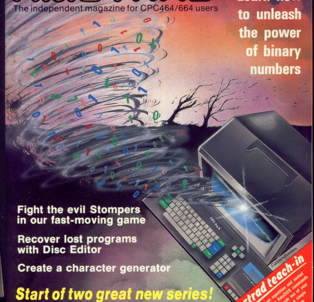
A Database Publication

Computing with the AMSTRAD

No. 8
August 1985
£1

The independent magazine for CPC464/664 users

EXCLUSIVE!
Graphics Light Pen and full software package – special offer to readers



Learn how to unleash the power of binary numbers

Fight the evil Stompers in our fast-moving game

Recover lost programs with Disc Editor

Create a character generator

Start of two great new series!

Explore the weird and wonderful world of adventures
Unravel the mysteries of machine code graphics

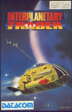
Amstrad teach-in

Learn all about repetition and nested loops ● Guide to AmsDOS commands ● Coax more sounds out of your Amstrad ● Write a simple game ● Conjure up fantastic graphics displays

WELCOME TO THE WORLD OF DATACOM

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

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



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

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

Trade enquiries
welcome

All games just

£5.95

Each

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

* U.K. only - Overseas add £1 p & p

Name ....
Address ....
.... Tel: ....
Please rush me
□ Tapes of Interplanetary Trader
□ Tapes of Execution
□ Tapes of Snail Pace
I enclose a cheque/P.O. for
£ ....

NEW!!

RED MOON MoonQuest fantasy adventure for most micros. The Moon Crystal of Xax supplied good magik for the honest rulers of the kingdom of Baskalos. Then it was stolen and the land fell into chaos. You must recover the Moon Crystal from the magik-rich castle where it is hidden, armed only with your wits and spells.

RED MOON‡ is a Level 9 adventure with over 200 locations. Amstrad, BBC, Commodore 64, MSX and Spectrum versions have 150+ pictures.

RED MOON is available now on cassette at *£6.95.* It joins Level 9s catalogue of acclaimed adventures for

AMSTRAD cpc464

ATARI 48k

BBC 32k

COMMODORE 64

ENTERPRISE 64

MEMOTECH 500/512

MSX 64k

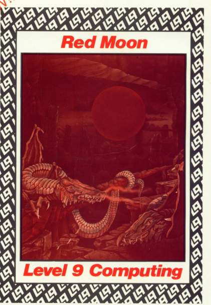
SPECTRUM 48k

Here's what some reviewers have recently said about Level 9.

"the series of adventures which has (since) flowed from Level 9 is renowned for high standards of plot and literate description".
-Sinclair User, May 1985

"I'll be kept more than busy by the absorbing Level 9 adventures".
-MSX User, June 1985

"If only more adventure houses could achieve Level 9".
-Crash, May 1985



So don't forget our other great adventures:
COLOSSAL ADVENTURE*† *(cass £9.95)* The classic mainframe game with 70 bonus rooms.
ADVENTURE QUEST*† *(cass £9.95)* "A brilliant adventure that confirms Level 9's place as the ultimate adventure software house" -Computing with the Amstrad, May 1985.
DUNGEON ADVENTURE*† *(cass £9.95)* "A wonderful adventure..a world where electricty bills and mortgages have no place" -MSX Computing, June 1985
LORDS OF TIME*† *(cass £9.95)* An imaginative romp through world history.
SNOWBALL*† *(cass £9.95)* Immense science fiction game with 7000+ locations.
RETURN TO EDEN*‡ *(cass £9.95)* Popular SF adventure. Amstrad, CBM64, MSX and Spectrum versions have 240 pictures.
EMERALD ISLE ‡ *(cass £6.95)* Fantasy adventure. Amstrad, BBC, CBM64, MSX and Spectrum versions have 230 pictures.
ERIK THE VIKING *(cass £9.95)* Game of Terry Jones' book for Amstrad, BBC, CBM64 and Spectrum only. All with 180+ pictures.

Level 9 adventures are available from all good computer shops. If you can't find the game you want please ask them to order it.

You can also buy games directly from us. Just send us a cheque made payable to Level 9 Computing, or a postal order. Don't forget to tell us your address, which computer you have and which game you'd like. (Add £1 for each item if outside the UK.)

If you would just like a **FREE FULL COLOUR POSTER,** with more details of Level 9 Adventures on the back, please write to us enclosing a stamped, self addressed envelope at least 9" by 6". Write 'POSTER' above our address.

Please write to

LEVEL 9 COMPUTING
P.O. Box 39
Weston-super-Mare
Avon BS24 9UR

BBC (Acornsoft dfs)/CBM64 disk versions are now available for those games marked * at £11.95 new or £2.50 upgrade (please return your original cassette).
†Text adventure on all 8 micros. ‡ Selected versions have pictures in addition to text.

Computing with the AMSTRAD
Learn how to unleash the power of binary numbers
Fight the evil Stompers in our fast-moving game
Recover lost programs with Disc Editor
Create a character generator
Start of two great new series!
Amstrad teach-in

Vol. 1 No. 8 August 1985

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

Editorial: 061-456 8835
Administration: 061-456 8383
Advertising: 061-456 8500
Subscriptions: 061-480 0171

Telecom Gold Mailbox: 79:MAG001
Telex: 256871 MONREF G
Quoting Ref. 79:MAG001
Prestel Mailbox: 614568383

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

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

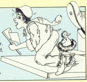
ABC Member of Audit
Bureau of Circulations

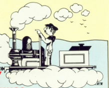
"Computing with the Amstrad" welcomes program listings and articles for publication. Material should be typed or computer-printed, and preferably double-spaced. Program listings should be accompanied by cassette tape or disc. Please enclose a stamped, self-addressed envelope, otherwise the return of material cannot be guaranteed. Contributions accepted for publication by Database Publications Ltd will be on an all-rights basis.

While every care is taken, the publishers cannot be held legally responsible for any errors in articles, listings or advertisements.

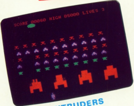
"Computing with the Amstrad" is an independent publication and neither Amstrad Consumer Electronics plc or Amsoft are responsible for any of the articles in this issue or for any of the opinions expressed.

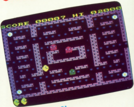
News trade distribution:
Europress Sales and Distribution Limited, 11 Brighton Road, Crawley, West Sussex RH10 6AF. Tel: 0293 27053.





4 GREAT GAMES FOR THE PRICE OF ONE!



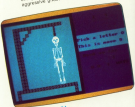


SNAPMAN

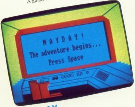
Steer your man around the maze, gobbling up energy pellets while keeping away from the aggressive ghosts who are out to get you.

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!



Pick a letter 0
This is move 5



MAYDAY!
The adventure begins...
Press Space

DEADMAN

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

Computing with the Amstrad presents

CLASSIC GAMES on the Amstrad

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

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

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

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

Cassette £5.95	*6048*	□
3" Disc £9.95	*6049*	□

Payment: please indicate method (✓)

□ Access/Mastercard/Eurocard

No. ____ ____ ____ ____

□ Barclaycard/Visa

No. ____ ____ ____ ____

□ Cheque/PO made payable to Database Publications Ltd.

Name ______________ Signed ______________

Address ______________

____

____

Send to:
***Computing with the Amstrad,* FREEPOST, Europa House,**
68 Chester Road, Hazel Grove, Stockport SK7 5NY.

(No stamp needed if posted in UK) *Please allow 28 days for delivery*

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

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

CA8

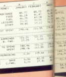
QUICK TO LEA

THAT'S...

FINALIST
BRITISH MICRO COMPUTING AWARDS
1985

MINI OFFICE

SPREADSHEET



```
     A          B        C        D
1: MONEY     JANUARY FEBRUARY   MARCH

2: MORTGAGE     85.72    85.72
3:     FOOD     46.24    41.47
4:     FUEL     46.25    47.28
5:  LEISURE     20.00    20.00
6:    OTHER     99.85    17.12
-------------------------------------
8:TOT SPENT    298.06   211.55
-------------------------------------
11: EARNINGS   321.21   321.21
12:  B. FWD.    27.25     0.00
-------------------------------------
14: TO SPEND   348.46   321.21
15:    SPENT   298.06   211.55
-------------------------------------
17:REMAINING     0.00   109.66
-------------------------------------
19:     SAVE     0.00    82.25
20:  C.FWD.      0.00    27.41
```

JUST LOOK WHAT THIS PACKAGE CAN DO!

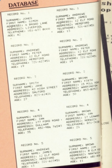
WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

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

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

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

DATABASE



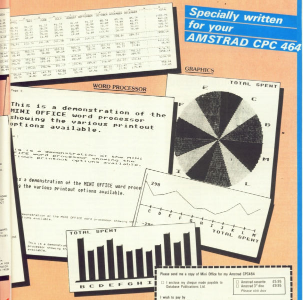
```
RECORD No. 1

SURNAME: JONES
FIRST NAME: SIMON
ADDRESS1: 6 BROAD LANE
ADDRESS2: LIVERPOOL
TELEPHONE: 051-633 8000
AGE: 42

RECORD No. 2

SURNAME: ANDREWS
FIRST NAME: PETER
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 19

RECORD No. 3

SURNAME: SMITH
FIRST NAME: JANE
ADDRESS1: 42 HIGH STREET
ADDRESS2: SALFORD
TELEPHONE: 823-61421
AGE: 27

RECORD No. 4

SURNAME: YATES
FIRST NAME: IAN
ADDRESS1: 177 FORD ROAD
ADDRESS2: OULLHAM
TELEPHONE: 452-986 76543
AGE: 35

RECORD No. 5

SURNAME: ANDREWS
FIRST NAME: JAMES
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 13
```

```
RECORD No. 1

SURNAME: ANDREWS
FIRST NAME: JAMES
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 13

RECORD No. 2

SURNAME: ANDREWS
FIRST NAME: PETER
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 19

RECORD No. 3

SURNAME: BRINN
FIRST NAME: KIETH
ADDRESS1: 15 MILL ROA
ADDRESS2: WARRINGTON
TELEPHONE: 853-80923
AGE: 30

RECORD No. 4

SURNAME: BROWN
FIRST NAME: IAN
ADDRESS1: 17 LEAWA
ADDRESS2: NORWICH
TELEPHONE: 871-343
AGE: 21

RECORD No. 5

SURNAME: BROWN
FIRST NAME: JIM
ADDRESS1: 8 ELM RO
ADDRESS2: NANTWICH
TELEPHONE: 681-45
AGE: 11
```

...and it's all at price of just

RN, EASY TO USE

Specially written for your AMSTRAD CPC 464



```
I----   ---F---  ---G---  ---H---  ---I----  ---J----  ---K----  ---L----  ---M----  ---N---  ---O---
APRIL     MAY     JUNE     JULY    AUGUST  SEPTEMBER  OCTOBER  NOVEMBER  DECEMBER            TOTAL

 85.72    91.37    91.37    91.37    91.37    91.37    85.24    85.24    85.24      :      1055.75
 32.71    41.23    30.29    22.71    27.98    35.99    40.89    39.89    46.45      :       460.26
 22.61    25.41    20.04    22.54    16.85    24.96    29.77    35.55    48.23      :       385.57
 25.00    25.00    25.00    25.00    25.00    30.00    30.00    30.00    30.00      :       305.00
100.87    49.29    16.45    29.96    19.49    36.89   107.90    38.02    39.49      :       651.56
------------------------------------------------------------------------------------------------
276.91   232.50   191.15   201.58   180.69   219.21   293.90   228.80   289.51      :      2858.14
------------------------------------------------------------------------------------------------
                                                                                  :      4111.32
                                                                                  :        25.40
321.21   353.31   353.31   353.31   353.31   353.31   353.31   353.31   353.31      :
 28.49    18.20    34.80    49.24    50.29    55.73    47.46    26.72    37.81      :
------------------------------------------------------------------------------------------------
349.70   371.51   388.11   402.55   403.60   409.04   400.77   380.03   391.12      :      4136.72
276.91   232.50   191.15   201.58   180.69   219.21   293.90   228.00   289.51      :      2858.14
------------------------------------------------------------------------------------------------
 72.79   139.21   196.96   201.17   222.91   189.83   106.87   151.23   101.61      :      1278.58
------------------------------------------------------------------------------------------------
 54.59   104.40   147.72   150.80   167.18   142.37    80.15   113.42    76.21      :       958.94
 18.20    34.80    49.24    50.29    55.73    47.46    26.72    37.81    25.40      :       319.65
```

WORD PROCESSOR

Page 1

This is a demonstration of the MINI OFFICE word processor showing the various printout options available.

Page 1

This is a demonstration of the MINI OFFICE word processor showing the various printout options available.

is a demonstration of the MINI OFFICE word processor showing the various printout options available.

demonstration of the MINI OFFICE word processor showing the options available.

This is a demonstration of the MINI OFFICE word processor showing the available.

