGE STREET, EVESHAM, WORCS., WR11 4RY.
Tel: 0386 41989

MICRO CENTRE
1756 PERSHORE ROAD, COTTERIDGE, BIRMINGHAM.
Tel: 021-458 4564

M & A SOFTWARE
announce the launching of their

AMSTRAD ADVENTURE CLUB

FOR THE AMSTRAD CPC464

Each month members will receive a double cassette. Cassette One will consist of an exciting Adventure Game (text only) for you to solve with logic, common sense and a little cunning and the chance to win a super prize.

Cassette Two will consist of a scenario of the game and instructions for playing. Also details of the month's prizes and how to claim. It will also contain hints and tips for solving the previous game (if you haven't already done so), so no follow up books to buy to help you. Plus details of next months game and prizes plus all our latest news, views and special offers to Club Members.

This month's exciting game is

MASTER SPY

An Adventure set in wartime Europe with over 100 locations and a vocabulary of over 300 words. Your task is to steal a top secret and prevent the enemy from learning its secret and get it back safely to base whilst avoiding the secret police and double agents.

This month's Super Prize for the first correct solution we receive will be:

THE NEW & EXCITING AMSTRAD CPC6128
with Colour Monitor. Plus 3 months free supply of games to the ten runners-up.
(All winners will be announced in this magazine)

Enrolment in the Club costs just £5 and includes your first game. After that each Double Cassette costs just £3.99 and membership can be cancelled at any time.

Send your Cheque or Postal Order for £5 payable to M & A Software to:

M & A SOFTWARE
DEPT ACW, 4 SHIPQUAY STREET, LONDONDERRY BT48 6DN

LOW PRICES – FAST SERVICE

DISCOUNT HARDWARE

NOTE: All Prices include VAT & Carriage

	Ex VAT	Inc VAT & Carriage
DOT MATRIX PRINTERS		
Canon PW1080A (nlq)	244.71	285.95
Canon PW1156	340.16	395.95
Epson RX80 (100cps)	194.21	223.95
Epson LX80 (new) 100cps Normal, 22cps NLQ. Friction	201.98	235.74
Epson LX80 Tractor Attachment	20.00	25.50
Epson LX80 Sheet Feeder	49.95	61.53
Epson FX80 (160cps)	320.21	372.95
Epson QI1500 (200cps, professional)	948.21	1104.00
Epson P40 (Thermal, 80 col)	77.16	92.95
DAISYWHEEL PRINTERS		
Quen Data/Daisystep 2000	200.16	235.95
Juki 6100	295.16	345.25
COLOUR PRINTERS		
Seikosha GP700A (Special offer – few only)	235.21	275.88
HIGH QUALITY DISK AT VERY LOW PRICES (Ex Dysan France Factory)		
High quality diskettes 5.25" boxed in 10's with plastic library case		
40 Track SSSD	13.50	16.50
40 Track DSDD	13.50	16.50
80 Track SSSD	13.50	16.50
80 Track DSDD	13.50	16.50
HEAD CLEANING KITS		
3M Standard	12.50	15.95
Rexel Drive/KBD/VDU Cleaning Kit	19.11	22.95
LISTING PAPER		
60 GSM (listing) 9.5" x 11" 2000 sheets	10.50	17.25
80 GSM (letter grade) 9.25" x 12" 2000 sheets	13.40	20.58
Labels – Box of 8000	34.51	44.86
(2 abreast on 9.5" backing with central perf.)		
HIGH QUALITY DISK DRIVES Opus Drives with Mitsubishi/Sanyo mechanism 2 year guarantee		
With Built In Power Supply		
S100 40 track single side (100K)	96.80	115.95
S2000 40 track double side (200K)	119.95	142.50
S400S 80 track double side (400K)	134.20	158.95
Dual Drives (with Power Supply)		
D200 2 x 40 track single side (200K)	185.95	218.95
D400S 2 x 80 track single side (400K)	238.95	279.95
D4000 2 x 40 track double side (400K)	218.95	256.50
D800S 2 x 80 track double side (800K)	242.95	284.95

NOTE – All 80 track drives have a 40/80 switch!

FREE!
WITH EVERY PRINTER
– LISTING PAPER + AMSTRAD LEAD

MAIL ORDER ONLY – DESPATCHED BY RETURN

Microdis

Dept. CA.11.85
46 Middle Street, Blackhall, Hartlepool, Cleveland TS27 4EA.
Tel: 0783 872430.

PRINTERLAND

ORDERED TODAY – DELIVERED TOMORROW

UK's LOWEST PRINTER PRICES
FULL PRINTER SUPPORT FOR THE AMSTRAD
ORDERED TODAY – DELIVERED TOMORROW

		EX VAT	INC VAT
■ DOT MATRIX			
SHINWA CPA80		£165.00	£189.75
EPSON FX80		£314.00	£361.10
EPSON FX100		£425.00	£488.75
■ NEAR LETTER QUALITY			
KAGA KP-810		£235.00	£270.25
CANON 1080A	SPECIAL OFFER	£240.00	£276.00
KAGA P-910		£335.00	£385.25
CANON 1156A	SPECIAL OFFER	£335.00	£385.25
■ DAISY WHEEL			
DAISY STEP 2000		£225.00	£258.75
JUKI 6100		£325.00	£373.75
EPSON DX100		£356.00	£409.40
■ AMSTRAD ADD ONS			
TASWORD TWO		£15.60	£17.95
TASPRINT		£7.74	£8.90
TASCOPY		£7.74	£8.90
PRINTER CABLE		£11.00	£12.65
■ COLOUR PRINTERS			
EPSON JX-80	SPECIAL OFFER	£450.00	£517.50
■ DISK DRIVES			
AMSTRAD DDI-1		£160.00	£184.00

Educational, Government plus Overseas Orders Welcome. Delivery Printers (Securicor) £10.00

Printerland
Unit 27, Estate Buildings, Railway St, Huddersfield HD1 1JP
Showroom open Mon-Fri 9-6pm. Sat morn 9-1pm
Tel: Huddersfield (0484) 514105 or (0484) 687875

Z80 disassembler

A must for the keen machine code programmer's toolkit, by ROLAND WADDILOVE

TO complement the Z80 assembler listed in the July issue of *Computing with the Amstrad*, RAW, here's a Z80 disassembler. As the name implies, it's the exact opposite of an assembler and is a valuable machine code programmer's tool.

Machine code is pretty unintelligible. As you probably know, it's just a series of binary numbers in the range 0 to 255. When we write machine code we write it in assembly language first, as it's much easier to understand, and then translate the mnemonics into machine code.

This is fine, but what happens if you haven't got the original assembly listing? This may occur if you've lost it, but more commonly if you didn't have it in the first place.

Many games and utilities are written in machine code and although most games are protected, utilities are often not.

Just how does that screen dump or RSX work? There's a lot to be learnt by looking at other people's programming techniques.

Assembly listings also take up a great deal of space in the magazine and not everyone has an assembler – order a back issue immediately – so a routine may be listed as a series of hex numbers in data statements.

On top of this are two ROMs containing 16k of machine code – the Basic and operation system. Many of you will have a disc interface with a further 16k of code. There's a lot of interesting material lurking about in there.

What is needed is a program which will convert the machine code back into an assembly listing. This is the function of this disassembler.

All of the Amstrad's 64k of RAM can be disassembled, except for the 7k taken up by the program itself. Also any of the 252 possible ROMs can be selected and disassembled.

A memory lister producing a hexadecimal and Ascii dump has been included. Again this can display data in RAM or any ROM.

All the ROMs are interrogated when the program is first run and their type and select address printed. The addresses are needed when choosing a ROM. There's also an option to send all output to the printer if a hard copy is required.

Unfortunately a disassembler can't reproduce an exact copy of an assembly listing as it has no way of knowing the labels used by the programmer. The address or number is printed instead.

Another area of difficulty is data tables. A disassembler can't detect data in the middle of a program. It simply attempts to disassemble the lot.

Apart from these minor difficulties it produces a very close copy of the original listing. You should find it an extremely useful tool.

The program uses virtually no data and is very compact.

It works out each instruction by looking at the bit pattern of the object code. It then selects the appropriate subroutine.

The algorithm used is superb and can be found in Mastering Machine Code On Your ZX81 OR ZX80, by Toni Baker. There you'll find a full description of how the program works.