GRAPHICS

Please send me a copy of Mini Office for my Amstrad CPC464

□ I enclose my cheque made payable to Database Publications Ltd.	□ Amstrad cassette £5.95 □ Amstrad 3" disc £9.95 *Please tick box*

I wish to pay by
□ Access □ Visa No.__________ Expiry date________

Signed__________

Name__________

Address__________

POST TO: Mini-Office Offer, Database Software,
68 Chester Road, Hazel Grove, Stockport SK7 5NY.

CA8

at the unbelievable

£5.95

CASSETTE
3" DISC £9.95

DATABASE SOFTWARE

ACCESS ORDERS NOW TAKEN

Haystack Peripherals

CPC464 Software

Haystack Peripherals

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

	RRP	OUR PRICE
A'n'F		
Chuckie Egg	7.90	6.90
ADDICTIVE		
Football Manager	7.95	6.95
ALLIGATA		
Blagger	8.95	7.95
Defend or Die	7.95	6.95
ACTIVISION		
Ghostbusters	10.99	9.99
AMSOFT		
American Football	9.95	8.95
Amsword (Advanced)	19.95	17.95
Amsword (Easy)	9.95	8.95
Centrecourt	8.95	7.95
Concise Basic Spec.	19.95	17.95
Concise Firmware Spec.	19.95	17.95
DFM Database*	14.95	12.95
Decision Maker	21.95	19.95
Devpac Ass/Disass	24.95	21.95
Entrepreneur	21.95	19.95
Guide to Basic I	19.95	17.95
Guide to Basic II	19.95	17.95
Hisoft DEVPAC*	34.95	29.95
Hisoft PASCAL	34.95	29.95
Home Accounts Manager*	14.95	12.95
Hurricane Higgins Pool	8.95	7.95
Hurricane Higgins Snooker	8.95	7.95
Invostat*	29.95	26.95
Jet Boot Jack	8.95	7.95
Master Chess	8.95	7.95
Masterfile	24.95	22.95
Pitman Typing Tutor	9.25	8.25
Project Planner*	24.95	22.95
Screen Designer	14.95	12.95
Spreadsheet*	19.95	17.95
Starwatcher	17.45	15.45
Stock Aid*	29.95	26.95
Stockmarket*	8.95	7.95
Transact*	29.95	26.95
AMSOFT/BES		
Animal, Vegetable, Mineral	8.95	7.95
Happy Letters	8.95	7.95
Happy Numbers	8.95	7.95
Happy Writing	8.95	7.95
Map Rally	8.95	7.95
Osprey!	9.95	8.95
Timeman One	8.95	7.95
Timeman Two	8.95	7.95
Wordhang	8.95	7.95
Worldwise	8.95	7.95
ANIROG		
Flightpath 737	6.95	5.95
Moon Buggy	7.95	6.95
Surviver	6.95	5.95
ARIOLASOFT		
Archon	10.95	9.95
Hard Hat Mack*	8.95	7.95
One On One	8.95	7.95
ARTIC		
World Cup Football	7.95	6.95
ASK		
Number Painter	8.95	7.95
BLACK KNIGHT		
Timebomb	7.65	6.65
CCS		
Warzone	6.95	5.95
CDS		
Castle Blackstar*	6.95	5.95
Steve Davis Snooker	7.95	6.95
C.P. SOFTWARE		
Backgammon	8.95	7.95
Bridge Player	9.95	8.95
Draughts	8.95	7.95
Pinball Wizard	8.95	7.95
Superchess	9.95	8.95
CRL		
Amstrad Artist	9.95	8.95
Handicap Golf	6.95	5.95
Test Match	6.95	5.95
CABLE		
Digger Barnes	7.95	6.95

	RRP	OUR PRICE
CAMEL		
FlexiFrend	7.50	6.50
Grasp	8.50	7.50
CREATIVE SPARKS		
Dangermouse in Double Trouble	8.95	7.95
DESIGN DESIGN		
Dark Star	7.95	6.95
Tank Busters	7.95	6.95
DIGITAL INTEGRATION		
Fighter Pilot	8.95	7.95
DK'Tronics		
Minder	9.95	8.95
DOMARK		
A View to a Kill*	10.99	9.99
DURELL		
Combat Lynx	8.95	7.95
Death Pit	7.95	6.95
GARGOYLE		
Dun Darach	9.95	8.95
GEMINI		
Cashbook* (inc. VAT file)	59.95	51.95
Database	19.95	17.95
Final Accounts*	59.95	51.95
Home Accounts	19.95	17.95
Report Generator	19.95	17.95
VAT File*	19.95	17.95
Combo Pack* (Cashbook + final Acc.)	89.95	79.95
GILSOFT		
The Quill	16.95	15.25
HEWSON		
Zapp	14.95	13.45
HISOFT		
Font 464*	7.95	6.95
IMAGINE		
World Series Baseball*	8.95	7.95
INCENTIVE		
Confuzion	6.95	6.25
Millionaire	6.95	5.95
INTERCEPTOR		
Azimuth 3000	8.95	7.95
Forest at Worlds End	6.00	5.25
Heroes of Karn	6.00	5.25
Jewels of Babylon	6.00	5.25
Message from Andromeda	6.00	5.25
Warlord*	6.00	5.25
INTRIGUE		
Tomb of Kuslak	7.50	6.50
KNIGHTSOFT		
Animated Strip Poker	8.95	7.95
KUMA		
Artwork	9.95	8.95
Database	14.95	12.95
Easi VAT*	39.50	34.50
Forth	19.95	17.95
Fruity Frank	6.95	5.95
Galaxia	5.95	5.25
Gems of Stradus	7.95	6.95
Holdfast	5.95	5.25
Home Budget	19.95	17.95
Music Composer	9.95	8.95
Star Avenger	6.95	5.95
LEVEL 9		
Adventure Quest	9.95	8.45
Colossal Adventure	9.95	8.45
Dungeon Adventure	9.95	8.45
Emerald Isle	6.95	5.95
Lords of Time	9.95	8.45
Red Moon*	6.95	5.95
Return to Eden	9.95	8.45
Snowball	9.95	8.45
LLAMASOFT		
Psychedelia	6.00	5.25
LOTHLORIEN (WARMASTER)		
Johnny Reb	6.95	5.95
Redcoats	6.95	5.95
Special Operations	6.95	5.95
MARTECH		
Brian Jack's Superstar Chall.*	7.95	6.95
MELBOURNE HOUSE		
Hobbit	14.95	12.95

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

ADD ONS	
Amstrad Speech Synth	28.95
DK'Tronics Speech Synth	34.95
DK'Tronics Light Pen*	TBA
Pro-Ace Joystick	11.95
Superpower Sideways ROM card*	39.95
Superpower Disk User Utilities ROM*	39.95
Superpower Mailing List ROM*	39.95

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

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

Some disc versions now available – ring for prices.

* Please ring to confirm availability

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

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

Please Send:	Price
TOTAL	

ALL OUR PRICES INCLUDE VAT AND CARRIAGE

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

NAME ....

ADDRESS ....

Phone No. ....

Access No. ....

Expiry Date ....

France welcomes Amstrad

BOOMING sales across the Channel have led Amstrad to claim it now has a 50 per cent share of the French home computer market previously dominated by Oric and Thompson machines.

But there's a major threat to the success story of Amstrad's wholly owned French subsidiary looming on the horizon.

Eureka Informatique which took over Oric says it will launch a micro to undercut Amstrad's – by a hefty £100.

Amstrad is understood to have sold about 60,000 CPC464s in France so far this year. Now orders are building up for the CPC664, which was launched at the recent Sicob exhibition in Paris.

A major factor in the company's penetration of the French market has been the easy transportability of the machines due to their built in monitor which requires no conversion of video output when used abroad.

But its success will be challenged by output from Eureka's Normandy factory, which managing director Jean-Claude Tallar has said will produce up to 35,000 Oric-based micro systems this year.

The new Oric will come complete with Eureka screen and tape recorder – and it may even confront the Amstrad on this side of the Channel following a proposed UK launch later this year.

New machine takes on American market

But no CPC6128s for the UK until next year

AMSTRAD is poised to invade the cut-throat United States marketplace with a new 128k machine.

Known as the CPC6128, it is scheduled to go on sale there this autumn priced at between $699 and $799, depending on configuration.

However this latest addition to the CPC range is unlikely to be available in the UK until early next year.

Amstrad has gone to considerable lengths to avoid the traumas experienced by its rival Acorn when that company failed in its attempt to gain a foothold in the lucrative American market.

This setback cost the manufacturers of the BBC Micro some £6 million and almost brought the company to its knees early this year.

Unlike Acorn, Amstrad will not be having a significant corporate presence in the US but rather will rely solely on a major American importer to run the operation.

The company in question is Indescomp, a well known Chicago based distributor.

Nor will the venture Stateside cost Amstrad a penny in that it will be totally funded by Indescomp.

"We view the American market with interest, but the trading conditions there are difficult", admits Amstrad chairman Alan Sugar.

"And we are not prepared to invest there, preferring to trade through an established consumer electronics importer".

It was left to Alan Sugar to announce that the new machine – it has an RGB high resolution monitor and disc drive – will not be sold in the UK for some time.

"We have no plans to sell the 6128 in the UK this year", he says.

"We are currently marketing two highly successful machines in the 464 and the 664 and there is no need to extend our product range at this time".

Nor is Alan Sugar currently prepared to put a British price tag on the 6128.

"It is certainly too early to predict prices. That is something we will defer until we are about to launch".

A NEW graphic and text adventure game with 240 locations and five skill levels has been launched for the CPC464.

The Tomb of Kuslak, priced at £7.50, comes from Intrigue Software. The game plays in real time but incorporates a feature which allows speed changes to deal with the aliens which appear on a separate screen within the main display.

Three-in-one

THE Code Machine from Picturesque is an editor assembler plus monitor for the CPC464, designed to work either together or to be loaded and used separately by beginners and professional software writers alike.

Picturesque says the monitor is less complex than other systems, giving a clear understanding of how machine code works, yet offering the more experienced programmer all the commands necessary to run and debug programs.

The Code Machine comes on cassette with the option to make working back-up copies on either cassette or disc and costs £19.95.

RED ARROWS CLIMB HIGH

THE ultra-realistic flight simulator Red Arrows, from Database, is set to break all sales records and zoom to the top of the software charts before the end of summer.

Orders are flooding in for the program that puts the Amstrad user at the controls of a British Aerospace Hawk as a member of the world-famous RAF aerobatics team.

Programmers have described it as the most accurate reproduction possible of the thrilling manoeuvres of a real Red Arrows flying display – a degree of realism achieved with input from the pilots and flight crews themselves.

Contest

Another reason for the program's popularity is that each purchaser can enter a free high-score competition to win an expenses-paid trip to meet the Red Arrows pilots at their base at RAF Scampton and watch the team in action.

The game, which will be on sale at air shows where the Red Arrows display their skills, costs £8.95 on cassette and £12.95 on 3in disc.

Part of the proceeds from the sale of the program will go to Service charities, including the RAF Benevolent Fund.

Changes on the board

Ken Ashcroft

AMSTRAD has made two changes to its board of directors. Former finance director Jim Rice has been appointed group operations director with responsibility for Amstrad's companies in Paris and Hong Kong and the Shoeburyness factory.

New finance director is Ken Ashcroft, who joins Amstrad from Dixons where he was director of finance and administration. He has also worked for Hepworths and Comet.

Jim Rice

Second place for the 464

THE CPC464 has failed in its bid to carry off two titles in the British Microcomputing Awards 1985. It shared the runners up position in both major categories for which it had been shortlisted.

The What Micro? Home Microcomputer Award went to the Atari 800XL, leaving the CPC464 and the Spectrum+ in the also ran slot.

And the machine's hopes of carrying off the Personal Computer World Home Microcomputer of the Year Award were dashed when the Sinclair QL turned out to be the judges' favourite.

The CPC464 was the bridesmaid once again, this time sharing the honour with the MSX.

"It all goes to show just how far the so called experts are out of touch", one industry observer said.

"After all, their selections are simply not those of the public at large, these being the people who really count".

In all, more than 1,000 nominations were received.

This year's awards were used as a vehicle to raise money for Comet – Concerned Micros in Education and Training – the charity which seeks to improve the quality of life for disabled young people through micros.

Education net 'for the 1990s'

AN educational network claimed to be five years ahead of its time has been developed based on the Amstrad machines.

Rumours have circulated for some time that systems specialists Northern Computers, of Frodsham, had a spectacular project under wraps.

Now *Computing with the Amstrad* has learned that it is a network, the like of which has never been seen before.

"It will be superior to any other micro network", Gareth Littler, a Northern Computers director, revealed.

"This will be a system for 1990, not just 1985, for a lot of money and time has gone into this project".

It seems that Northern Computers have had some help in developing the network, although Gareth Littler confided: "Much of it is our product".

The first of the networks is likely to be unveiled within the next few weeks.

However the Northern Computer official baulked at being drawn into revealing further details until all contractual paperwork has been completed.

Downloading service

AMSTRAD owners will soon be able to download a wide range of software, much of it free, from Viewfax 258. The service is a joint project of Viewfax, Prestel and Micronet 800.

At present there is a telesoftware test program for the CPC464 and 664 available on Viewfax. The Locomotive Basic program uses the high resolution graphics of the Amstrad to create a patriotic symbol.

The program is in a standard CET format, but future telesoftware for the Amstrad will include enhancements to the CET structure that will allow programs to be protected and machine code files to be delivered with load and execute addresses included in the checksummed data.

A Micronet spokesman told *Computing with the Amstrad* that there will shortly be a dozen free programs available – "a mixture of games, utilities and others, in fact something for everyone".

Later it is planned to widen the telesoftware bank by adding commercial software.

Colour palette package

FOLLOWING the success of its speech synthesizer, DK'Tronics has brought out its second Amstrad/Schneider product using a new concept developed by Mark Anson and Graham Pointer.

It is a light pen supported by a sophisticated software graphics package which gives colour palette – tonal monochrome for green screen models – a choice of brush size, airbrush mode and icon-driven pull down menus.

Users can define their own graphics, move them around, enlarge or shrink them, edit finished screens and save/load to disc or tape.

Features include "nudge" for accurate positioning down to a single pixel size and an extra sensitive light detector.

PLOTTING EDITOR

A PLOTTING package for the CPC464, Genplot from Al-Tukhaim, contains data editing and plotting facilities and can support a variety of dot matrix printers and pen plotters.

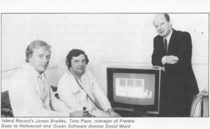
The editor permits adding data sets, copying from one to another, save and re-store. Data sets can be compared by superimposing graphs and bar charts.

The plotting facility enables the user to represent data sets visually on screen in the form of graphs, bar charts, histograms and high-low charts. Cassette price is £14.95. A disc version will be available later.

KEEPING BUSY

KUMA Computers is maintaining its remarkable output of programs for the CPC464 with its 14th title in less than a year.

The latest is Bridge, a challenging version of the traditional game using Blackwood and Stayman conventions. Price: £8.95.



Island Record's James Bradley, Tony Pope, manager of Frankie Goes to Hollywood and Ocean Software director David Ward

Utility in blue shocks customers

PURCHASERS of a mail order tape-to-disc transfer utility for the CPC464 called Amsdisk got more for their money than they expected.

Those who printed out the unprotected program listing were greeted by a blast of abusive language containing 14 words of the four letter variety.

Among those who bought the £5.95 utility, produced by Ultratec Computing of Sunderland, was a Romford man who gave it to his 11-year-old niece.

The girl's father said: "My brother bought Amsdisk for my daughter who is very interested in computing and has her own Amstrad.

"Upon receiving the program she listed it, was unable to understand the words, and called me to assist. You can imagine my horror as I looked at the screen. The result was one very upset family".

The shocked father wrote to members of the software industry with Amstrad connections to alert them about Amsdisk.

David Price, of utilities supplier Interlock Services, said: "This disgusting tape goes beyond a joke, if that was what the author intended. It's disgraceful".

David Snoxell, of Pride Utilities, said: "I find this program really appalling. It gives our section of the industry a bad name".

Author of Amsdisk is 18-year-old Sunderland schoolboy Phil Murray, who started Ultratec Computing with two friends last December.

He told *Computing with the Amstrad:* "It was all a terrible mistake. I made the offending tape out of frustration – I was on edge after hearing that my programs were being pirated.

"The version with the rude words in it was never meant to be sent out to clients, but there was a mix up in the master tapes.

"Fewer than 10 copies went out to customers before my mistake was brought to my attention and I immediately checked the masters and destroyed the offending version.

"I'm sincerely sorry for causing any distress – it certainly won't happen again. I want to make a reputation as a respectable software creator".

Stage to tape

TWO new releases for the CPC464 from CRL are "The Rocky Horror Show" and "Amstrad Artist".

The first is a game based on the stage production written by Richard O'Brien and costs £8.95.

The second is a graphics package containing facilities for the design of screens and the creation of sophisticated graphics. It also contains a character generator, a sprite designer, enabling the creation of up to 12 sprites, and a data to machine code conversion program which enables sprites to be utilised within the user's own programs.

Price is £9.95.

Relax with Frankie

A FREE audio-cassette containing a previously un-released recording of "Relax" will be included with the Frankie Goes to Hollywood game program for the CPC464 from Ocean Software.

The game is a joint publishing venture of the band, their recording company, Island Records, creative producers ZTT and Ocean.

Ocean director David Ward said: "Datatune is a new idea – players load the game from the program cassette and then insert the audio cassette.

"A voice-over describes how to play the game on side one, and on the flip side players can hear some inspirational music in the form of one of the band's hit recordings".

Frankie Goes to Hollywood will cost £9.95.

LIGHT PEN DOWNLOAD

AMSTRAD owners will soon be able to download utilities software from their TV sets using the new Visicode Light Software system.

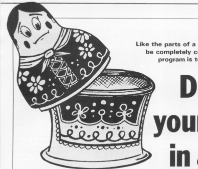
Michael Feldman, producer of Thames Television's networked computer programme "Database", told *Computing with the Amstrad:* "We are currently developing utilities software for Visicode and hope to start broadcasting it toward the end of September".

The downloading medium is a 400 baud light pen that picks up at 50 bits a second.

It is available by mail order only from Magenta Electronics of Burton-on-Trent and costs £6.96 in kit form and £8.60 ready built.

Amstrad owners got their first chance to use Visicode last month when "Database" broadcast a four part text-based adventure game.

The new light pen downloads nearly 10 times faster than the systems used earlier this year on Channel 4's "4 Computer Buffs" programme.



Like the parts of a Russian doll, each loop must be completely contained by the other if your program is to fit together properly. So . . .

Don't get your NEXTs in a twist!

LAST time we had a look at the way we can use FOR ... NEXT loops to repeat a body of code for a fixed number of times. We saw that we could use the same program lines over and over again by sandwiching them between a FOR and a NEXT.

By PETE BIBBY

This not only increases the power of a program but also saves a lot of typing. Program I should hold no difficulties for you.

```
10 REM PROGRAM I
20 FOR row=1 TO 10
30 PRINT "*";
40 NEXT row
```

Program I

This is a very elementary use of the FOR ... NEXT loop formed by lines 20 and 40. The body of the loop is line 30. This prints an asterisk on the screen each time the loop cycles.

The apostrophe at the end of the line makes sure that the asterisks are "glued" together, one after the other. Leave it out and see what happens.

The number of times the loop cycles is determined by the loop control variable *row*. This starts with a value of 1 and is incremented by 1 each time the program reaches a NEXT.

When *row* is eventually increased to 11 the loop finishes, having cycled 10 times. The result is a line of 10 asterisks stretched across the screen.

Don't just run the program and leave it at that, though. Try varying the control variable *row* to produce lines of 20 or 30 asterisks. Notice that:

```
FOR row=10 TO 19
```

or:

```
FOR row=99 TO 90 STEP -1
```

also produces 10 asterisks. You don't always have to have 1 as the initial value of the loop control variable.

The main point to grasp is that every piece of code between the FOR and the NEXT is repeated. It doesn't matter if it's a PRINT command or a LET or whatever. While the loop is still in operation everything inside the FOR ... NEXT's boundaries is performed repeatedly.

This is so even if it's another FOR ... NEXT loop that makes up the body of the loop. Program II shows you what I mean.

Lines 30 to 50 should hold no mysteries, they're the same as the FOR ... NEXT loop of the previous program. As we saw before, that particular loop will cycle 10 times and produce a line of 10 asterisks. However with Program II we get five rows of 10 asterisks. How has that happened?

```
10 REM PROGRAM II
20 FOR freshrow=1 TO 5
30 FOR row=1 TO 10
40 PRINT "*";
50 NEXT row
60 PRINT CHR$(13)
70 NEXT freshrow
```

Program II

The answer lies in the FOR ... NEXT loop formed by lines 20 and 70. By looking at line 20 we can see that this loop will cycle five times while the control variable *freshrow* takes values from 1 to 5.

Never mind about the stuff in the middle. For the moment just concentrate on that outer loop cycling five times.

Now, as we've seen before, everything inside a FOR ... NEXT loop is repeated as the loop cycles. So the body of the loop – that's everything between the FOR of line 20 and the NEXT of line 70 – will be repeated five times.

However, instead of just being a PRINT command as in previous programs, the body of the loop is now another FOR ... NEXT loop. And, as we've seen, this FOR ... NEXT loop wants to cycle 10 times. This is, in fact, what happens.

Each time the outer loop, controlled by *freshrow*, repeats once the inner loop is performed in its entirety. That is, for each cycle of the outer loop, the inner loop, controlled by *row*, repeats 10 times producing the familiar row of 10 asterisks.

As the outer loop cycles five times in all, and as the inner loop cycles 10 times for each time round the outer loop, we get five rows of 10 asterisks.

This is an example of what is known as nested loops. In this case there is one FOR . . . NEXT loop (lines 30-50) nested inside another (lines 20-70). Each time the outer loop cycles once the inner loop cycles for its full quota of repetitions.

Again, don't be content with the example program, try varying it. What would lines like:

```
20 FOR freshrow=10 TO 15
30 FOR row=10 TO 5 STEP-1
```

or:

```
10 FOR freshrow= 10 TO 50 STEP 5
20 FOR row= 15 TO 21 STEP 2
```

produce? Did the number of rows of asterisks and the number of asterisks in each row tally with your expectations?

Notice the care you have to take with your control variables to avoid having more cycles of each loop than you really wanted.

Incidentally, can you see the point of line 60? It's there to undo the effects of the semi-colon at the end of line 40 when the program leaves the inner loop. It does this by producing what's known as a carriage return. Try leaving it out and see what happens.

Armed with our new-found mastery of nested loops we can turn to last month's Program XIII (this month's Program III) and see how it works.

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

Program III

As you can see, it consists of two loops, one nested inside the other. The inside loop is formed by lines 50 to 70 and has control variable *inner*. This takes values from 1 to 3 and each time it cycles the "inner loop" message is displayed.

The outer loop is formed by lines 30 and 80 and, unsurprisingly, has control variable *outer*. This, too, ranges from 1 to 3. However, each time this outer loop cycles *it not only prints out a message but also performs all three cycles of the inner loop.*

By the time the outer loop has completed its quota of repeats the inner loop will have been through its full quota three times.

Let's follow the program as it runs. Lines 10 and 20 are just REMs, which the Amstrad ignores. Line 30 tells the micro that it is starting a loop and that the variable *outer* is to be used to control the loop. Initially this has the value 1.

The program now goes on to line 40, and displays the message:

```
This is outer loop 1
```

Line 50 marks the start of another FOR . . . NEXT loop. This has its control variable *inner* taking values from 1 to 3. At this stage the Amstrad gives *inner* the value 1 and goes on to line 60. This results in the message:

```
This is inner loop 1
```

and the program goes on to the NEXT of line 70. Here *inner* is increased by 1 and the program goes back to the FOR of line 50. Since *inner* is within the limits set, control goes to line 60 and the message:

```
This is inner loop 2
```

appears on the screen.

Again the NEXT is met at line 70, *inner* is increased to 3 and the program goes back to line 60. Line 60 now prints:

```
This is inner loop 3
```

and the program comes across the NEXT of line 70 again. Now, however, when *inner* is increased to 4 the conditions of the loop are exceeded. The program drops out of the loop and reaches line 80. Here it finds the NEXT of the outer loop.

Now *outer* is increased to 2 and the program goes back to line 30. Line 40 displays:

```
This is outer loop 2
```

and line 50 takes the program back into the inner loop, producing the three "inner" messages as before.

Once this is done the program drops out of the inner loop and again encounters the NEXT of line 80, *outer* now becomes 3 and the program goes back to line 30 for the final time. When *outer* is 4 the program finishes.

Don't worry if you don't grasp it in all its complexity. Nested loops take a bit of getting used to, but once you've played around with them for a while they become second nature.

All you have to grasp is that for each single cycle of the outer loop the inner loop goes through all its cycles. Adding a line like:

```
55 PRINT "outer="outer "inner="inner
```

might make things clearer.

While we're still with Program III I should point out that you don't need to put the control variables after the NEXT. Program IV shows this.

```
10 REM PROGRAM IV
20 FOR outer=1 TO 5
30 PRINT "This is outer loop "outer
40 FOR inner=1 TO 4
50 PRINT TAB(15)"This is inner loop "
   inner
60 NEXT
70 NEXT
```

Program IV

However, while the Amstrad may be clever enough to keep track of things, you might not be. Take my advice, and always put in the variable, at least until you've got your program working properly.