VARIABLES

A	Address.
dis$	Disassembled instruction.
char$	Object code in Ascii.
byte	1 byte of object code.
stream	Where to send the text.
rom	ROM selected.

```
10 REM Z80 Disassembler
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 DEFINT b-z
50 MODE 1:INK 0,0:BORDER 0:PEN 1:PAPE
R 3
60 LOCATE 1,25:PRINT "        Press S
PACE BAR for menu        ";
70 LOCATE 1,1:PRINT "        Z80   Disa
ssembler/Hex dump        ";:PRINT:PAPER
0:WINDOW #0,1,39,2,24:PRINT
80 DIM r$(7),s$(3),q$(3),c$(7),x$(7)
90 FOR i=0 TO 7:READ r$(i),c$(i),x$(i
):NEXT
100 DATA B,NZ,"ADD A,",C,Z,"ADC A,",D
,NC,SUB,E,C,"SBC A,",H,PO,AND,L,PE,XO
R,x,P,OR,A,M,CP
110 FOR i=0 TO 3:READ s$(i),q$(i):NEx
T
120 DATA BC,BC,DE,DE,y,y,SP,AF
130 FOR i=0 TO 69:READ char$:POKE &B4
48+i,VAL("&"+char$):NEXT
```

```
140 DATA DD,4E,00,CD,15,B9,DD,6E,02,D
D,66,03,77,23,36,00,C9,DD,4E,00,CD,0F
,B9,C5,DD,6E,04,DD,66,05,7E,DD,6E,02,
DD,66,03,77,23,36,00,C1,C3,18,B9,CD,0
6,B9,F5,DD,6E,00,DD,66,01,7E,DD,6E,02
,DD,66,03,77,23,36,00,F1,C3,0C,B9
150 PEN 2:PRINT TAB(6)"ROM's.....":PR
INT:j=0:byte=0
160 FOR i=0 TO 251
170 CALL &B448,@byte,i
180 IF byte(>&80 THEN PEN 2:PRINT TAB
(6);:CHR$(15);CHR$(3);": ";:PEN 1 ELS
E IF NOT j THEN PEN 2:PRINT TAB(6);:C
HR$(15);CHR$(3);": ";:PEN 1:PRINT"On
Board ROM (BASIC).":j=-1
190 IF byte=0 THEN PRINT "Foreground
ROM." ELSE IF byte=1 THEN PRINT "Back
ground ROM." ELSE IF byte=3 THEN PRIN
T "Extension Foreground ROM."
200 NEXT
210 PEN 2:PRINT:INPUT "Start address"
;a
220 IF a(0 THEN a=65536+a
230 IF a>49151 THEN PRINT CHR$(11);SP
C(35);CHR$(13);:INPUT "Which ROM";rom
 ELSE rom=255
240 IF a(&4000 THEN PRINT CHR$(11);SP
C(35);CHR$(13);:INPUT "ROM or RAM";ch
ar$:IF UPPER$(char$)="ROM" THEN rom=2
54
250 PRINT CHR$(11);SPC(35);CHR$(13);:
INPUT "Printer (Y/N)";char$:IF UPPER$
(char$)="Y" THEN stream=8 ELSE stream
=0
260 PRINT CHR$(11);SPC(35);CHR$(13);:
INPUT "Hex or Disassemble (H/D)";opt$
:CLS:IF UPPER$(opt$)="H" THEN 1110
270 PEN #1,1:PAPER #1,3:PRINT #1,CHR$
(30);"Addr:Hex             Mnemonic
  ASCII "
280 dis$="":char$=""
290 PEN 2:PRINT #stream,TAB(20)dis$;:
PEN 1:PRINT #stream,TAB(35)char$
300 class=0:index=0:char$=""
310 PRINT #stream,HEX$(a,4);":";:PEN
3
320 GOSUB 530
330 IF byte=&76 THEN IF class=0 AND i
ndex=0 THEN dis$="HALT":GOTO 290
340 IF byte=&CB THEN class=1:GOTO 320
350 IF byte=&ED THEN class=2:GOTO 320
360 IF byte=&DD THEN index=1:GOTO 320
370 IF byte=&FD THEN index=2:GOTO 320
380 IF class=0 THEN ON f+1 GOSUB 690,
820,840,860
390 IF class=1 AND index>0 THEN GOSUB
 530:a=a-2
400 IF class=1 THEN ON f+1 GOSUB 960,
970,980,990
410 IF class=2 THEN ON f GOSUB 1010,1
100
420 x1=INSTR(dis$,"x"):y1=INSTR(dis$,
"y")
430 IF index=1 AND x1>0 THEN GOSUB 63
0:dis$=LEFT$(dis$,x1-1)+"(IX+&"+HEX$(
byte)+")"+MID$(dis$,x1+1):IF class(>1
 THEN GOSUB 590 ELSE a=a+2
440 IF index=1 AND y1>0 THEN dis$=LEF
T$(dis$,y1-1)+"IX"+MID$(dis$,y1+1):GO
TO 420
450 IF index=0 AND y1>0 THEN dis$=LEF
T$(dis$,y1-1)+"HL"+MID$(dis$,y1+1):GO
TO 420
460 IF index=0 AND x1>1 THEN dis$=LEF
T$(dis$,x1-1)+"(HL)"+MID$(dis$,x1+1)
470 IF index=2 AND y1>0 THEN dis$=LEF
T$(dis$,y1-1)+"IY"+MID$(dis$,y1+1):GO
TO 420
480 IF index=2 AND x1>0 THEN GOSUB 63
0:dis$=LEFT$(dis$,x1-1)+"(IY+&"+HEX$(
byte)+")"+MID$(dis$,x1+1):IF class(>1
 THEN GOSUB 590 ELSE a=a+2
490 IF INSTR(dis$,"VV") THEN GOSUB 63
0:i=byte:a=a+1:GOSUB 630:a=a-1:dis$=L
EFT$(dis$,INSTR(dis$,"VV")-1)+"&"+HEX
$(i+256*byte)+MID$(dis$,INSTR(dis$,"V
V")+2):GOSUB 590:GOSUB 590
500 IF INSTR(dis$,"V") THEN GOSUB 630
:dis$=LEFT$(dis$,INSTR(dis$,"V")-1)+"
&"+HEX$(byte)+MID$(dis$,INSTR(dis$,"V
")+1):GOSUB 590
510 IF INKEY$=" " THEN 210 ELSE 290
520 REM ------- get data ----------
530 GOSUB 630:bit$=BIN$(byte,8)
540 f=VAL("&X"+LEFT$(bit$,2))
550 g=VAL("&X"+MID$(bit$,3,3))
560 h=VAL("&X"+RIGHT$(bit$,3))
570 j=VAL("&X"+MID$(bit$,3,2))
580 k=VAL("&X"+MID$(bit$,5,1))
590 GOSUB 630:IF byte)31 AND byte(127
 THEN char$=char$+CHR$(byte) ELSE cha
rs=char$+"."
600 PRINT #stream,HEX$(byte,2);" ";:a
=a+1
610 RETURN
620 REM ------- read byte ----------
630 IF rom=255 THEN byte=PEEK(a):RETU
RN
640 IF rom=254 THEN CALL &B475,@byte,
a:RETURN
650 CALL &B459,a,@byte,rom
660 RETURN
670 REM ---------- class=0 ---------
680 REM ---------- f=0 ----------
690 ON h+1 GOTO 700,740,750,760,770,7
80,790,800
700 IF g>3 THEN dis$="JR "+c$(g-4)+",
W" ELSE dis$=MID$("NOP      EX AF,AF'
DJNZ W   JR W      ",1+g*9,9)
710 GOSUB 630:IF INSTR(dis$,"W")>0 AN
D byte>127 THEN dis$=LEFT$(dis$,INSTR
(dis$,"W")-1)+"&"+HEX$(a+byte-255):GO
TO 590
720 IF INSTR(dis$,"W")>0 AND byte(128
 THEN dis$=LEFT$(dis$,INSTR(dis$,"W")
-1)+"&"+HEX$(1+a+byte):GOTO 590
730 RETURN
740 IF k THEN dis$="ADD y,"+s$(j):RET
URN ELSE dis$="LD "+s$(j)+",VV":RETUR
N
750 dis$="LD "+MID$("(BC),AA,(BC)(DE)
,AA,(DE)(VV),yy,(VV)(VV),AA,(VV)",1+g
*6,6):RETURN
760 IF k THEN dis$="DEC "+s$(j):RETUR
N ELSE dis$="INC "+s$(j):RETURN
770 dis$="INC "+r$(g):RETURN
780 dis$="DEC "+r$(g):RETURN
790 dis$="LD "+r$(g)+",V":RETURN
800 dis$=MID$("RLCARRCARLA RRA DAA CP
L SCF CCF ",1+g*4,4):RETURN
810 REM ---------- f=1 ----------
820 dis$="LD "+r$(g)+","+r$(h):RETURN
830 REM ---------- f=2 ----------
840 dis$=x$(g)+" "+r$(h):RETURN
850 REM ---------- f=3 ----------
860 ON h+1 GOTO 870,880,890,900,910,9
20,930,940
870 dis$="RET "+c$(g):RETURN
880 IF k THEN  dis$=MID$("RET    EXX
   JP (y) LD SP,y",1+j*7,7):RETURN EL
SE dis$="POP "+q$(j):RETURN
890 dis$="JP "+c$(g)+",VV":RETURN
900 dis$=MID$("JP VV    **********OUT
(V),AIN A,(V) EX (SP),yEX DE,HL DI
   EI       ",1+g*9,9):RETURN
910 dis$="CALL "+c$(g)+",VV":RETURN
```