Notice that Program IV is a variant of Program III. There have only been two minor alterations to parts of lines, but look at the difference in output. As you might appreciate, nested FOR . . . NEXT loops are both flexible and powerful.

So far we've only nested one FOR . . . NEXT loop inside another. It's perfectly possible to have three or more loops nested, one inside the other rather like those Russian dolls on the front of spy stories. They work in much the same way but are easier to understand in practice than in theory. Add:

```
45 FOR middle=1 TO 3
47 PRINT TAB(8)"This is middle
loop"middle
75 NEXT middle
```

to Program III and you'll see what I mean. And notice the vast increase in output for just three extra lines. As I said, these nested loops are powerful.

There's one thing to be wary of when you're messing around with nested loops. Don't get your NEXTs in a twist. Program V shows this very well.

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

Program V

When you run this you'll get the error message:

```
Unexpected NEXT in 70
```

glaring at you from the screen. It won't take you long to realise that you've confused the computer by getting the control variables in lines 60 and 70 the wrong way round. They should have been:

```
60 NEXT inner
70 NEXT outer
```

If you really want to test your understanding of nested loops can you explain why changing the last lines to:

```
60 NEXT outer
70 NEXT outer
```

should produce the message:

```
This is outer loop 1
```

before the error message?

But enough of what can go wrong, it's too theoretical. After all, you and I don't make mistakes, do we? We're too busy doing important things like producing triangles of asterisks. Have a look at Program VI.

```
10 REM PROGRAM VI
20 FOR length=1 TO 10
30 FOR row=1 TO length
40 PRINT "*";
50 NEXT row
60 PRINT CHR$(13)
70 NEXT length
```

Program VI

While the output is nothing to rave about, the program does contain some interesting points. By now you should be able to recognise the two sets of nested loops and the CHR$(13) of line 60 should hold no fears.

The outer loop with control variable *length* ranging from 1 to 10 is fairly straightforward. The inner loop is rather different. Here the control variable *row* goes from 1 to *length*. Now *length* is changing each time round the outer loop.

The first time round *length* is 1, so the control variable of the inner loop goes from 1 to 1. This results in the first, solitary asterisk.

The next time round the outer loop *length* is 2, so now when the program reaches the inner loop, *row* takes values from 1 to 2. A pair of asterisks appears.

The third time round the outer loop, *length* is 3, so the inner loop produces a trio of asterisks as *row* goes from 1 to 3. By the time that *length* is 10, 10 asterisks are displayed.

The point to grasp is that the number of repetitions of the inner loop depends on the control variable of the outer loop. The loops are not only nested, the outer controls the inner.

Can you alter the program so that it produces the same triangle but upside down? Program VII shows how it's done.

```
10 REM PROGRAM VII
20 FOR length=10 TO 1 STEP -1
30 FOR row=1 TO length
40 PRINT "*";
50 NEXT row
60 PRINT CHR$(13)
70 NEXT length
```

Program VII

Here *length* is decreasing each time round the outer loop, hence the reducing number of asterisks in each line.

Once you're sure that you understand the last two programs have a go at producing the mirror image of Program VI's asterisks.

It's not quite as simple as the previous programs, but if you remember how LOCATE works you should have no problem seeing how it works. Program VIII shows how it's done.

```
10 REM PROGRAM VIII
20 CLS
30 FOR row=1 TO 10
40 FOR length=1 TO row
50 LOCATE 11-length,row
60 PRINT "*"
70 NEXT length
80 NEXT row
```

Program VIII

Now can you turn it upside down? The alterations are:

```
30 FOR row=10 TO 1 STEP-1
50 LOCATE 11-length,11-row
```

And that's it for this month. I'll leave you to mess around with nested loops. Keep on trying to produce patterns of asterisks.

While it's hardly the most important use of your Amstrad, you'll be amazed at how it will increase your grasp of nested loops.

And when you get tired of that, can you figure out what's happening in Program IX?

```
10 REM PROGRAM IX
20 CLS
30 asterisk$="*"
40 FOR row=1 TO 10
50 LOCATE 11-row,row
60 PRINT asterisk$
70 asterisk$=asterisk$+"*"
80 NEXT row
```

Program IX

It produces the same output as Program VIII, but there's only one loop instead of two. How's that happened? The answer lies with string variables, and we'll be looking at them next time.

AmsDOS disc commands at your fingertips in the seventh of our Amstrad quick reference charts

FILENAMES: The names of files saved on discs are in two parts. The first can be up to eight characters long, the second up to three. These parts have a fullstop, . , separating them.

Hence:

```
        EXPENSES.1
        TEST
        A.END
```

are all valid filenames while:

```
        OVERFLOWS.ONE
        TEST.FOUR
```

aren't.

The last three letters of the filename are called the filetype. If you don't supply them, the Amstrad might, using the following filetypes:

.BAS Usual basic program.
.BIN Binary file.
.BAK Backup version created when you save a file with a previously used name. You can use your own filetypes. For example, AUG may suffix all files concerned with August, SEP those with September and so on.

When two drives are in use they are known as A and B. The micro looks to only one drive, known as the default drive, for a file. Unless it is changed, drive A is the default drive. To overcome this, filenames can be prefixed with A: or B: to pick the drive. A:JUNE.PAY refers to the file JUNE.PAY on the disc in drive A while B:JUNE.PAY means a file on that in drive B.

WILDCARDS: These are symbols used in place of part or all of a filename in order to select filenames of similar types or names. The question mark? stands for one character, while the asterisk * takes the place of several characters.

AMSDOS COMMANDS make use of two symbols, ¦ and @. Both are found on the key to the right of P.

WHICH DRIVE?

¦A Drive A default drive.
¦B Drive B default.

or you can use DRIVE with a string to pass data.

```
        choice$="b"
        ¦DRIVE,@choice$
```

selects drive B as the default.

TAPE OR DISC? Five commands decide whether tape or disc or a combination of both are used for input and output:

¦DISC.IN Discs to be used for input
¦DISC.OUT Discs to be used for output.
¦DISC Combines the above.
¦TAPE.IN Cassette input.
¦TAPE.OUT Cassette output.

NEW NAME: REN is used to rename a file. This makes use of two strings to pass data. To rename the program AAAA.AAA to ZZZZ.ZZZ use:

```
        past$="AAAA.AAA"
        future$="ZZZZ.ZZZ"
        ¦REN,@future$,@past$
```

WHAT'S ON THE DISC? CAT gives you the directory.

¦DIR Gives CPM style directory.

```
        name$="?.BAK"
        ¦DIR,@name$
```

uses a string, in this case *name$*, and a wildcard to display all the files with a name made up of a single character followed by the filetype BAK.

ERASE FILE: ¦ERA, with the filename held in a string, is used to wipe files from a disc. Wildcards can be used. To get rid of all files with the filetype DOG use:

```
destroy$="*.DOG"
¦ERA,@destroy$
```

It takes two to rendezvo

NIGEL PETERS takes another step or two down Melody Lane in Part VII of his series on CPC464 sounds

LAST month we introduced the concept of the sound queue and saw how we could use just one channel parameter to ensure that the same note plays on all or any of the three sound channels available. We did this by adding together the channel parameters 1, 2, and 4 as necessary.

Also we saw how we could cause notes to wait for sounds on other channels by using the appropriate rendezvous factor – 8, 16, or 32.

Bearing all this in mind, Program I should cause you no difficulties:

```
10 REM PROGRAM I
20 SOUND 17,239,100,5
30 SOUND 1,190,100,5
40 FOR delay=1 TO 200:NEXT delay
50 SOUND 1,213,100,5
60 SOUND 1,239,100,5
70 SOUND 10,119,100,7
80 SOUND 2,127,100,7
90 SOUND 2,159,200,7
```

Program I

Here there are four SOUND commands producing one-second notes on channel A, and three giving notes on channel B. However, the tune doesn't start playing until after the delay loop of line 40 has finished, hence the slight pause before the "music".

The reason is that the SOUND command of line 20 has a channel parameter of 17. This pops a note on the channel A sound queue, but it won't start playing it as the parameter of 17 (1+16) tells the Amstrad to wait for a note on channel B.

However, it's not just any note – it has to be one with a parameter of 10 (2+8) which marks it as one waiting for a rendezvous with channel A.

This means that the note produced by line 20 has to wait until the program gets to the note produced by line 70 before the rendezvous is complete and both notes can play. In the meantime three more notes are put on the channel A queue and the program has worked its way round the delay loop of line 40.

Those with a long memory may recall that Program I is very much like the Program VIII of last month. The only difference is that I haven't bothered to rendezvous any of the notes after the first pair.

Program II is identical to the old Program VIII except that the delay loop has been increased so that there's a longer pause before the tune starts.

```
10 REM PROGRAM II
20 SOUND 17,239,100,5
30 SOUND 17,190,100,5
40 FOR delay=1 TO 1000:NEXT delay
50 SOUND 17,213,100,5
60 SOUND 1,239,100,5
70 SOUND 10,119,100,7
80 SOUND 10,127,100,7
90 SOUND 10,159,200,7
```

Program II

As you can hear, it works but there's not real need for all the 10s and 17s in the channel parameters. In this case, so long as the first two notes start together, the rest will be fine.

However if we introduce the delay loop at different points in the program we can get problems. Try deleting line 40 and adding:

```
75 FOR delay=1 TO 1000:NEXT delay
```

to Program I and see what happens. Now you have to rendezvous the next notes on each channel to overcome the hiccup.

Sometimes, when you've got a string of notes all over the program and you're not sure what delays may occur between them, it's better to rendezvous the lot. You may get some odd gaps, but it's better than the notes of a tune getting out of step.

But if you're going to rendezvous notes, remember that both notes have to have the fact flagged in their channel parameters. Have a look at Program III.

```
10 REM PROGRAM III
20 SOUND 17,239,100,5
30 SOUND 17,190,100,5
40 FOR delay=1 TO 200:NEXT delay
50 SOUND 17,213,100,5
60 SOUND 1,239,100,5
70 SOUND 2,119,100,7
80 SOUND 2,127,100,7
90 SOUND 2,159,200,7
```

Program III

Here we've got seven SOUND commands, yet we only get three notes. Where are the other four?

The answer is that they're still hanging round in the channel. A sound queue waiting forlornly for a rendezvous with some gentle, well-meaning notes on channel B.

After popping these notes on the queue the program, came to the last three SOUND commands and played them. These were the three notes we heard.

Of course we know that we meant them to rendezvous with the channel B notes, but we didn't tell the Amstrad that, so channel A remained mute.

ous..

The notes are still lurking there, waiting to be summoned. Try:

```
SOUND 10,119,100,7:
SOUND 10,127,100,7:
SOUND 10,159,100,7
```

and you'll hear them. It's a kind of magic. You enter three SOUND commands and you get seven notes. However it's not the kind of magic that you want in a tune.

Those phantom notes were taking up room on the channel A queue. If we'd tried to put any more on the queue it would have been full and the program would have ground to a halt. If you don't believe me, try adding:

```
55 SOUND 1,100,100,7
```

to Program III and explain where the three notes went!

Incidentally, you might find it worth your while to set up the small Enter key with:

```
KEY 139,"SOUND 135,0,0,0"+CHR$(13)
```

Now, when the sound channels get out of hand, just press the small Enter key and all the garbage will be cleared.

So far we've been using the SOUND command to make noises. There can be times when we don't want it to make a sound. Have a look at Program IV and you'll see what I mean.

The program uses channels B and C to produce a series of notes. However, the second and third notes on channel C are silent, corresponding to the "rests" in a piece of music. Let's take a closer look at the listing.

```
10 REM PROGRAM IV
20 SOUND 2,239,100,7
30 SOUND 4,119,100,5
40 SOUND 2,213,100,7
50 SOUND 4,0,100,0
60 SOUND 2,190,100,7
70 SOUND 4,0,100,0
80 SOUND 2,179,100,7
90 SOUND 4,239,100,5
```

Program IV

Lines 20 and 30 produce one-second notes on channels B and C respectively. So do lines 40 and 50, but take a look at line 50's SOUND command. This produces a one-second note on channel C but it's a strange note. Its pitch and volume parameters are both 0.

The result is one second of pure silence on channel C, coinciding with a note on channel B.

Lines 60 and 70 work the same way, producing a note on channel B and silence on channel C. The last pair of SOUND commands sound a final note on each channel.

As you use the Amstrad's sound facilities to produce more complex tunes, with two and three-part harmony, you'll find that there are lots of times when you want to produce silence. And you'll also find that the above method can have drawbacks. A much neater way of ensuring that a channel stays silent until you want it to sound is shown by Program V:

```
10 REM PROGRAM V
20 SOUND 34,239,100,7
30 SOUND 20,119,100,5
40 SOUND 2,213,100,7
50 SOUND 2,190,100,7
60 SOUND 34,179,100,7
70 SOUND 20,239,100,5
```

Program V

This produces the same notes as the previous program but uses two fewer SOUND commands. The four commands that produce the notes on channel B are the same. What's happened is that we've dropped the silent dummy notes.