ARE YOU READY FOR BRAIN TO BRAIN COMBAT
DOCTOR WHO
AND THE MINES OF TERROR
MICROPOWER
IN ASSOCIATION WITH THE BBC
SPECTRUM COMMODORE £14.95 (CASSETTE)
AMSTRAD
BBC (ROM/CASSETTE) £8.95
(ROM/DISK) £12.95

```
920 IF k THEN dis$="CALL VV":RETURN E
LSE dis$="PUSH "+o$(j):RETURN
930 dis$=x$(g)+" V":RETURN
940 dis$="RST "+MID$("0008101820283038
",1+g*2,2):RETURN
950 REM --------- class 1 ---------
960 dis$=MID$("RLCRRCRL RR SLASRA***S
RL",1+g*3,3)+" "+r$(h):RETURN
970 dis$="BIT"+STR$(g)+","+r$(h):RETU
RN
980 dis$="RES"+STR$(g)+","+r$(h):RETU
RN
990 dis$="SET"+STR$(g)+","+r$(h):RETU
RN
1000 REM -------- class 2 ---------
1010 ON h+1 GOTO 1020,1030,1040,1050,
1060,1070,1080,1090
1020 dis$="IN "+r$(g)+",(C)":RETURN
1030 dis$="OUT (C),"+r$(g):RETURN
1040 IF k THEN dis$="ADC HL,"+s$(j):R
ETURN ELSE dis$="SBC HL,"+s$(j):RETUR
N
1050 IF k THEN dis$="LD "+s$(j)+",(VV
)":RETURN ELSE dis$="LD (VV),"+s$(j):
RETURN
1060 dis$="NEG":RETURN
1070 IF k THEN dis$="RETI":RETURN ELS
E dis$="RETN":RETURN
1080 dis$=MID$("IM 0***IM 1IM 2*****
**********",1+g*4,4):RETURN
1090 dis$=MID$("LD I,ALD R,ALD A,ILD
A,RRRD   RLD   ***********",1+g*6,6)
:RETURN
1100 dis$=MID$("LDCPINOT********",1+h
*2,2)+MID$("********I D IRDR",1+g*2,2
):RETURN
1110 REM ---------- Hex dump --------
1120 PEN #1,1:PAPER #1,3:PRINT #1,CHR
$(30);"Addr        Hex  Dump
 ASCII   "
1130 WHILE INKEY$<>" "
1140 PEN 2:PRINT #stream,HEX$(a,4);":
";:PEN 1
1150 char$=""
1160 FOR i=0 TO 7
1170 GOSUB 590
1180 NEXT
1190 PEN 3:PRINT #stream," "char$
1200 WEND
1210 GOTO 210
```

Give your fingers a rest . . .

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

See our special offer on Page 101.

Super teach-in for the tyro businessman

THE majority of businessmen who wish to use the family Amstrad in their company look no further than programs which tackle word processing or simple bookkeeping.

While Entrepreneur, from Triptych, does not address itself to either of these areas, it performs an even more essential service, especially for those who are unsure of the difference between profit and cash, or assets and stock.

The most important figures any organisation needs – for they define the point from which every business decision should start – are what profit margin is required at any given level of sales, in order to generate the desired return on investment.

To the owner of a small company, this last translates into the level of income the proprietor needs to support the desired life-style.

If they only stopped to think about it, they would realise that there is little point in keeping beautiful columns of figures if, when they are added up, they only show a thumping loss has been made.

It is a whole raft of calculations such as these which have to be considered and yet are easily overlooked.

Furthermore, if an "expert" is employed to look after them, whether computer or commercial, this consultant all too often forgets the times when he was still learning his craft.

A common outcome is that the expensive advice passes high over the baffled head of the listener and therefore goes unused.

What I particularly like about Entrepreneur is that it has obviously been designed by persons who fully appreciate these points.

They not only fully understand the commercial sector, but equally importantly they have not forgotten their early days before acquiring this knowledge. In consequence there are two parts to the package.

The first of these is a "teaching program" to lead the user through the essential terms and procedures employed in managing the full range of company finances, not just the cash records.

Even if the user is familiar with business plans, VAT, fixed assets and so on, he would do well to review his knowledge with this program.

For the newcomer it is without doubt the clearest, briefest explanation of the fundamentals of accounts I have seen.

This excellent feature alone is worth the money that is asked for the whole system.

I repeat that the second and main portion of Entrepreneur is not another Debit, Credit, Plonk-type program, but a planning aid.

Rather than recording moneys entering or leaving the company, it acts more like a traditional spreadsheet, but without the major drawback which all spreadsheets possess, particularly so for the less experienced – that when drawing up the matrix, one needs to know in considerable detail the headings for the columns and rows.

Once the basic layout is decided upon, the various formulae need creating.

Thereafter the spreadsheet can churn out figures at will.

If, however, you omit an important column, or your formula is erroneous, then all the figures produced are of dubious worth.

Entrepreneur should be used as a first-class pre-created spreadsheet.

In other words all the essential columns, rows, formulae and layouts dealing with general financial matters are already awaiting data entry, having first made certain you understand the significance of the data requested.

The presentation leads one step by step through the lengthy input required.

If one has any queries, then the comprehensive checklists and clear text tackle these impressively.

It asks for very detailed information on a welter of subjects ranging from the fundamentals such as expenses or initial finance plan to the detailed items within them, that is credit terms or equity fund.

Once this data has been entered, then just as in a spreadsheet, figures may be changed and the whole recalculated as many times as necessary.

I spent a happy hour running a rigorous test of Entrepreneur as a sophisticated "What If" simulation system. This confirmed that it provides all the answers to the essential questions given at the start of this review.

Perhaps the financial director of ICI would find it limited, but if one wants more information then something much larger than an Amstrad is needed to produce the profit and loss accounts or balance sheets.

There are criticisms I could make, but they are all minor, personal niggles rather than misgivings with the software.

An example of this is that all borrowing is assumed to be by banker's overdraft. Other sources could have been taken into consideration.

Also the program appears better suited to the manufacturing or retailing sectors than to the service sector.

Nonetheless I can only conclude that Entrepreneur is superb. Nice one, Triptych. A copy of your product should be in every commercial Amstrad-user's desk.

- *General rating – 9/10.*
- *Value for money – 10/10.*

Jo Stork

GOOD NEWS!

THE CHAOS FACTOR. Our second graphics adventure for the CPC464 is now available. Special price £4.95 for this month only.
Note: **SMUGGLERS COVE** is available through C.R.L.

CREATIVE SOFTWARE FOR THE 464

TOOLBOX. If you don't know about this utility by now, where have you been? Still **£4.95.**

TOOLBOX 2. IT'S HERE! The new music/sound effects toolbox with unique 'edit-by-ear' editor. 3-track sounds, independent editing of rhythm, sound shaping, unlimited sequence lengths, 8-octave range, auto-play mode, full manual. If you can move at least one finger, it's easy! The sounds or music can be used in your own programs. **£5.95.**

GRASP. £8.50. Graphing package.

FLEXIFREND. £7.50. Home accounts.

Cheques/P.O.s to:
CAMEL MICROS
Wellpark, Willeys Ave., EXETER, Devon EX2 8BE

WIN UP TO £10,000 (or more)
Playing
THE ISLAND OF RIDDLES
Adventure Competition

Only £9.50 (inc p&p)
On Tape for Amstrad CPC464

Available by mail order from
REDBEARD COMPUTERS
(Dept B), 45 Regent Place, Birmingham 1
Trade and Overseas enquiries welcome

SIMPLE ACCOUNTS
AMSTRAD CPC464/664

Easy To Use, Comprehensive Book-Keeping System
** Used In Business Throughout The UK **

* Full Analysis of Income & Expenditure
* Powerful Entry Search Routines
* Monthly and Year to Date Reports
* Creditors/Debtors Report
* Quarterly VAT Report
* Printed or Screen Output

Only £29.95 including VAT and Carriage (£34.95 on Disc)

Available from Approved Dealers or Direct from:

CORNIX SOFTWARE LTD
16 Kneesworth Street, Royston, Herts. SG8 5AA
Tel: Royston (0763) 852989
Full Specifications and Sample Output Available on Request

Printers:
CP-80 80cps Fric & Trac **£189.95**
CPA-80 100cps Fric & Trac **£199.95**
Amstrad Printer Cable 6'6" **£7.95**
3" HITACHI Second disc drive complete **£99.95**
Second drive cable only **£7.95**
3" double sided discs for 2 **£8.95** – 10 **£42.95**
Cassettes:
C15's boxed with label & card
3 **£1.15** – 10 **£3.20** – 100 **£30.00**
Dk'tronics Speech Syntheiser **£37.95**
RS232 Serial Interface For CPC464 ... **£43.95**
For CPC664 ... **£53.95**
SIDEWAYS ROM BOX for four
ROM's CPC464 ..... **£29.95**
+ Connector for CPC664 ..... **£9.95**

ROM's
Super Power MAILING LIST ROM .. **£39.95**
Super Power DISK UTILITY ROM **£39.95**
Super Power PROGRAMMERS TOOL ROM **£39.95**
Super Power WORD PROCESSOR ROM **TBA**
Super Power DATABASE ROM ..... **TBA**
(Buy any three ROM's and deduct £5.00)
DISC PAYROLL Contracted in or out (Weekly, 2 or 4 Weekly & Monthly) **£49.95**
CASHBOOK Hundreds of transactions in double entry format with ledger facilities ..... **£11.95** + **£4** on disc
MAILING 250 addresses correctly maintained and printable onto any number of labels ..... **£11.95** + **£4** on disc

Prices include VAT, Postage & Packing add 50p

Micro-Aid (Amstrad), 25 Fore Street, Praze, Camborne, Cornwall. or Tel: 0209 831274 with ACCESS

1ST ON MERSEYSIDE FOR AMSTRAD

6128 & PCW 8256
NOW AVAILABLE!!

HOME & BUSINESS SOFTWARE

RELIABLE MAIL ORDER SERVICE

- Software from all the best suppliers, over the counter
- Books & Accessories
- Games, Education & Programming Aids
- Late night opening Mon-Sat until 6pm
- See it & try it before you buy it

Microbyte

051-630 6933
71, SEAVIEW ROAD WALLASEY

SUNARO software

		FRANK BRUNOS BOXING	7.50		
** STAR **		WAY OF THE EXPLODING FIST	8.50	** STAR **	
CHOICE				**CHOICE**	
JUMP JET	8.50	GREMLINS	8.50	HI-SOFT 'C'	29.95
RED ARROWS	7.50	ALIEN	7.50	THE QUILL	14.95
KNIGHT LORE	8.50	RED MOON	5.95	TASPRINT	8.50
ALIEN 8	8.50	MINDER	8.50	TASWORD	16.95
BINKY	5.25	MORDENS QUEST	5.95	TASCOPY	8.50
ROCKY HORROR SHOW	7.50	MASTER OF THE LAMPS	9.50	FONT 464	6.75
BEACH HEAD	8.50	SCOUT STEPS OUT	7.50	AZIMUTH TAPE ALIGN	7.50
HARD HAT MACK	7.50	CONFUZION	5.95	USE IT	7.50
DUN DURACH	8.50	THE COVENANT	5.95	OPERATION CARETAKER	8.50
HIGGINS POOL	7.50	BRAXX BLUFF	7.50	SCREEN DESIGNER	10.95
HIGGINS SNOOKER	7.50	EVERYONES A WALLY	8.50	CONCISE BASIC	10.95
PSYCHEDLIA	5.25	ERIK THE VIKING	8.50	CONCISE FIRMWARE	17.95
FRANK N STEIN	7.50	THE WILD BUNCH	ANY	MINI OFFICE	5.25
MUTANT MONTY	7.00	MR FREEZE	TWO	DFM DATABASE	21.95
TECHNICIAN TED	6.00	SHORT FUZE	ONLY	MACHINE CODE TUTOR	12.75
WONGS LAUNDRY	7.00	SUB SUNK	4.75	TASWORD DISK	21.95

New titles always available on release
Prices include postage and VAT

SUNARO SOFTWARE (CA)
PO Box 78, Macclesfield, Cheshire SK10 3PF

THE WORM IN PARADISE

Everybody's micro needs magic

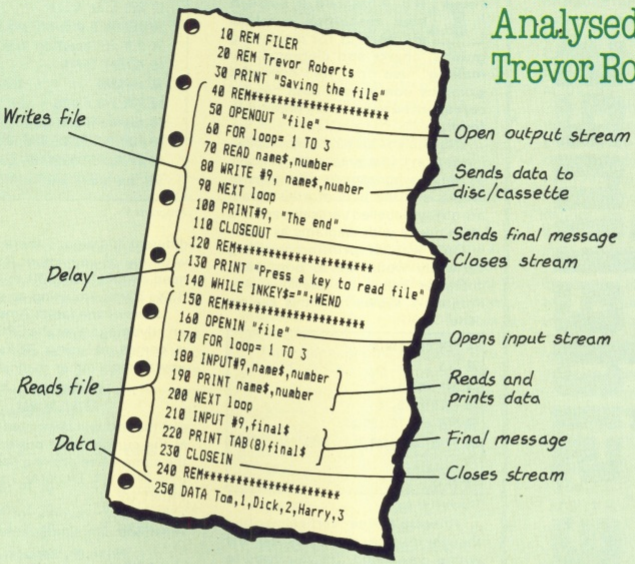
£9.95 on AMSTRAD, ATARI, BBC, CBM, MSX, SPECTRUM etc.

Available from Level 9 Computing, PO Box 39, Weston-super-Mare, Avon BS24 9UR

THIS month's program shows how simple it is to save data to a file on cassette or disc and then read it back.

Filer

Analysed by Trevor Roberts



10, 20 The usual REMs.

30 Prints a message telling you the data is about to be stored. This is needed as disc systems give you no message and you may think nothing is happening. Tape systems, of course, prompt you to press the play and record keys on the recorder.

50 This opens a pathway from the program to the cassette or disc allowing data to be stored in a file. In this case the file is called *file*. Try out other names.

60-90 Make up a FOR . . . NEXT loop which reads in the data and sends it out to the file. When you understand the program try getting the loop to cycle twice or four times and see what happens.

70 Reads in the data, storing it temporarily in the variables *name$* and *number*. In this case the information comes from a data statement, but it could be from other sources such as the keyboard.

80 WRITE#9 causes the information in the following variables to be sent to the cassette or disc. The first time round the loop it will be Tom and 1, the second time, Dick and 2.

90 The loop finishes after three cycles, having read in all the information from the data line. Can you change this to a WHILE . . . WEND loop that reads the information until a flag value occurs?

100 Sends a string to be stored in the output file.

110 Closes down the output stream. The file is written in 2k blocks. If the total amount of data is less than this it doesn't get sent until the CLOSEOUT.

130, 140 Provide a pause between storing the data and reading it. The data is now stored on a cassette or disc file called *file*.

160-230 Read the data back from *file* and display it.

160 Sets up a pathway between the program and the storage device to read from the file called file. If you're using cassette, make sure that you've rewound the tape. What happens if the OPENIN has a different name from the file we OPENedOUT?

170-200 Form another FOR . . . NEXT loop which repeats three times. What happens if it cycles twice or four times?

180 INPUT#9 takes information from the disc or cassette, holding it in *name$* and *number*.

190 Displays the retrieved data.

210 Reads the last part of file and stores it in *final$*.

220 Prints *final$*.

230 Closes the input file.

250 Stores the data. Try changing it.

Set your characters on the sidelines

ROLAND WADDILOVE shows how to manipulate character definitions so that you can display vertical text

THE Amstrad's superb high resolution graphics are perfect for displaying graphs, charts and diagrams. By making use of its powerful graphics commands data can be represented in an easily digestible form.

It is often necessary when displaying data in this way to write vertically as well as horizontally. The axes of graphs and the bars of a histogram are always labelled vertically, and it is often useful with diagrams as well.

One way of printing a string vertically would be to print each character below the previous one. Program I shows how this can be done.

```
10 REM Program I
20 CLS
30 a$="Hello..."
40 LOCATE 10,10
50 FOR i=1 TO LEN(a$)
60 PRINT MID$(a$,i,1);CHR$(10);CHR$(8
);
70 NEXT
```

Program I

This isn't true vertical writing though, as the characters are the wrong way round. Tilt your head to one side and try to read the writing and you'll see what I mean.

It would be nice if there was a command which rotated the characters and printed them vertically for us. Side Writer is a short routine which adds two new commands to Amstrad Basic using RSXs.

One command, |PRINT.UP, rotates the characters of a given string 90 degrees in an anticlockwise direction and prints vertically upwards. The other, |PRINT.DOWN, rotates the characters 90 degrees clockwise and prints them in the opposite direction.

Program II is a short Basic routine to poke the machine code above HIMEM, and Program III is an assembler listing. If you have an assembler you can place the code at a different location if there is a clash with any of your other RSXs. You'll need to CALL &A000 to enable the RSXs if you are using an assembler.

```
10 REM Side Writer
20 REM By R.A.Waddilove
30 REM (c)Computing With The Amstrad
40 MEMORY &9FFF
50 a=&A000
60 FOR i=1 TO 14
70 sum=0:READ a$,b$
80 FOR j=1 TO 21 STEP 2
90 k=VAL("&"+MID$(a$,j,2)):POKE a,k
100 sum=sum+k:a=a+1
```

Program II

Users of the latest Amstrads have an advantage over the CPC464 when it comes to using RSXs. To print upwards on either machine use:

```
a$="Message..."
|PRINT.UP,@A$
```

and the text will be printed up starting at the current print position. It's just the same when printing down, except use |PRINT.DOWN instead of |PRINT.UP.

Similarly, owners of the newer Amstrads can simply enter:

```
|PRINT.UP,"Message..."
```

to get the same effect. This is much easier and closer to the ordinary Basic PRINT command.

The routine starts at &A031 and first picks up the address of the string descriptor from the parameter block pointed to by the IX register. The length of the string and address it is stored at is then collected.

The main loop starts at &A03D and is labelled *another*. Each character in the string is loaded into the A register and the address at which the data making up the character is stored is found by calling the firmware at &BBA5. This may be in the lower ROM, so this is enabled.

A check is made whether the character is to be printed up or down and the appropriate subroutine is

```
110 NEXT
120 IF sum<>VAL("&"+b$) THEN PRINT "E
rror in data...":STOP
130 NEXT
140 CALL &A000
150 PRINT "Side writer set up..."
160 DATA 0109A02124A0C3D1BC11A0,490
170 DATA C328A0C32CA05052494E54,4A7
180 DATA 2E444F57CE5052494E542E,3A1
190 DATA 55D0000000000003E0A1802,187
200 DATA 3E0B3296A0DD6E00DD6601,440
210 DATA 46235E2356EB7EE5C5CDA5,5C5
220 DATA B8CD06B9E00E083A96A0F5,5AD
230 DATA FE0ACC74A0C485A0CD09B9,660
240 DATA 3EFF2197A0CDABBB3EFFCD,6CF
250 DATA 5ABBF1CD5ABB3E08CD5ABB,610
260 DATA C1E12310CAC92197A01A06,4E0
270 DATA 0B17CB1E2310FA130D20F0,365
280 DATA C9219FA01A06082B17CB16,374
290 DATA 10FA130D20F0C9000000000,303
```

called. The two subroutines *up* and *down* rotate the character data left or right. They are both very similar, so if you can follow one you'll be able to cope with the other.

If you look at *up* you'll see that there are two loops. The outer loop runs from byte 0 to byte 8 of the character, loading the data into the A register. The address of the character is in DE, and HL is loaded with the start of the 8 byte workspace reserved for the routine. This is labelled *chr255*.

The inner loop runs from byte 0 to byte 7 of the workspace. The character data in the A register is rotated left each time so that bit 7 ends up in the Carry flag. Then the Carry is rotated right into the corresponding byte of the workspace at *chr255*.