This is made possible by using the rendezvous facility. The channel parameters of lines 20 and 30 are arranged so that they rendezvous with each other. These two notes will start playing straight away while the channel B notes produced by lines 40 and 50 are popped on to the queue.

Finally the SOUND commands of lines 60 and 70 are made to rendezvous. This means that the second note on channel C won't sound until the fourth note on channel B starts sounding.

In other words, channel C is silent while the second and third notes are played on channel B. We have our periods of silence without the need for dummy notes.

Strictly speaking, there's no real need to rendezvous the first two notes of Program V – the program will work perfectly well without it. However, I always like to rendezvous the notes that start a tune just in case. The last two notes have to be periods.

Program VI shows what happens if they're not:

```
10 REM PROGRAM VI
20 SOUND 2,239,100,7
30 SOUND 4,119,100,5
40 SOUND 2,213,100,7
50 SOUND 2,190,100,7
60 SOUND 2,179,100,7
70 SOUND 4,239,100,5
```

Program VI

The result is chaos!

One thing you may have noticed about using the rendezvous facility is that you can only make a note wait until another note is ready to play. In other words, it takes two to rendezvous.

If you're only using one channel and you want to stop a note playing straight away, this can cause problems. Of course you can use delay loops or rendezvous with dummy notes on other channels, but it's not very neat.

What you need is a way of putting a note on the sound queue and telling it to wait there until you tell it different. What you want is the ability to "hold" a note.

You can get this by adding 64 to the channel parameter of the note that you want to be held on the queue. Suppose you want a one-second note on channel A but you want it held on the queue until you need it. One way you could do it would be to use:

```
SOUND 17,200,100,7
```

and then invoke it with a dummy note on channel B such as:

```
SOUND 10,0,0,0
```

However, if there are already some notes in the channel B queue, you've got problems.

The way to do it is to add 64 to the channel parameter of the note you want held. In this case 64 + 1 is 65 so the SOUND command you want is:

```
SOUND 65,200,100,7
```

Try it and you'll hear nothing as the note has been held on the queue. To hear the note you've got to tell the Amstrad to let it out of custody. This is done with the aptly named RELEASE command, followed by the appropriate channel parameter. So to hear our note we would use:

```
RELEASE 1
```

It's the same for channels B and C. To hold a note on them we add 64 to the channel parameters (2 and 4 respectively) and release them with:

```
RELEASE 2
```

and

```
RELEASE 4
```

as needed.

From our previous experiences of combining channel parameters you shouldn't be surprised to learn that:

```
SOUND 71,500,100,7
```

holds the same note on all three channels (1+2+4+64) while:

```
RELEASE 7
```

(1+2+4) releases them. Program VII shows how it works.

```
10 REM PROGRAM VII
20 FOR CHANNEL=1 TO 7
30 SOUND CHANNEL+64,200-CHANNEL*10,10
0,7
40 FOR DELAY=1 TO 1000:NEXT DELAY
50 RELEASE CHANNEL
60 PRINT "SOUND WITH CHANNEL PARAMETE
R "CHANNEL
70 NEXT CHANNEL
```

Program VII

Let's go back to the case of holding a note on just one channel for a moment. Once a note is held on a queue, any more notes for that channel get piled up behind it. For example:

```
SOUND 65,100,100,7:
SOUND 1,400,100,7
```

results in silence. The first note has a hold on it while the second is stuck behind it, waiting for its release. Try:

```
RELEASE 1
```

and you'll hear both notes. By releasing the first note you've freed the log jam.

However, if you've two held notes on a queue you've got to release them both.

If you've entered something like:

```
SOUND 65,600,100,7:
SOUND 65,300,100,7
```

then

```
RELEASE 1
```

only gives you the first note. The second moves up the channel A sound queue and sits there. It's held until you free it with another:

```
RELEASE 1
```

Let's recap on what we've covered on channel parameters. We've seen that 1,2 and 4 refer to channels A, B and C respectively. These can be combined as necessary to produce notes on more than one channel.

number	result
1	uses channel A
2	uses channel B
4	uses channel C
8	rendezvous with A
16	rendezvous with B
32	rendezvous with C
64	hold until RELEASEd

Table I: Channel parameters

We've also seen that by adding 8, 16, and 32 to these channel parameters we can cause the notes to rendezvous with notes on channels A, B and C respectively. And on top of all this we've learnt that adding 64 to the parameter holds that note.

It's a sort of mix and match. You decide what channel you want a note to sound on and what are the conditions that will cause it to play and add together the necessary parameters.

Try entering:

```
SOUND 97,200,100,7
```

The channel parameter of 97 means that the note is to be on channel A, it is to rendezvous with a note on channel C and also it is to be held. You can see this from the fact that 97=1+32+64. Now give it a note on channel C for it to rendezvous with:

```
SOUND 12,500,100,7
```

The result is silence. The reason is that while we've given it the right note for a rendezvous (12=4+8) the channel A note is held until further notice.

The channel A note needs releasing while the channel C note needs to rendezvous with one on channel A. So both notes are held on queue. Put them out of their agony with:

```
RELEASE 1
```

which will cause both notes to sound.

Notice that the sound they make is exactly the same as that produced by:

```
SOUND 1,200,100,7:
SOUND 4,500,100,7
```

the result of taking away all the rendezvous and hold values in the channel parameters of the previous pair of notes.

I'll leave it to you to figure out why:

```
SOUND 97,800,100,7
```

followed by:

```
RELEASE 1
```

produces silence until you give it a note such as:

```
SOUND 12,400,100,7
```

Remember, it takes two to make a rendezvous, even if one has just been released from custody.

And that's where I'll stop for this month. We've covered a lot of ground, some of it fairly obscure at first glance. The hold and rendezvous facilities are one of those things that can be difficult in theory but become clear in practice. And practice is the key word.

Have at go at playing notes on all three channels using combinations of the channel parameters we've looked at so far. Table I gives a summary of them all.

Make sure you understand what's happening, why you get the notes you do, and, more especially, why you don't get the notes you intended!

And, when you get completely stuck, don't forget that you've got the small Enter key set up as a panic button. Press this and you'll clear all the queues.

We'll see how that works next time.

Java Star's a sparkler!

THE object of **Mystery of the Java Star,** by Shards, is to locate the wreck of the Sea Witch, a ship that sank in 1767, and recover its cargo of gold and a mysterious ruby called the Java Star.

The first part of the adventure takes place in Bristol. You have found an old sea chest which contains the torn pieces of a map and a handwritten note.

The note, once put together, gives information about the site of the shipwreck, and the map is one of a nearby island. Now save your progress on tape and load in the next part of the adventure.

You find yourself in London with various famous locations to visit.

Some of these, such as Hatton Garden, provide information necessary to your quest, while others are red herrings.

Having gathered all the information, you are given a short test. Remember to write everything down – I didn't the first time! You must again save your progress and load in the next part.

Your task now is to locate the island. Armed with a map of the South Atlantic, you have to determine the area in which the Sea Witch sank, and then find the island.

There are several displayed and you have the option of examining your own map, zooming in on an island, continuing on to another group of islands, or registering a particular island as being the one that you seek.

On finding the correct island, you must once again save your position and load in the last part of the game. A map of the island is displayed and you must use the cursor keys to position a boat over the spot where you think the Sea Witch sank.

The note you found in Bristol should guide you here – and, having determined a position, you send a diver down to investigate.

If you've found the right spot, a plan of the ship is displayed and you must move your diver around searching the various cabins and lockers until you find the gold and the ruby.

You have five minutes to recover the cargo – though, if you are running short of time, you can resurface for another five minutes' air supply.

Run out of air and you're back to the start of this section. Succeed and you are given scores based on how well you tackled each stage of the adventure and an overall score.

All in all this is an excellent program that is really four adventures in one, and is recommended.

Paul Gardner

Not a scintillating Prize...

AS captain of the spaceship Almazoon – courtesy of Arcade Software – you are faced with the daunting task of negotiating each floor of a multi-level maze.

The climax of your quest will be the discovery of the secret which is held in the chamber of Midas. This is presumably **The Prize.**

Each screen display represents a small section of the maze. Fly through one of the exits and a new part of the maze is quickly drawn.

The walls are constructed of some very colourful and quite artistic patterns, but unfortunately these are the high point of the game.

The Almazoon can be flown in any of eight directions and is steered via the cursor keys or, thankfully, a joystick.

Your ship is armed with laser pulse torpedoes, which are very effective against the aliens you encounter – though targets should be chosen carefully as you can only carry 75 torpedoes.

This may seem a lot when you start out, but the trigger-happy pilot may soon find himself in the embarrassing situation of having to run from the enemy in a frantic attempt to locate a power plant and re-arm his ship.

The characters themselves are colourful and move smoothly, but getting hit is a bit of a disappointment. The inlay card describes this as having your "debris scattered through all eternity". Dramatic, eh?

Actually all that happens is all the characters are halted and a crashing noise is heard. Quite often half of your ship is obscured by a black square which was used to erase the missile that hit you. This is what I call tatty programming.

Another little gem, which may or may not be a bug, is the ability of the Almazoon to fire around corners.

Find yourself a suitable corner and park the ship tight against the wall with just a fraction of the front end protruding.

Next, pull the joystick towards the wall – the ship does not move as it is too close to the wall. Finally, press the fire button and you will find that the torpedoes will now emerge at right angles from the ship.

Having endured The Prize for the past few hours, I have come to the conclusion that Amsoft are in such a hurry to build up the amount of software available for their machine that they are accepting some very mediocre material into their once impressive collection.

Jon Revis

Beware parrot!

RETURN to Eden is the second in Level 9's silicon dream trilogy and the first of their adventures to have graphics.

The plot follows closely on from where Snowball left off,

▶

though you don't need to have played the latter.

You start the game in a stratoglider on the planet's surface. Unfortunately, the crew of the Snowball believe you to be a saboteur and are trying to kill you. Too much deliberation at this point and they will succeed.

An underground hiding-place is simple to find, though I hope you remember to wear your radiation suit.

Your task now is to get into the robot-controlled city to the east. All the planet's flora and fauna have been attacking the city for some time and so it has, in self-defence, erected several defence systems.

These are a fence, a zone guarded by sensors, a minefield, a high wall, a field patrolled by deadly robot vehicles, and, finally, an impenetrable dome over the city.

You will also find that helicopter gunships patrol outside the city shooting anything that moves. The animals will attack anything that looks like a robot. As you can see, getting to the city is no easy task.

The first locations you should explore after the threat from the Snowball has been avoided are those in a forest and by a river that you will find to the east. Beware the pirate parrot who will steal treasure from you and hide it in his nest in the maze of maize.

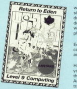
I'm sure Level 9's sense of humour is getting worse.

You have to cross the river and to do so you'll need a boat. Since this is a frontier planet, don't expect your boat to look like one.

You'll also need a present for the leviathan hiding in the river. Making a present of a sweet pea should get you a brick to plant. Having done so, you should enter chez plant – all very fishy.

I think you will find a use, eventually, for everything you find here, though you will need to find a means of increasing your strength if you want to carry everything.

One tip, or should I say pun, is that bulbs grow into shoots – para-shoots!



Provided you have managed to avoid radiation sickness by taking your medicine, you should find yourself floating down to another forest.

Typing SCORE about now should give you a rating of Space Ensign 7th class!

You will find the balloons to be invaluable if you have left anything behind.

Your first obstacle, the fence, is close by and you will need to up the ANTe to get past it. If you can't reach the branch, throw something at it. When you get up there, you will find quite a few branch lines.

A few last tips: A sad fruit will STICK, or something similar, a fragile object. Don't drop that cherry whatever you do. You will need to make a wish with it, if you can spot the physical similarities.

As you can see, Return to Eden is as complicated and as complex as Level 9's other adventures.

If you have read Harry Harrison's Deathworld series, you will have a rough idea of the scenario. And, before Pete Austin writes any more contradictory letters, let me point out that the booklet that comes with the adventure itself claims to have got inspiration from the first of the Deathworld books.

The problems are as fiendish as ever to solve, but to my mind are not as logical as they should be. I mean, SQUEEZE LOG, to water the bulb! I confess that had Level 9 not included a hint sheet, I would still probably be wandering along the river.

I've mapped about 70 locations and have only just managed to work out how to get past the first obstacle! As you can see it is a BIG adventure.

You are probably wondering why I have not yet mentioned the graphics. To be honest I have not got much to say. I'm afraid I didn't like them and used the WORDS command to turn them off.

Possibly I missed a few clues by doing this, but I found they slowed things down too much – though to be fair, they were quickly drawn.

Also, the text quite often didn't fit in the screen area left and had to be scrolled up using the Shift key. This got quite confusing.