The result of all these rotates is that the bit pattern of the workspace ends up as an exact copy of the character data, except that it has been rotated through 90 degrees.

The rest of the main loop simply defines the workspace as character 255 and prints it on the screen. The print position is then updated.

As you can see, it's quite easy to manipulate character definitions. The two commands should spruce up your graphs quite nicely.

Just for fun, can you alter the routine so that it mirrors text and graphics characters?

```
RAW Assembler V.3

Pass... 2                ORG &A000

A000:01 09 A0            LD BC,comtable
A003:21 24 A0            LD HL,workspace
A006:C3 D1 BC            JP &BCD1

A009:                    .comtable
A009:11 A0               DEFW names
A00B:C3 28 A0            JP printdown
A00E:C3 2C A0            JP printup
A011:                    .names
A011:                    DEF$ 'PRINT.DOW"
A01A:CE                  DEFB &CE
A01B:                    DEF$ 'PRINT.U"
A022:D0                  DEFB &D0
A023:00                  DEFB 0
A024:                    .workspace
A024:                    DEFS 4

A028:                    .printdown
A028:3E 0A               LD A,10
A02A:18 02               JR here
A02C:                    .printup
A02C:3E 0B               LD A,11
A02E:                    .here
A02E:32 96 A0            LD (updown),A
A031:DD 6E 00            LD L,(IX+0)
A034:DD 66 01            LD H,(IX+1)
A037:46                  LD B,(HL)
A038:23                  INC HL
A039:5E                  LD E,(HL)
A03A:23                  INC HL
A03B:56                  LD D,(HL)
A03C:EB                  EX DE,HL
A03D:                    .another
A03D:7E                  LD A,(HL)
A03E:E5                  PUSH HL
A03F:C5                  PUSH BC
A040:CD A5 BB            CALL &BBA5
A043:CD 06 B9            CALL &B906
A046:EB                  EX DE,HL
A047:0E 08               LD C,8
A049:3A 96 A0            LD A,(updown)
A04C:F5                  PUSH AF
A04D:FE 0A               CP 10
A04F:CC 74 A0            CALL Z,up
A052:C4 85 A0            CALL NZ,down
A055:CD 09 B9            CALL &B909
A058:3E FF               LD A,255
A05A:21 97 A0            LD HL,chr255
A05D:CD A8 BB            CALL &BBA8
A060:3E FF               LD A,255
A062:CD 5A BB            CALL &BB5A
A065:F1                  POP AF
A066:CD 5A BB            CALL &BB5A
A069:3E 00               LD A,0
A06B:CD 5A BB            CALL &BB5A
A06E:C1                  POP BC
A06F:E1                  POP HL
A070:23                  INC HL
A071:10 CA               DJNZ another
A073:C9                  RET

A074:                    .up
A074:21 97 A0            LD HL,chr255
A077:1A                  LD A,(DE)
A078:06 08               LD B,8
A07A:                    .uloop
A07A:17                  RLA
A07B:CB 1E               RR (HL)
A07D:23                  INC HL
A07E:10 FA               DJNZ uloop
A080:13                  INC DE
A081:0D                  DEC C
A082:20 F0               JR NZ,up
A084:C9                  RET

A085:                    .down
A085:21 9F A0            LD HL,newchar
A088:1A                  LD A,(DE)
A089:06 08               LD B,8
A08B:                    .dloop
A08B:2B                  DEC HL
A08C:17                  RLA
A08D:CB 16               RL (HL)
A08F:10 FA               DJNZ dloop
A091:13                  INC DE
A092:0D                  DEC C
A093:20 F0               JR NZ,down
A095:C9                  RET

A096:                    .updown
A096:00                  DEFB 0
A097:                    .chr255
A097:                    DEFS 8
A09F:                    .newchar
A09F:                    END
```

Program III

PRINTER BARGAINS

PRICES INCLUDE VAT, CABLES AND NEXT DAY DELIVERY BY SECURICOR

NEW SEIKOSHA PRINTER SP/1000A ..... ONLY £219

* 100 characters per second draft mode.
* 25cps in Near Letter Quality mode.
* Friction and tractor feed standard.
* Subscript and superscript.
* Bold and underscore.
* Enlarged.
* Condensed

TAXAN KP810 ..... ONLY £279

* 160 characters per second. 80 column.
* 27cps in Near Letter Quality mode (23 x 18 matrix).
* Friction and tractor feed standard.
* Subscript and Superscript.
* Bold and underscore.
* Enlarged.
* Condensed.

QUEN DATA DAISYWHEEL ..... ONLY £239

* 18 characters per second 14 inch platen.
* Friction feed Sheet feeder option £219.
* Bold and shadow printing.
* Underscore.

Printer cable £12

2000 sheets fanfold paper (supplied with printer) £16

Tasword 464 ..... £16 **Tascopy 464 ..... £9**

Tasword 464-D . £22 **Tasprint 464 ..... £9**

Amstrad 6128 with monochrome monitor £299

Amstrad 6128 with colour monitor £399

PHONE (0267) 231246 FOR PERSONAL ASSISTANCE!!!

STRONG COMPUTER SYSTEMS

BRYN COTTAGE, PENIEL, CARMARTHEN, DYFED, SA32 7DJ

LOAD &RUN

Send your Cheques/POs (Sterling only) or stamp and address for list to:
LOAD & RUN, DEPT, CWA P.O. BOX 123 DARTFORD DA1 5AA

RING (0322) 73128 FOR ENQUIRIES

OUR POLICY FOR 1985 IS TO PROVIDE SOFTWARE THAT EVERYONE CAN AFFORD.

AMSOFT 3" DISC'S BOX OF 10 STILL ONLY £37.95

CASSETTE BASED GAMES

Item	RRP	Our Price
3D Stunt Rider	£8.95	£7.95
Brian Jacks Superstar Challenge	£7.95	£6.95
Battle Beyond the Stars	£8.95	£7.95
Braxx Bluff	£8.95	£7.95
Brian Bloodaxe	£8.95	£7.95
Codename Mat II	£8.95	£7.95
Dragons	£8.95	£7.95
Frank Bruno's Boxing	£8.95	£7.95
Fu Kung in Las Vegas	£8.95	£7.95
Macadam Bumper	£7.95	£6.95
Project Future	£8.95	£7.95
Robin of Sherwood	£9.95	£8.95
Rocco	£7.95	£6.95
Scrabble	£9.95	£8.95
Starion	£9.95	£8.95
Subterranean Stryker	£8.95	£7.95
View to a Kill	£10.99	£9.95

CASSETTE BASED – ADVENTURE & STRATEGY

Item	RRP	Our Price
Ashkeron	£6.95	£5.95
Warlord	£6.00	£4.95
Wizards Lair	£8.95	£7.95

DISC BASED – ADVENTURE & STRATEGY

Item	RRP	Our Price
Fighter Pilot	£14.95	£12.95

CASSETTE BASED – UTILITIES

Item	RRP	Our Price
Money Manager	£14.95	£13.95
Use It	£8.95	£7.95

DISC BASED – UTILITIES

Item	RRP	Our Price
Money Manager	£17.95	£16.95

"As we do not advertise software we haven't actually seen, theres bound to be dozens of new items by the time this is published. So ring now for the very latest !!!"
*** MEMBERS OF BESA ***

UK: All prices include VAT and P&P in U.K. Add P&P in brackets for hardware.
For European & Overseas: All prices exclude VAT & include P&P in Europe. Elsewhere add 50p extra for P&P hardware outside Europe add £2.50.

THIS IS ONLY A SELECTION OF NEW RELEASES AND RECENT ADDITIONS TO OUR STOCK. PLEASE RING FOR LATEST RELEASES
E.&.O.E.

MONEY MANAGER

"Quite the best and most flexible personal accounts system I've seen – also an inexpensive business system" – C.W.T.A. September 1985.

for the

AMSTRAD CPC464/664/6128

Money Manager is an easy-to-use system for recording all financial transactions as a matter of record (unlike most 'budgeting' systems), and for analysing them in a number of very powerful ways in order to facilitate sound financial management. It is ideal for controlling the finances of a small business, or for users wishing to control their personal finances in a businesslike way.

UNRIVALLED RANGE OF FEATURES:

- 50 user-defined income/expenditure classes, in logical groups.
- 9 user-defined accounts (bank, cash, credit card etc.)
- 12 months of data in memory.
- Automatic updating of starting balances when advancing months.
- 100 entries per month.
- Validated entry of date, account, reference, class, descriptive text, amount, user-defined flag.
- Easy editing of entries (also validated).
- Entries optionally sorted e.g. by date.
- Full selectivity of report categories by account, class and flag.
- Detailed statements for selected months.
- Summary tables (class vs month) for whole year, with totals.
- Summary tables (class vs account) for whole year, with totals.
- Bar graphs (amount vs month) of 4 categories simultaneously.
- Pie charts (amount per year) of 10 categories simultaneously.
- Global search for any string.
- User-selected colours/shades.
- Drives any 80-column printer.
- Realistic set of demonstration/practice data.
- Comprehensive 12-page manual.

Cassette: ***£14.95*** *Disc:* ***£17.95*** *(incl. VAT, p&p)*

Connect Systems

3 Flanchford Road, London W12 9ND

Hi-res graphics aren't always essential..

I READ the letter from Graham Stride in the August, 1985 issue of Computing with the Amstrad, *and it made me think of a common misconception about graphics programming.*

It is not necessary when writing Basic games programs always to use high resolution graphics. Respectable programming for simple games can be done in character graphics, using LOCATE and PRINT for movement.

The advantage in this for those learning programming is that it is easier to keep track of and control 1,000 character positions on the screen (Mode 1) than over 50,000 pixel positions in high resolution graphics.

Of course you are limited in the scope of your game. My game Mouse in the May issue only used character graphics – and that was of a high enough standard to be published by you.

Unfortunately Amstrad Basic does not provide a keyword in character graphics that is equivalent to TEST in high resolution graphics, that could be used to detect collisions. What is needed is a simple machine code routine that will return the Ascii value of the location of the text cursor on the screen.

```
10 CLS:GOSUB 140:REM m/c ro
utine
20 REM *** demo ***
30 REM
40 LOCATE 1,1:INPUT "Enter
a string ";a$
50 CLS:ch=0:LOCATE 20,13:PR
INT a$
60 LOCATE 20,13:GOSUB 110
70 LOCATE 1,13: PRINT ch
80 GOTO 40
90 REM call routine to read
 char
100 REM
110 CALL &88CD:ch=PEEK(&88C
C):RETURN
120 REM m/c routine
130 REM
140 MEMORY &88CB:RESTORE 15
0:FOR n=0 TO 13:READ A$:POK
E (&88CC+n),VAL("&"+a$):NEX
T:RETURN
150 DATA 00,3E,F4,32,CC,88,
CD,60,BB,D0,32,CC,88,C9
```

Program I

Program I is a routine that will do this. The machine code routine must be POKEd into the memory at the beginning of the program and then is CALLed up when required.

The Ascii value of the cursor position is at &88CC and can be PEEKed after being called up.

Try changing the location in line 60 and you should find 32 will be returned from the routine.

Graham also asked about printing multicoloured characters using Basic. This can be done quite easily using control codes. Program II is a demonstration routine that prints several pink faces with blue eyes and a red mouth.

Each part of the character must be defined using SYMBOL and the whole character defined, which can be quite complex (face$in this case). From then on your string character can be used wherever you want.

If you analyse line 70 and refer to the User Instructions, you will see how the face is made. – **Graham Redman, Sittingbourne, Kent.**

■ Thanks for the useful contribution. CPC664 owners of course will not need your first routine as their machine has a specific command COPYCHR$ to achieve the same result.

```
10 REM INITIALISE
20 INK 0,0:INK 1,16:INK 2,2
:INK 3,6
30 SYMBOL 241,0,0,102,0,0,0
,0,0:REM EYES
40 SYMBOL 242,0,0,0,0,66,0,
0:REM MOUTH
50 REM DEFINE CHAR
60 FOR n=1 TO 15:READ a:fac
e$=face$+CHR$(a):NEXT
70 DATA 15,1,224,15,2,8,22,
1,241,15,3,8,242,22,0
80 REM PRINT CHAR
90 MODE 1: FOR n=1 TO 20 ST
EP 2:LOCATE n,n:PRINT face$
:NEXT
```

Program II

Line feed cure

IN your issues 1 and 4, January and April, 1985, suggestions were offered to overcome unwanted line feeds when using Epson FX80 and FX100 printers. These suggestions ranged from cutting the interface cable at wire 14 to amending the printer command using #8.

I found the first suggestion a bit too drastic and could not get any result from the second.

A rather more satisfactory suggestion when using Mastercalc or Masterfile is to go into Basic through the menu choice and enter without using any line number POKE 899,0. Then return to main menu using GOTO 100. – **M.D. Constantine, Canterbury, Kent.**

Editor alterations

I HAVE a problem with the Text Editor that you published in your February issue. Although the program itself works fine. I seem unable to make certain alterations which I would like.

I am studying French and since my printer (SC 200) supports a French character set, I thought it would be better to get the computer to show French characters on screen instead of the undecipherable {,\,@ and so on which come out in French on my printer as e acute, c cedilla and a grave.

So, using a character definer I have written, I worked out the relevant data for the SYMBOL command. I have included an option to have the French on screen or not, in case I am writing in English:

```
25 SYMBOL AFTER 64
65 INPUT"Do you want a
French char set (y/n)
";ans$
66 IF LOWER$(ans$)="y" THEN
GOSUB 3000
```

Line 3000 is the routine to redefine six characters to their French equivalent. However, the program refuses to run like this giving either Improper Argument or just redefining all the letter characters to a bar.

Obviously the problem is in the code used and so I have tried to relocate this, changing all the calls, but this makes no difference at all. I do hope you can help with this annoying problem, as I would find a solution very helpful indeed. – **A.J. Carless, Worcester Park, Surrey.**

■ Unfortunately SYMBOL AFTER 64 takes up too much memory so Texted will not run without major modification to the machine code routines.

I HAVE been following Mike Bibby's articles in your magazine since they started and have found them very well written, easy to follow with difficult concepts beautifully explained.

However, I did get into trouble with Part VIII which appeared in the September 1985 issue. Let me explain.

Program VI, with address 3006 successively changed from 1 to 2, 3, 4, 5, 6 and so on, produces as expected a dep-

leted selection of Ascii characters from &20 to &FF.

Program VII, on the other hand, which introduces the comparison instruction CP, behaves very differently.

When address 3003 is successively changed from 1 to 2, 4, 6, 8, 10 all is OK, but 12 produces a pulsating screen display with some of the Ascii characters in a window, together with a bleep. The program crashes.

Likewise, odd numbers 3, 5 and 9 produce odd results. With 3 the program just crashes, 5 causes a limited display of Ascii characters but the program crashes because there is no way back to the menu.

For some reason ADD A,7 behaves normally producing a limited number of characters. ADD A,9 again produces spectacular results with a flashing border and the program crashes.

There is no hint as to why this should happen in the text, so please can you suggest the reason?

Incidentally, I loaded these programs using Hexer from the March 1985 issue. I have noticed that there seems to be something strange when Examine or Alter options are selected. The address appears as a mixture of hex and decimal, for example 3010 rather than 300A, 3012 instead of 300C, and so on. Please can you help? – **J.D. Butler, Sutton Coldfield.**

■ The problem arises because you're modifying the programs to do something they weren't intended to do. Mind you, that's the best way to learn machine code.

Program VI, as you can see, continually adds one to the

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

contents of the A register. Then if Carry isn't set it branches back to 'print it out'.

The point is that Carry is set when the number in the A register is increased from 255 (&FF) – when it cycles round to 0.

By altering &3006 you're altering the number added to the A register, so this will increase in steps. However, as soon as it goes past 255, Carry is set and you drop out of the loop.

Program VII, though, isn't

```
address  hex code  anemonics
3000     3E 1F     LD A,&1F
3002     C6 01     ADD A,1
3004     CD 5A BB  CALL CharOut
3007     FE FF     CP &FF
3009     DA 02 30  JP C,&3002
300C     C9        RET
```

testing for anything as general as going past 256. It's waiting for the A register to be exactly 255 by using CP &FF.

Since A starts at &1F then goes up in ones, the only time CP &FF will not set the Carry flag (which implies that A is less than &FF) is when A reaches &FF. We then drop out of the loop.

If you change the contents of location &3003, you're again changing what's added to A each time. Unless you pick a number that reaches &FF exactly the loop will continue cycling past zero, but this time A will be less than 32 – so you'll be printing the dreaded control codes which cause your micro to go beserk.

That's why we've always made the first character we print 32 – to avoid this problem.

If I were you I'd check your version of Hexer, there's probably a typing error in there. Our listing is correct.

Is there anyone else with machine code queries? It's about time we put Mike Bibby to some real use . . .

Rotating ellipses

IN the August 1985 edition of Computing with the Amstrad, *Page 29, Figure 1 shows a rotated ellipse.*

Unfortunately the formula used in the six Basic programs will only draw upright ellipses. That is a pity as the formula for drawing rotated ellipses is no more complicated than the one used. The following program illustrates it:

```
10 INK 0,0:INK 1,26:MODE 2:
LOCATE 1,4:PRINT STRING$(80
,"-"):DEG
20 WHILE 1:PRINT CHR$(30);
21 INPUT "Rotation angle ",
r:REM may be negative
25 INPUT "Short radius",a
30 INPUT "Long radius",b:RE
M a>b allowed
35 ORIGIN 320,170,0,639,340
,0
40 PLOT a * COS(r), a * SIN
(r),1
45 FOR t=0 TO 360 STEP 2:RE
M higher steps for speed
50 DRAW a*COS(t)*COS(r)-b*S
IN(t)*SIN(r),a*COS(t)*SIN(r
)+b*SIN(t)*COS(r)
55 NEXT:PRINT CHR$(19);:WEN
D
```

A rotation of zero degrees gives r=0. It is easily verified that the formula can be reduced to the formula for an upright ellipse because SIN(0)=0 and COS(0)=1.

May I also draw your attention to a bug residing in the program Dedit at line number 750. The value returned by statement:

```
.....INSTR("",+V$,UPPER$((
$$))
```

should be smaller than 1 instead of 2, thus allowing the user to edit zeros in the sector bytes.