Level 9 has long been master of the text adventure. However, I'm afraid it is not going to win any awards for graphics.

While Return to Eden is superior to any adventure I've seen from other software houses, it is nowhere near as good as previous Level 9 games.

I think the humour has been overdone and as a result the "atmosphere" has suffered. However, it is still good value for money and well worth adding to the collection.

Paul Gardner

A winner for a beginner

BRAWN Free from Nemesis, is aimed at the newcomer to adventuring. The object of the game is for you, as Johnny Backwater, to get a bottle of patent medicine from the medicine man in Tombstone for your ailing granny.

The money requred to buy it is to be found in various locations, and you have to collect fifty dollars before you can purchase the medicine and complete the adventure.

You begin your quest in Boot Hill and the first three bills are lying there ready for you. You'll discover that none of the bills is hidden – just the locations they are to be found in.

Your initial explorations lead you to several dead-ends.

The ditch isn't that wide, but you'll need a ticket to get into the OK corral. Cattlemen used to do it all the time to fences and Doris Day would know how to handle that recalcitrant mule.

The ticket is now available and if you have a good look round, you'll find that rifle, too. Make sure, however, that you can match things up at the camp site.

The paddle will come in handy, inless you want to be up the proverbial creek.

The final clue I'll give is that it looks like Eskimos are sharing the reservation with the Indians.

The whole adventure is full of puns. For instance, you find a lot of tin-starred cats – *posse* cats.

The screen is split into several windows showing your location, objects visible, your input and the computer's response to it.

Your last 18 moves are also displayed on-screen.

The HELP facility really does help. On entering QUIT, you are presented with a menu giving you several options – quit, save game, and so on. Other software house please note.

There are no sudden deaths in the game. Something that is often the bane of the novice adventurer.

At the price, £2, I would be hard put to criticise it, even if I wanted to.

I don't like split-screen displays in my adventures, but that is a purely personal preference. For Nemesis's benefit, let me say that this adventure did amuse an old hand.

Overall, quite an easy adventure – but then that is of definite benefit to the novice. I consider it to be outstanding value for money, extremely witty, and definitely the best adventure for beginners that I have ever seen. Excellent.

Paul Gardner

Gorilla swings in with a double punch

MICRO Power has released a double game pack containing two of their most successful products, **Killer Gorilla** and **Gauntlet.**

In Killer Gorilla, a version of Donkey Kong, you must scale a series of iron girders to rescue a kidnapped maiden.

On your way, you encounter barrels which the Gorilla continuously rains down upon you – and you will die instantly if hit by one.

You can, however, jump over them, or alternatively snatch one of a couple of hammers to smash them to pieces.

Should you be successful in reaching the top of the screen, the Gorilla picks up your bird and takes off, and you must tackle him on the next screen.

There are four screens in all and they increase in difficulty to include conveyor belts carrying pies, and moving elevators.

The sound and graphics are good and the game is as addictive as ever.

Gauntlet is a version of the arcade favourite Defender, and a pretty good version it is, too. You fly your X15 fighter over mountainous terrain, defending, from invading forces, canisters which are vital to your existence.

You start the mission with four lives and four smart bombs.

The smart bombs come in very handy for destroying everything in sight – apart from the canisters – when you're in trouble.

Your lives diminish each time you are hit by one of the enemy or its ammunition.

There are six different types of alien, whose sole object is to see you off.

The most difficult to deal with are the mutants, who follow you relentlessly and are quite difficult to kill.

Others are Mine Layers, Crawlers, Killers, Cruisers and Buzzers – and they all have their individual characteristics.

Your X15 has but a single front laser, so you can only fire on the enemy directly ahead – but you can turn on a sixpence to fire in the opposite direction and zap the ones sneaking up on you from behind.

The long-range radar screen gives you ample warning of the presence of the enemy.

The original game was very exciting and tremendous fun to play, and this version is no exception. Excellent smooth scrolling graphics and good sounds will make it appeal to all Space Invader fans.

The six keys take some getting used to, but there is always a joystick option to fall back on if your fingers get tied in a knot.

All in all, this is a tremendous double bill that is an absolute must for arcade fans.

David Andrews

Programs are oh, so friendly...

A RELATED package on one cassette, **DFM Database 464** and **DFM Labels 464** from Dialog come with a demonstration file that can be used for either program.

Their comprehensive instructions are helpful, with examples where needed, while the programs are friendly to the point where they begin to annoy.

Each one is menu-driven and on loading the database the main menu presents nine options, one being for a utilities menu with six more.

Accessing a record gives a window at the bottom of the screen offering another 12, so one expects them to form a fairly powerful program and they do, yet I found it strangely restrictive.

For instance, there are two ways of outputting information to a printer, the first one being via the Reports option which allows selected records and fields to be printed. This produces nicely formatted results when the fields are short and few, but hits wrap-around troubles if they are longer or more numerous.

Using condensed mode can help, but for many files a wide printer would be needed to avoid a messy layout.

Further irritation was caused by the automatic appendage of "totals" at the bottom of each report, regardless of whether it contained numeric fields or not!

The second form of printout is that of a single record via the Access facility. Here, I lost the spacing between field labels and data when using files created by me, though not with the demo file.

Most likely this was due to a bug, which has probably been remedied by now.

Fields that contain between 24 and the limit of 36 characters won't fit on the screen when accessed, so an excrutiating sideways scroll is

utilised to view them.

A disciplined user should obtain about 200 records from a simple address book file, so for longer file requirements they may have to double up by using categories such as A-I, J-Q and R-Z for surname initials and so on.

This is the usual snag with cassette databases, which restricts their usage, even among small businesses and clubs, thus rather defeating the object in producing such a cassette.

As for the labels program, I found that it worked very efficiently, though one can only use files made with the database.

Individual addresses may be selected and printed in any quantity and it allows positioning for window envelopes – though only a single column of labels, lacking the tab facility for a double spread.

This means that to use two sheets of "two up" labels, rather than a roll of single ones, you have to feed the backing sheets through twice.

My inclination is to wait until a suitable ROM based database and mail-merge package comes along, especially in view of the price. This one might help those who cannot wait.

Ray Miller-Lawson

It's fun in the freezer

MR FREEZE, from Firebird, is a simple, enjoyable, but very frustrating ladders-and-levels type of game. Equipped with a flame-thrower and special thermal suit, your task is to de-ice the six compartments of a giant freezer.

The flame-thrower isn't for de-icing – it's to protect you from the many flying hazards and robot guards.

If you're hit in the face by a frozen steak-and-kidney pie or bump into a flying frozen fish carcase, you lose one of your six lives.

A quick blast with the flame-thrower sends the flying food temporarily back where it came from, while robots are stunned for a second, allowing you to run away.

In the top left corner of each compartment is a large button which when pressed de-ices it. You have to make your way up the ladders along the platforms to reach it, then all the way back down and on to the next room.

Fixed to the roof, but able to slide along, is a laser.

When you climb a ladder to the next plátform, it moves until it is directly above the ladder and fires a bolt which can turn you into a solid block of ice. If you hang about even for a fraction of a second, you've had it.

The game runs in Mode 1, so there aren't many colours, but the graphics are sharp and the characters quite well animated.

There's an excellent tune playing on two channels and sound effects on the third. The tune can be switched off at any time during the game.

There isn't a high-score table, which is a bit of a disappointment, but it does remember the best score. The keys chosen can easily be changed if they aren't to your liking.

Overall, it's quite fun to play and has that addictive quality that makes you want to have "just one more go". A good simple game in the Manic Miner tradition.

Roland Waddilove

REVIEWED SO FAR

Adventure Quest	Level 9
American Football	APS
Amstradraw	B.S. McAlley
Android One	Vortex
Arnold goes to somewhere else	Nemesis
Artwork	Kuma
Astro Attack	Amsoft
Amstrad Tutorial	Amsoft
Atomsmasher	Romik
Blagger	Alligata
Brawn Free	Nemesis
Centre Court	Epicsoft
Chopper Squad	Interceptor
Colossal Adventure	Level 9
Country Cottages	Sterling
Crystal Theft	Wiccasoft
Cubit	Mr Micro
Defend or Die	Alligata
Detective	Argus Press
DFM Database + Labels	Dialog
Dragon's Gold	Romik
'Er*bert	Microbyte
Fighter Pilot	Digital Integration
Flexi-friend	Camel
Flight Path 737	Anirog
Flight Simulation	Myrrdin
Football Manager	Addictive
Fruit Machine	Amsoft
Fruity Frank	Kuma
Forest at World's End	Interceptor
Galaxia	Kuma
Gems of Stradus	Kuma
Ghouls	Micro Power
Happy Letters	B.E.S.
Harrier Attack	Durell
Happy Writing	B.E.S.
Holdfast	Kuma
Home Accounts Manager	Dialog
Home Budget	Kuma
Hunchback	Ocean
Jet Boot Jack	English
Killer Gorilla/Gauntlet	Micro Power
Lords of Time	Level 9
Manic Miner	Software Projects
Map Rally	B.E.S.
Masterfile	Campbell
Message from Andromeda	Interceptor
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.
Punchy	Mr Micro
Pyjamarama	Microgen
Return to Eden	Level 9
Roland in the Caves	Amsoft
Roland Goes Square Bashing	Durell
Rollaball	Timeslip
Royal Quest	Timeslip
Screen Designer	DJL
Sir Lancelot	Melbourne House
Snooker	CDS
Snooker	Gem
Snowball	Level 9
Software Star	Addictive
Sorcery	Virgin
Splat	Incentive
Star Avenger	Kuma
Star Watcher	Amsoft
Stockmarket	Argus Press
Strip Poker	Knightsoft
Survivor	Anirog
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

Bomb brings a shock

BE prepared for a shock when you load **Time Bomb,** a new game from Black Knight Software. The title screen hits you right between the eyes.

You see, Black Knight has chosen to redesign the Amstrad's rather nice character set, replacing it with large scrawly letters resembling a right-handed person's attempt to write with his left.

They are almost unreadable – which is a pity, because the screen doubles as a high-score table as well.

The game itself is a version of the well-known Danger UXB. You play the part of a bomb-disposal expert who is up against the clock, stomping boots and mines as he strives to defuse a number of bombs before they explode.

The expert – a head complete with blue hard hat – starts from the top left of a 162 square grid.

A bomb appears at random on the screen and you have ten seconds to reach it before it explodes and takes one of your three lives.

You can only travel on the squares, which disappear behind you – making it harder to reach the bombs.

To make life a little easier, it is possible to scroll the row of blocks left or right and make a path to the bombs.

The program offers 10 skill levels, user-defined controls, and a choice of 10 frames from which to start. The graphics are large and bright and the sound capabilities of the Amstrad have been used well.

Sound effects include a ticking clock, loud explosions, and a tolling bell when you die. When a bomb explodes, the whole screen shakes.

Apart from the opening screen, Time Bomb is a well-presented program, but there wasn't enough variation to maintain my interest after a few tries.

The screens get harder and faster, but the format stays the same.

It's a good implementation of the arcade game, but alongside some of the latest Amstrad games it looks very dated.

Brian Finnerty

You can bank on this!

CONSIDERING pencil and paper versus loadtime, customers for **Home Accounts Manager,** on cassette from Dialog Software, are likely to be the "not-accounts-minded" seeking salvation.

The expense account has 20 user-definable headings, the ability to enter expenses into each heading, to view the details, and see the total expenditure.

Bank account makes entering cheques, standing orders, bank charges or the pennies you may receive, very simple. Note here that cheques (only) are automatically entered in the appropriate expense account.

Address file seems a bit pointless. It manupulates well, but the one-line Street field is too small and, when writing a cheque and envelope, you usually have the bill (with the address on it) in front of you (how would you know what amount to pay, otherwise?).

A mail order address might, I suppose, come in handy when you wanted to order another, or complain about the first.

Utilities reports memory left, calculates loans (a nice option – working out interest payable and monthly repayments), sets up for monitor-type – although colour is merely blue on cyan – and wipes expenses and addresses. A Save/Load facility deals with the data for next time.

Full marks for excellent error-trapping, print-out facility, ease of use, and all-round user friendliness – all spoiled by an inadequate knowledge of accountancy.

Although cheques entered in the bank section are automatically debited to the expense accounts, the other bank items are not. Many people pay their mortgage, electricity budget payments, and so on, by standing order and, if these are not included in the total expenses, then the total expense figure is useless.

If you must keep a note of all items entered on the bank account, other than cheques, and add these back to the program's total expenditure figure by hand to get the accurate figure, then you might as well do it all on paper in the first place.

Allowing direct entry into the expense account traps the unwary novice into duplicate entries and omissions and, what's worse, there is no control of cash expenditure.

What's needed is several bank accounts plus a cash account, each organised to automatically record all entries into the appropriate expense or income account, removal of the direct entry into expense account option, and a mini-balance sheet – total bank/cash opening balances, less total expense, plus total income equals total bank/cash closing balances.