If programs like Ches Jeske's Dedit keep appearing in your magazine, I guarantee it a lifelong success. It's a sign that you regard us readers as adult and serious users who do not like to be put off with some fair promises. Thanks. – **Patrick De Geest, Wezembeek-Oppem, Belgium.**

■ Our readers should find your routine very useful. And thanks for pointing out the bug in Dedit. Somehow a control character found its way between the quotation marks which was ignored by the printer. The line should end as follows:

```
.....INSTR(CHR$(4),+V$,UPP
ER$((S$))
```

Trapping disc errors

I OWN a CPC464 with disc drive and I am trying to write a disc based database for my computer studies project. The problem is that when I get disc errors like the ones listed below my program immediately stops and has to be re-run:

Disc Full
File not Found

I am quite aware that the CPC664 has commands that enable programs to continue after one of the above disc errors have occurred but is there any way that the CPC464 can mimic the CPC664 in this respect?

I also have a better solution to disable Break than the one published in your July edition. If you type POKE –16914,201 into your program then it will still print:

BREAK

when you are in direct mode, but as soon as you execute the program you will not be able to stop or break out of the program. Nor will you be able to reset the computer with Ctrl + Shift + Esc.

Lastly, may I just say how

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

much I enjoy reading your magazine and I particularly like the Postbag pages. – **B. Gibson, Waterlooville, Hants.**

■ To trap disc errors you'll have to interrupt the Amsdos routines by replacing the jump block entries with jumps to your own routines. It's too complicated to go into here, Chapter 5 in the disc manual gives you the information you need.

As for Postbag, it's one of the most enjoyable features to put together – thanks to all the interesting mail we receive from our enthusiastic readers.

Please remember, though, if you want a personal reply, enclose an SAE.

Exit the inputs

ANOTHER beginner begging assistance. I know that I can buy a commercial piece of software to handle my problem, but I want pleasure as well as utility, so . . .

My program, obviously naive, inputs sales of several product lines and purchases of same, applies my equation for mark up, calculates, total sales, stock balance, and gross margin. The program "saves" OK but I lose all my inputs and results. – **E. Nelson, Darlington.**

■ All your data will be lost when you save your program, so save the data as a separate file. Then when you come to use your program again, load the data back in.

Have a look at the text editor in the February issue to see how it's done. Alternatively, you might find our reply to Jean Allman's letter useful – it's on the right of this page.

We'll also be covering the whole area of saving files in a forthcoming series.

CLEAR cleans up

I AM not normally given to typing in magazine listings but I have had a certain amount of success with most of your games, once I have sorted out my typing errors.

However one game that had me stumped for a while was Drogna from the September issue. The high score table doesn't work.

The reason is that Mr Turner at line 170 has used the CLEAR command to clear the contents of the arrays. Unfortunately he overlooked that this also resets the high score variable h% to zero.

I found a way round this by storing the value of h% in memory while the CLEAR is executed, bringing it back again when needed. This is achieved successfully with the addition of the following lines:

```
2785 h%=PEEK(&8000)+PEEK
(&8001)*256
2855 POKE(&8000),h% MOD
256: POKE (&8001),h% \ 256
```

Apart from this minor error the game runs perfectly and has kept my two youngsters amused for hours.

Keep up the excellent work, you seem to have found the recipe for success. – **John Riley, Somercotes, shire.**

Easy with Easy Draw

I AM writing to you to comment on the Easy Draw program featured in the June, Computing with the Amstrad.

I was put off somewhat by the length of the program and subsequently the risk of errors in typing it out. I therefore sent for your tape which contains an error free copy of Easy Draw.

I strongly recommend all readers to send for the tape. Not only having to type in and subsequently de-bug the program but also for the other programs it contains. And such good value too!

I have enclosed my latest effort which is based on a magazine picture. I feel the photos of the screen pictures which accompany the program in issue 6 do not do justice to the kind of pictures you are able to produce with this program. – **A.J. Steel, Yeovil, Somerset.**

■ We must compliment you on your artwork and share your pleasure in such an excellent program.

Tip for Al

ALTHOUGH I am a reasonably experienced programmer I still read Al's Beat *as it helps me when I teach my friends Basic programming, something I find I am doing rather a lot recently.*

However to the point. With regards to your Smiley Hunt program in the August, 1985 issue of Computing with the Amstrad, *you make reference to the difficulty found printing the numbers at the top of the grid. Well you seem to have used a large sledgehammer to crack a small nut.*

I have enclosed my version of printing the numbers. Line 740 is simplicity itself, no need for the string fiddling you have employed. Line 750 uses a little used function of WRITE which will not place any space before it, so the grid looks more uniform and tidier.

Anyway keep up the good work. – **Adrian Steel, Yeovil, Somerset.**

```
740 LOCATE #0,5,5:PRINT
#0,"123456789"
750 FOR y=6 TO 15:LOCATE
15,y:WRITE y-6:NEXT
```

■ Thanks Adrian. I never even thought of your line 740 as a solution, and although I knew the WRITE command existed, I never realised what it could do. – Al.

Setting up data files

HAVING just recently bought an Amstrad CPC464, I have not been able to find any detailed information on the setting-up and re-reading of cassette data files.

Having (I assume) written data via fixed variables to tape I am unable to read the data via another program with the same variables. Trying to set up some sort of stock control system has thus proved difficult!

Any advice or help on the matter would be greatly appreciated as the biorhythm program from the July, 1985 issue of Computing with the Amstrad, *shows me to be totally out of steam!* – **Jean Allman, Southsea, Hants.**

■ This short program should be just what you require. It allows you to input 10 numbers and then prints them out on screen. You will have to alter it, of course, to suit your own needs.

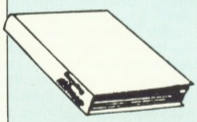
```
10 REM Save Data
20 OPENOUT "data"
30 FOR i=1 TO 10
40 INPUT "Number";n
50 WRITE #9,n
60 NEXT
70 CLOSEOUT
80 REM Read Data
90 OPENIN "data"
100 FOR i=1 TO 10
110 INPUT #9,n
120 PRINT "Number i was";n
130 NEXT
140 CLOSEIN
```

Exclusive offer to subscribers!

It's been given rave reviews in Computing with the Amstrad and other magazines. Everyone who's played it says its one of the most gripping arcade games they've ever experienced. It's already taken America by storm. Now it's doing the same in Britain. It's called . . .

BEACH-HEAD

– and it's a game no Amstrad owner will want to be without! It's in the shops now and costs **£9.95** on cassette. But we'll send it to you for just **£3** if you take out a subscription on the form opposite. *(Not available without subscription.)*

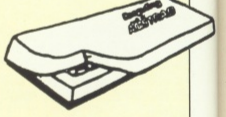


Binder: £3.95 (UK)

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.

Dust cover: £3.95

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.



GIVE YOUR FINGERS A REST!

JANUARY: ***Smiley:*** *Guide Smiley avoiding Grumpies.* ***Code:*** *Be a master-mind.* ***Binary:*** *Baffled by binary bits? Let our utility help you out.* ***Dancer:*** *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.* ***Origins:*** *Changing the graphics origin.*

MARCH: ***Starfleet:*** *Can you qualify as a starfleet pilot?* ***Swamp:*** *Join the battle against giant frogs.* ***Parachute:*** *Parachute to earth.* ***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:*** *A CAL language.* ***Spelling:*** *Learn to spell.* ***Aleatoire:*** *The odd ways mariners make decisions.* ***Palindrome Tester:*** *Slick string handling techniques.*

MAY: ***Tronn Cycles:*** *Two player arcade classic.* ***Castle of Fear:*** *A baffling fantasy adventure.* ***Mouse:*** *A simple but addictive game.* ***Cedric:*** *Shape recognition.* ***Character Maker:*** *Define your own characters.* ***RSX:*** *New commands for your micro.* ***Cards:*** *An apparently simple computer card trick.* ***Trees:*** *A clever routine to create 'trees'.*

JUNE: ***Easydraw:*** *A utility to create and manipulate your own pictures.* ***Da Bells:*** *Hilarious arcade action.* ***AMmon:*** *A superb machine code monitor.* ***Menu:*** *Create colourful menus.* ***Nim:*** *Baffle your friends.* ***Quads:*** *Display expanding quadrilaterals.*

JULY: ***Missile Command:*** *Save earth from the invading hordes.* ***Jumbo's Wordgame:*** *Our Amstrad version of Odd One Out.* ***Biorhythms:*** *Find out if you're going to have "one of those days".*

Computing with the AMSTRAD

MAIL ORDER OFFERS

ORDER FORM

All prices include postage, packing and VAT
Overseas rate stated as Surface Mail. Air Mail price on application.

Valid to November 30
Please enter number required in box. £ p

Annual subscription
UK & Eire £12 (Sterling only) 6001
Overseas (Surface) £20 6003
Overseas (Airmail) £40 6004

Add £3 to your subscription and receive your cassette of Beach Head (worth £9.95). **Only available if accompanied by subscription order.** 6058

Commence with _____ issue

TOTAL

AMX Mouse
SPECIAL OFFER!

£69.95 6060

See Page 31

TOTAL

Tapes
£3.75 UK
£4.75 Europe/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
August (Ah Bod!) 6027
September (Robot Ron) 6028
October (Le Mans) 6029
November (Space Trucker) 6030

TOTAL

3" Discs
£6.75 UK
£7.75 Europe/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
August (Ah Bod!) 6067
September (Robot Ron) 6068
October (Le Mans) 6069
November (Space Trucker) 6070

TOTAL

Back issues
£1.25 UK
£1.50 Overseas (Surface)

January 6006 N/A
February 6007
March 6008
April 6009
May 6010
June 6011
July 6012
August 6013
Sept 6014
Oct 6015

TOTAL

Binder £3.95 UK, £6.95 Europe/Overseas 6038

TOTAL

Dust Cover £3.95 UK, £4.50 Europe, £4.95 Overseas 6037

TOTAL

Red Arrows
Tape £8.95 UK £9.95 Overseas/Europe 6050
Disc £12.95 UK £13.95 Overseas/Europe 6051

TOTAL

dk'tronics Light Pen £24.95 (UK & Overseas) 6057

TOTAL

dk'tronics speech synthesiser £34.95 (UK & Overseas) 6040

TOTAL

Mini Office
Europe/Overseas add £1 per order
Tape £5.95 UK 6035
3" Disc (CPC464) £9.95 UK 6036
3" Disc (CPC664) £9.95 UK 6059

TOTAL

All the program listings from each month's issue are available for only **£3.75** *on cassette and* **£6.75** *on disc.*

Raw: *Superb machine code assembler.* ***Compacter:*** *Reduce the memory used while saving a screen.*
AUGUST: ***Ah Bod:*** *Collect the eggs and avoid the evil Stompers.* ***Dedit:*** *Edit disc sectors and recover lost programs.* ***Polygons:*** *Create and display superb rotating shapes.* ***Events:*** *Demonstrations of interrupt routines.* ***Password:*** *Generate your own password.* ***Graphics:*** *A character generator and demonstration programs.* ***M/code graphics:*** *Routines to examine the screen and pixel patterns.*
SEPTEMBER: ***Robot Ron:*** *Guide our hero in his quest to rid the screen of the evil weevils.* ***Drogna:*** *Can you survive on the planet Arg.* ***Magnified Characters:*** *A superb enhancement to EasyDraw. Create and manipulate your own colourful text screens.* ***Smiley Hunt:*** *Search for those happy smiling faces in Al's mega-game.* ***Graphics:*** *All the listings from this month's graphics article.*
OCTOBER: ***Le Mans:*** *Exciting 3D car racing spectacular. Be first across the line in this simulation of the 24 hour race.*
Pontoon: *Win a fortune or lose your shirt in this superb version.* ***Bellringers:*** *Take your lightpen bellringing with this entertaining music utility.* ***Pi:*** *A program to calculate PI and leave the first mainframe's attempts standing.* ***M/code Graphics:*** *A sprite designer and print routine.* ***Graphics:*** *All the listings from this month's graphics article.*
NOVEMBER: ***Frutties:*** *Rid the factory of the fearsome monsters in this fast action arcade game.* ***Space Trucker:*** *Be an intergalactic Yorkie as you take charge of the shuttle on its dangerous mission.* ***Alphabet Train:*** *Spelling and word recognition made easy.* ***Trace:*** *A superb utility to augment the Amstrad's TRON command.* ***Disassembler:*** *An invaluable addition to your machine code toolkit.* ***Subhunt:*** *The finished version of Al's game.* ***Sidewriter:*** *A utility to turn text on its side.* ***Bluff:*** *A testing little card game from Aleatoire.* ***Filer:*** *How to save and load files to cassette or disc.* ***PLUS*** *all the listings from this month's graphics article.*

Classic Games on the Amstrad
Tape £5.95 UK £6.95 Overseas/Europe 6048
Disc £9.95 UK £10.95 Overseas/Europe 6049

TOTAL

Payment: please indicate method (✓) **TOTAL** _____

☐ Access/Mastercharge/Eurocard
Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Barclaycard/Visa
Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Expiry date / /

☐ 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

Don't forget to quote your credit card number and full address CA/11

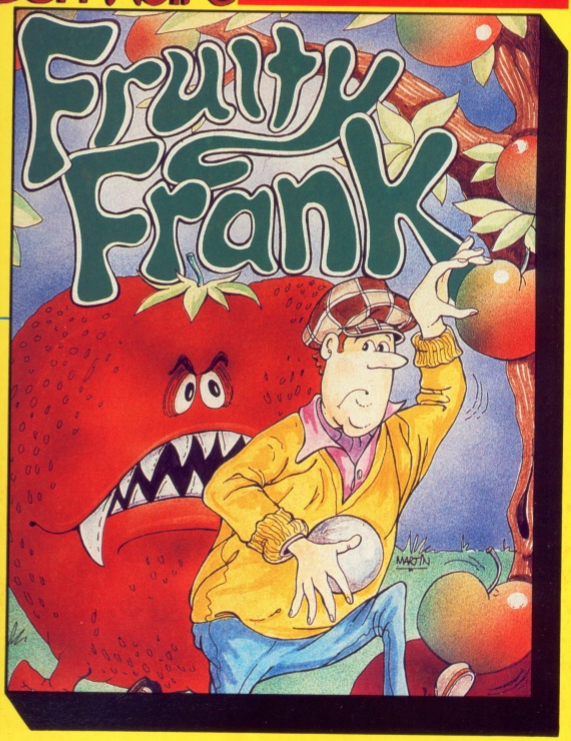
ADVERTISERS INDEX

Al-Tukhaim	102	Microdis	86
Arnor	42, 43	Micropower	89
Amstra-Draw	57	Microaid	92
Camsoft	102	Microbyte	92
Camel Micros	92	Northern Computers	102
Cascade	18	Picturesque	57
Campbell Systems	26	Pride Utilities	36
Cornix Software	92	Phoenix Publishing	69
Connect Systems	96	Printerland	86
Datastar Systems	80	Qualsoft	47
Dart Electronics	24	Redbeard Computers	92
Design Design	3	Saphire Software	57
DK'Tronics	104	Siven Software	15
The Electric Studio	2, 90	Superpower	34, 35
Evesham Micros	86	Simtron	69
Haystack Peripherals	20	Strong Computer	96
Interlock	69	Shekhana	69
K.D.S. Electronics	36	Software Design	81
Kosmos	48	S.J.B.	81
Kuma Computers	103	Selec Software	90
Level 9	92	Sunaro	92
Load & Run	96	Transform	28
Lothlorien	81	Timatic Systems	48
M&A Software	86	U.S. Gold	16
Microsoft	27	Vortex	8
Micro-byte	81		

Kuma AMSTRAD CPC464 software

the first choice

FRUITY FRANK – *Your fruit garden has been invaded by marauding monsters. Pick the fruit in the garden while avoiding the pursuing monsters. Push apples onto the monsters and as a last resort, throw your ball at them to kill them. Fill in the hole in the ground to stop the monsters from coming out. Watch out for dropping "plum monsters" for bonus points. High score table, choice of three speeds, demonstration game and musical accompaniment. Written extensively in machine code for the higher level of performance.*



Just one of our rapidly expanding range of Entertainment and Application Software for the Amstrad micro-computer .

Other titles include:

Star Avenger, Galaxia, Artwork, Rock Raid, Zen Assembler, Music Maestro, North Sea Bullion, Stock Control, Datafile II, Shadow of the Bear.

Books:

- ***THE AMSTRAD CPC 464 EXPLORED*** ***by John Braga***
- ***ZEN AND THE AMSTRAD CPC 464*** ***by Ian R. Sinclair***

Visitors wishing to call at our Pangbourne Manufacturing and Distribution Centre are advised to phone 07357-4335 first for an early appointment.

Kuma Computers Ltd., Unit 12, Horseshoe Park, Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad products.

Name .....

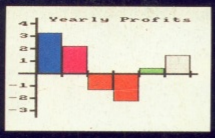
Address .....

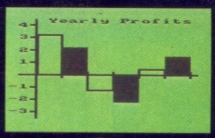
.....

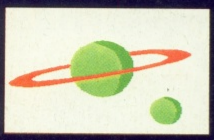
..... Phone.....

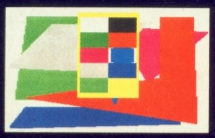
I own an Amstrad CPC 464 computer ☐

Trade Enquiries Phone 07357-4335

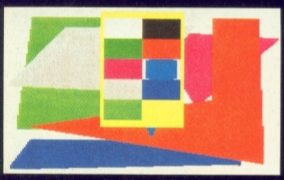






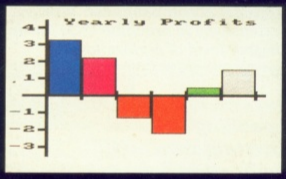


RRP £24·95

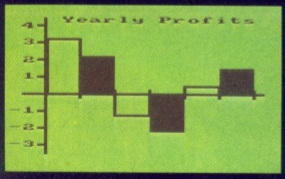


CHANGE COLOUR

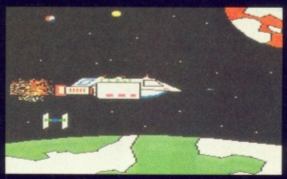
CHANGE BRUSH



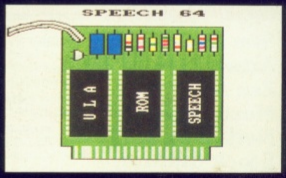
ADD TEXT



GREEN SCREEN COMPATIBLE



HIGH-RES COLOUR GRAPHICS



TECHNICAL DRAWING

Compatible with: GT64 • CTM640 • MP1 • DDI 1

Sophisticated graphics package on tape includes:

Colour palette • 'nudge' control for one pixel accuracy • brush choice • text handling • user defined characters • magnify • shrink • circles • rectangles • lines • curves • colour fill • tape & disc • picture storage & retrieval • pen calibration utility • printer dump

GRAPHICS LIGHTPEN

For AMSTRAD CPC464

dk'tronics

Colour box artwork produced using lightpen, Amstrad CPC464 & colour monitor

Saffron Walden, Essex CB11 3AQ
Tel: (0799) 26350 10 lines