At the moment, this package is a toy, well-made and with nice frills. But you don't buy a toy car to get to work.

Dorene Cox

Al's beat

ALAN McLACHLAN tries his hand at writing a simple games program

EVENIN' all. I've decided to have a break from debugging for a month because I fancied doing a bit of programming. I asked the lads in the office what I should write, and having thought better about Roland's suggestion of a super fast machine code levels-and-ladders, zap 'em chase 'em, mega production, I decided to take Mike's advice and write something extremely simple. (He must know me better than I thought.)

I wanted a program that a beginner could follow easily, but more important, one that I could write myself with very little help.

A couple of years ago I took a series of evening classes on Basic programming, and one week the class, with the teacher, put together a very simple Minefield game which contained a lot of useful programming ideas. So that's what I'm going to try to describe to you now.

The idea was simple: There were a number of mines hidden in a square grid, and by entering X,Y coordinates, you attempted to find them all in the least possible number of goes.

My work of art is a slight variation on that, consisting of a 10 by 10 grid of boxes in which are hidden 10 of our Amstrad favourites, the Smilies. You enter the coordinates of the square that you think he's in – column first, then row.

If you choose correctly, the Smiley is displayed accompanied by a perhaps suitable chorus of The Entrance of the Queen of Sheba or something similar. Should you choose wrongly, you could be greeted by an ear-splitting raspberry or similar noise, and given clues as to the whereabouts of the nearest target.

Well, that's it in theory – the main problem is starting to program it. I thought for a while and came to a monumental decision. I typed in:

```
10 REM Al's Smiley Hunt
```

No mug, this lad. If I don't manage to get the program working, at least I'll know what it's supposed to be.

Right, then, the first thing to do, and I don't know whether the experts do this, is to plan the program out on paper. So I wrote out a list of routines to carry out the following tasks:

1. Initialise Mode, arrays, colours.
2. Draw and set up grid.
3. Position Smilies.
4. Take input and check and validate.
5. Check input against 3.
6. Show Smiley if correct.
7. Give clue if incorrect.
8. Show how many guesses.

That's enough delay getting to the keyboard – it's about time we got stuck in. We're going to write the game in chunky Mode 0 so line 20 reads:

```
20 MODE 0
```

I thought for a long time how best to set up the two dimensional grid and be able to store values in the various squares.

There are several ways of doing this, and although we haven't yet covered the subject in our Beginners' series, I am going to use an array *box(9,9)* which is by far the most efficient way to solve the problem.

So line 30 reads:

```
30 DIM box (9,9)
```

The array reserves sufficient memory for 100 boxes, all accessible by the numeric variable *box*. The reason it is 100 locations and not 81 is that the array allows for 0 as X and Y. subscripts (the numbers in the brackets).

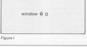


Figure I

You'll see why I have preferred to use the numbers 0-9, rather than 1-10 later, but for now just settle for the fact that it will be much easier in the long run.

It will make rather a nice display to split the screen into two parts, with the grid in one and the inputs in another. We covered the WINDOW command in the April issue, so we'll create a small window four lines deep at the top of the screen.

We'll use this window for the inputs and call it stream #1, leaving a larger one, stream #0, for the grid, as in Figure I.

We also want these two windows to have different background colours. The default colours in the Amstrad's Mode 2 are blue, yellow, cyan and red.

I want the large window to have a black background, so to achieve this we must change the INK of PEN 0 to 0 (Black). This is dealt with in line 40. Of course PAPER is already defaulting to PEN 0 so we don't need to even mention that.

```
40 WINDOW #0,1,20,6,25:INK 0,0
```

I want the smaller window to have a red background, so all we need to do is set the PAPER to 3 (red) and clear that screen/window, once again not forgetting the stream expression #1. Just as a matter of interest leave out the #1 in line 50 and see what happens.

```
50 WINDOW #1,1,20,1,5:PAPER #1,3:CLS
#1
```

During our game we are going to be using quite a few variables and where possible we'll try to give them names that mean something.

Two important variables are *turns* which I've chosen to signify how many guesses we've had, and *smileys* to indicate the total number of smiling faces we've found. It is important that these are set to zero when the game is re-run and we do this in line 60. Other variables will be initialised later or input from the keyboard.

```
60 turns=1:smileys=0
```

This completes the initialisation

stage and if you look back at our list you'll see that the next job is to draw the grid of boxes. You could show these boxes as any character you wanted, and even define one for yourself using the SYMBOL command. I've decided to keep it simple and use a shape from the Amstrad's character set – CHR$(233).

To put this character on screen in the form of a grid, we simply create two FOR ... NEXT loops containing the start and end locations of the screen coordinates, and PRINT the character. I've called this as a subroutine using line 70.

By the way, don't worry at this stage about any big gaps in line numbers as we're going to tie in the subroutines to the lines that call them. We can renumber the program once it is complete.

```
70 GOSUB 700 :REM draw/init grid
```

The subroutine at 700 starts with a REM statement to identify it and is followed by two FOR ... NEXT loops to PRINT the boxes. The first loop generates the X location of each box, the second loop generates the Y location.

These FOR ... NEXT loops are also used to place 0s in all the locations in our array *box( )* using line 725.

```
700 REM draw grid
705 CLS #0
710 PEN #0,1:FOR boxx = 5 TO 14:FOR b
oxy = 6 TO 15
720 LOCATE #0,boxx,boxy:PRINT #0,CHR$
(233)
725 box(boxx-5,boxy-6)=0
730 NEXT boxy:NEXT boxx
```

Remember, if you are going to identify your NEXTs by following them with variable names, you must put them in reverse order to the FORs, as in line 730. I didn't the first time I typed the line in and got the error message "Unexpected NEXT in 730".

You can avoid this slip by not identifying the NEXTs at all, but it makes it easier to read programs if you do.

You can check you've done this right so far by entering a dummy line to suppress the "Ready" prompt as in line 75. Also enter a Return at line 780.

```
75 GOTO 75
780 RETURN
```

Run the program and you should now see in the lower window 100 yellow boxes in 10 rows of 10 on a black background.

If you haven't, check everything you've done so far very carefully.

We now need to put in the numbers for the X and Y coordinates and it was in this little routine that the Amstrad caused me some difficulties.

We're going to print the numbers 0-9 over the columns, and also down the left hand side. We do this by locating the X and Y coordinates and printing the FOR ... NEXT variables *numx* and *numy*, with 5 and 6 subtracted from them respectively to give the numbers.

The reason I avoided the number 10, by the way, is because it has two digits and that would have made the screen untidy.

To demonstrate my problem, just try my original line 740 which includes a change of PEN to 2 (white).

```
740 FOR numx=5 TO 14: PEN #0,2:
LOCATE #0,numx,4: PRINT #0, numx-5;:
NEXT numx
```

The bit that left me floundering was the way the Amstrad prints its characters – always preceded and followed by a space. That's why there's nothing left of the numbers but one 9 – each successive number has wiped out its predecessor.

I must confess I hadn't realised the cause of this and after several minutes of banging the desk-top, I asked my mate Roland for help. Needless to say, he immediately dropped his copy of the Sun and rushed to my aid, beads of perspiration dripping from his brow.

Now, I'm not going into detail about the solution – that's for the Beginners' series. But the answer is to use the command MID$ to strip off either or both spaces as required.

So replace the *numx-5* in line 740 with *MID$(STR$(numx-5),2)*. This makes a string of *numx-5* and then removes the first space. Line 740 should now read:

```
740 FOR numx=5 TO 14:PEN #0,2:LOCATE
#0,numx,4:PRINT #0,MID$(STR$(numx-5),
2);:NEXT numx
```

and add lines 750 to 775 to complete the subroutine.

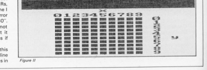
```
750 FOR numy= 6 TO 15:LOCATE #0,16,nu
my:PRINT MID$(STR$(numy-6),2,1);:NEXT
 numy
760 PEN #0,1:LOCATE #0,10,2:PRINT #0,
 "x"
770 LOCATE #0,18,11:PRINT #0,"y"
775 FOR count=1 TO 1000:NEXT count
```

Run the program as you have it now and you should see on the screen, in addition to 100 yellow boxes, the numbers 0 to 9 running across the top of the columns and also down the right-hand side, with a small x and y showing the axes, as in Figure II.

If your screen differs in any way, check your typing for errors, including all punctuation marks, and – most important – the semi-colons in lines 740 and 750 which cause the numbers to be printed one after the other rather than on separate lines.

Well I think you've had enough for one session. Next month we'll look at how to hide the Smileys, progress on to the input routine, and check for any correct or incorrect guesses.

Who knows, we even may finish the game completely!



```
          x
    0123456789
                  0
                  1
                  2
                  3
                  4   y
                  5
                  6
                  7
                  8
                  9
```

Figure II

MYRDDIN

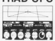
3D LANDMARKS
YOU CAN FLY AROUND

SUPERB REAL
TIME SIMULATION

MYRDDIN FLIGHT SIMULATION

AMSTRAD CPC 464

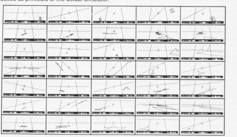
MANUAL



FULL SCREEN DISPLAY

FLIGHT CHART

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



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

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

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

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

3D landmarks you can fly around.

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

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

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

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

Name ....

Address ....

.... Postcode ....

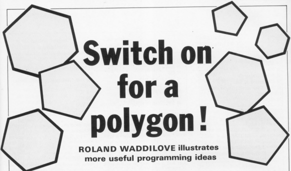
Cheque enclosed for cassette £11.95/£15.95 disc (inc. P.P.)
OR Debit my Access A/C No:-

□□□□ □□□□ □□□□ □□□□

OR Telephone through your Access Order.

Signed...

CWA 8



Switch on for a polygon!

ROLAND WADDILOVE illustrates more useful programming ideas

POLYGONS is a fairly short and simple program which produces quite an impressive display and illustrates some useful programming techniques.

After selecting the number of sides, a polygon is drawn which tumbles and spins about its horizontal and vertical axes.

The polygon is constantly being drawn, erased, rotated and re-drawn to produce a good animated display. Unfortunately, the only way to achieve fast-enough animation is through the use of machine code, though this has been kept to a minimum.

The best way to explain how the program works is to show how it was developed from a very simple idea. There are several short programs to type in at each stage, so switch your Amstrad on and let's first try to draw an ellipse.

Casting my mind back to my school days I can dimly recall that the coordinates of any point on the circumference of an ellipse is minor*COS(theta), major*SIN(theta). Where major and minor are the axes and theta is the angle. Figure I shows it a bit more clearly.

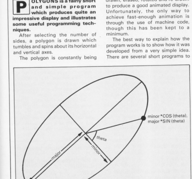


Figure I: Ellipse structure

Program I attempts to draw such

```
10 REM PROGRAM I
20 MODE 1
30 DEG
40 major=72:minor=36
50 FOR theta=0 TO 360
60 DRAW minor*COS(theta),major*SIN(th
eta)
70 NEXT
```

Program I

an ellipse. Theta is used as a loop counter, running from 0 to 360 degrees and each point is joined using draw commands. Run it and see what happens.

It's not quite right is it? The problem is that it's drawn around the origin in the bottom left-hand corner of the screen.

Either the origin can be moved, or a constant can be added to all the coordinates to get round this. ▶

```
10 REM PROGRAM II
20 MODE 1
30 DEG
40 major=72:minor=36
50 FOR theta=0 TO 360
60 DRAW 320+minor*COS(theta),200+majo
r*SIN(theta)
70 NEXT
```

Program II

Program II uses this second method to draw the ellipse in the centre of the screen. 320 is added to the x coordinate and 200 to the y.

It's better, but still not quite right as there's a line drawn from the origin to the ellipse. What we forgot to do is to move to the first point before we draw anything.

```
10 REM PROGRAM III
20 MODE 1
30 DEG
40 major=72:minor=36
50 MOVE 320+minor*COS(0),200+major*SI
N(0)
60 FOR theta=0 TO 360
70 DRAW 320+minor*COS(theta),200+majo
r*SIN(theta)
80 NEXT
```

Program III

Program III adds the necessary MOVE command. The program now draws a perfect ellipse and we can start working on it.

Try altering the size of the major and minor axes and see what happens. Notice that when they are equal you get a circle.

It's very slow at drawing the ellipse, so try altering the step size in the FOR ... NEXT loop. Add:

```
STEP 60
```

to the end of line 60 in Program III and run it again.

The ellipse has become a hexagon. It has six sides because 360, the loop limit, divided by 60, the step size is 6. The step size alters the number of sides so we can draw an *n* sided polygon using a step size of 360 divided by *n*.

```
10 REM PROGRAM IV
20 MODE 1
30 DEG
40 major=72:minor=36
50 WHILE dozed.off=false
60 FOR angle=0 TO 72 STEP 6
70 CLS
80 MOVE 320+minor*COS(angle),200+majo
r*SIN(angle)
90 FOR theta=angle+72 TO 360+angle ST
EP 72
100 DRAW 320+minor*COS(theta),200+maj
or*SIN(theta)
110 NEXT
120 NEXT
130 WEND
```

Program IV

Suppose we draw the polygon with the loop counter *theta* running from 0 to 360. Then if we erase it and draw it again with *theta* running from 6 to 366 it will appear to have rotated through six degrees.

Try altering the loop in line 60 of Program III and see what happens. Just add a constant on to the start and finish of the FOR ... NEXT loop.

Take a look at Program IV. What it does is to alter the start and finish of the loop which draws the polygon by the value of *angle*, clearing the screen every time the polygon is drawn. An outer loop is used to increment *angle* by 6 each time the polygon is drawn.

This short program now draws a rotating polygon. It's very flickery but we can get round that using machine code in the final program.

The polygon is only rotating about one axis at the moment. To make it appear to rotate about its other axis, what we do is to introduce another loop outside the two we've already got. In Program V this outer loop varies the size of the major axis. It now rotates about two axes.

We've got everything we need now to produce a fast animated display of a rotating polygon. Any polygon can be drawn at any angle with any degree of rotation using this last program. Program VI, Rotating Polygons, was developed from this.

The Basic part of Polygons calculates the coordinates of the corners of the polygon and stores them in the memory. Trigonometric calculations take a relatively long time, so working out the coordinates beforehand greatly improves performance.

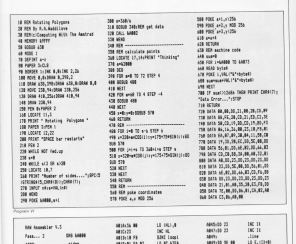
The polygon is rotated through a small angle and the new coordinates of the corners calculated and stored after the first. This is repeated until the polygon is back in its original position. This takes a long time, so be patient!

Once all the coordinates are in the memory, a machine code routine runs through the data moving, drawing and erasing the polygon. It's very fast, displaying several frames a second, mainly because it doesn't need to do any calculations. It's a technique worth remembering. For those of you who'd like to unravel it, an assembler listing is given in Figure II.

As a project, how about trying to write a routine that will rotate a prism such as a pyramid? Use the same method as I've described here. It's not as hard as it might seem – and quite interesting, too.

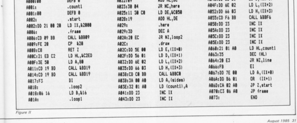
```
10 REM PROGRAM V
20 MODE 1
30 DEG
40 major=72
50 FOR minor=-major TO major STEP 4
60 FOR angle=0 TO 72 STEP 12
70 MOVE 320+minor*COS(angle),200+majo
r*SIN(angle)
80 FOR theta=angle+72 TO 360+angle ST
EP 72
90 DRAW 320+minor*COS(theta),200+majo
r*SIN(theta)
100 NEXT
110 CLS
120 NEXT
130 NEXT
```

Program V



```
10 REM Rotating Polygons
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 MEMORY &9FFF
50 GOSUB 630
60 MODE 1
70 DEFINT a-z
80 PAPER 3:CLS
90 BORDER 1:INK 0,0:INK 2,26
100 MOVE 0,0:DRAW 0,398,2
110 DRAW 638,398:DRAW 638,0:DRAW 0,0
120 MOVE 230,94:DRAW 230,256
130 DRAW 410,256:DRAW 410,94
140 DRAW 230,94
150 PEN 0:PAPER 2
160 LOCATE 11,3
170 PRINT " Rotating  Polygons "
180 PAPER 3:PEN 1
190 LOCATE 12,22
200 PRINT "SPACE bar restarts"
210 PEN 2
220 WHILE NOT fed.up
230 s=0
240 WHILE s<3 OR s>20
250 LOCATE 10,7
260 PRINT "Number of sides....";SPC(5
)STRING$(5,CHR$(8));CHR$(7);
270 INPUT n$:s=VAL(n$)
280 WEND
290 POKE &A000,s+1
300 s=360/s
310 GOSUB 340:REM get data
320 CALL &A002
330 WEND
340 REM --------------------------
350 REM calculate points
360 LOCATE 17,14:PRINT "Thinking"
370 a=&2000
380 DEG
390 FOR m=0 TO 72 STEP 4
400 GOSUB 480
410 NEXT
420 FOR m=68 TO 4 STEP -4
430 GOSUB 480
440 NEXT
450 x=0:y=0:GOSUB 570
460 RETURN
470 REM --------------------------
480 FOR i=0 TO s-6 STEP 6
490 x=320+m*COS(i):y=175+75*SIN(i):GO
SUB 570
500 FOR j=i+s TO 360+i+s STEP s
510 x=320+m*COS(j):y=175+75*SIN(j):GO
SUB 570
520 NEXT
530 NEXT
540 RETURN
550 REM --------------------------
560 REM poke coordinates
570 POKE a,x MOD 256
580 POKE a+1,x\256
590 POKE a+2,y MOD 256
600 POKE a+3,y\256
610 a=a+4
620 RETURN
630 REM machine code
640 sum=0
650 FOR i=&A000 TO &A073
660 READ byte$
670 POKE i,VAL("&"+byte$)
680 sum=sum+VAL("&"+byte$)
690 NEXT
700 IF sum<>12686 THEN PRINT CHR$(7);
"Data Error...":STOP
710 RETURN
720 DATA 00,00,DD,21,00,20,CD,09
730 DATA BB,FE,20,C8,21,ED,C2,3E
740 DATA 50,CD,19,BD,CD,19,BD,F3
750 DATA 06,16,36,00,23,10,FB,01
760 DATA EA,07,09,30,04,11,50,C0
770 DATA 19,3D,20,EC,DD,5E,00,DD
780 DATA 56,01,DD,6E,02,DD,66,03
790 DATA CD,C0,BB,3A,00,A0,32,01
800 DATA A0,DD,23,DD,23,DD,23,DD
810 DATA 23,DD,5E,00,DD,56,01,DD
820 DATA 6E,02,DD,66,03,CD,F6,BB
830 DATA DD,23,DD,23,DD,23,DD,23
840 DATA 21,01,A0,35,20,E3,FB,DD
850 DATA 7E,00,DD,B6,01,CA,02,A0
860 DATA C3,06,A0,00
```

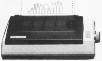
Program VI



```
RAW Assembler V.3

Pass... 2          ORG &A000

A000:              .sides
A000:00            DEFB 0
A001:              .count1
A001:00            DEFB 0
A002:              .start
A002:DD 21 00 20   LD IX,&2000
A006:              .frame
A006:CD 09 BB      CALL &BB09
A009:FE 20         CP  &20
A00B:C8            RET Z
A00C:21 ED C2      LD HL,&C2ED
A00F:3E 50         LD A,80
A011:CD 19 BD      CALL &BD19
A014:CD 19 BD      CALL &BD19
A017:F3            DI
A018:              .loop2
A018:06 16         LD B,&16
A01A:              .loop1
A01A:36 00         LD (HL),0
A01C:23            INC HL
A01D:10 FB         DJNZ loop1
A01F:01 EA 07      LD BC,&7EA
A022:09            ADD HL,BC
A023:30 04         JR NC,here
A025:11 50 C0      LD DE,&C050
A028:19            ADD HL,DE
A029:              .here
A029:3D            DEC A
A02A:20 EC         JR NZ,loop2
A02C:              .draw
A02C:DD 5E 00      LD E,(IX+0)
A02F:DD 56 01      LD D,(IX+1)
A032:DD 6E 02      LD L,(IX+2)
A035:DD 66 03      LD H,(IX+3)
A038:CD C0 BB      CALL &BBC0
A03B:3A 00 A0      LD A,(sides)
A03E:32 01 A0      LD (count1),A
A041:DD 23         INC IX
A043:DD 23         INC IX
A045:DD 23         INC IX
A047:DD 23         INC IX
A049:              .line
A049:DD 5E 00      LD E,(IX+0)
A04C:DD 56 01      LD D,(IX+1)
A04F:DD 6E 02      LD L,(IX+2)
A052:DD 66 03      LD H,(IX+3)
A055:CD F6 BB      CALL &BBF6
A058:DD 23         INC IX
A05A:DD 23         INC IX
A05C:DD 23         INC IX
A05E:DD 23         INC IX
A060:21 01 A0      LD HL,count1
A063:35            DEC (HL)
A064:20 E3         JR NZ,line
A066:FB            EI
A067:DD 7E 00      LD A,(IX+0)
A06A:DD B6 01      OR  (IX+1)
A06D:CA 02 A0      JP Z,start
A070:C3 06 A0      JP frame
A073:              END
```

Figure II

Yet another unbeatable deal from Datastar Systems!



THE *INCREDIBLE* NEW STAR SG-10 NEAR LETTER QUALITY PRINTER.

features:—

- Epson and IBM compatible print codes
- Friction and tractor feeds.
- Externally Accessible DIP-Switches
- High-resolution bit-image graphics
- Downloadable character sets
- NLQ, italics, condensed, enlarged, super & subscript styles.
- High 120 CPS print speed, logic seeking.
- Hex dump facility.
- Macro-instructions.
- Touch-sensitive control panel.
- Standard parallel Centronics interface.
- 2K print buffer — expandable to 10K
- Commodore 64 version available shortly

STAR SG10

Character Sets

96 Standard ASCII Characters
88 Standard International Characters
96 Italic Characters
88 Italic International Characters
96 NLQ Characters
88 NLQ International Characters

Character Matrix

Standard Characters: 9 dot × 11 dot
STAR Block Graphics: 6 dot × 6 dot
IBM Block Graphics: 12 dot × 11 dot
User Defined: 8 dot × 4 to 11 dot
Near Letter Quality: 17 dot × 11 dot

Character Fonts

Normal (10 CPI)
Elite (12 CPI)
Condensed (17 CPI)
NLQ (10 CPI)

Line Spacing $\frac{1}{6}$, $\frac{1}{8}$ in. or $\frac{7}{72}$ in. Standard

Dimensions (W × D × H)
392 × 315 × 145mm (15.4 × 12.4 × 5.7 in.)

Number of Columns
80, 96, 136, (40, 48, 68 Double Width)

Print Head 9 Wires (User Replaceable)

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!!!

Post your cheques to

Datastar Systems UK

Unicom House, 182 Royal College Street, London NW1 9NN. 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*

Credit Card Hot Line 01-482 1711

AS MAIN DISTRIBUTORS FOR THE STAR RANGE, WE ARE ABLE TO OFFER EXCELLENT DEALER MARGINS

SPECIAL reader offer!

Yearly Profits
Yearly Profits
dk'tronics
dk'tronics
dk

GRAPHICS LIGHTPEN

For AMSTRAD CPC464

-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 77.

SUPERCHARGE YOUR A

'SUPERPOWER' SIDEWAYS ROMCARD FOR THE AMSTRAD(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.

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

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

- Matching Case, with easily detachable cover.
- Fits snugly to rear of computer.
- Bus extension for fitting of Disk Interface or other units.
- Houses up to 7 foreground or background Roms – any mix of 8K and 16K.

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

HIGH QUALITY BLANK EPROMS AVAILABLE

These are 200 nanosecond devices, the speed recommended by AMSOFT. 8K Eprom £7.95 (Ref B104). 16K £11.95 (Ref B105).

SUPERPOWER SIDEWAYS ROMCARD (Ref B101) **£39.95**

'SUPERPOWER' ROM-BASED SOFTWARE

A complete range of Rom-based programs is under development and will be released as follows:–

- Mailing-List/Club Membership; Disk User's Utilities NOW AVAILABLE.
- Programmer's Toolbox END of JULY.
- Assembler/Disassembler/Machine-code Monitor EARLY AUGUST.
- Word Processor; Database END of AUGUST.
- Spreadsheet; Graphics/Statistics END of SEPTEMBER.

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

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

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

When used as Membership List, 12 can be nominated for monthly subscription reminders.

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

Program works with any parallel printer. Alphabetical Order is dealt with Entry. User choice of **keyword** e.g. 'J'ohn or 'S'mith. The Name field can be searched for the first part or the whole of a keyword. The whole file can also be searched for any string. There is sophisticated line and character editing, including change of keyword. Function Keys can be defined to give single key entry of commonly used string e.g. 'Membership No.' In Entry mode an automatically incrementing number is available. Foreground and Background colours can be selected.

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

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

The program also contains functions of use to the assembly language programmer.

Main Functions

- FILELOADS – loads first sector into buffer and remainder into memory for fast access later.
- READ – reads a sector into the buffer and enters Edit Mode.
- EDIT – displays the current buffer. Data displayed is Buffer Address, Hexadecimal representation of each byte and ASCII representation of each byte.

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

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

– comprehensive cursor controls for easy editing.

- FIND – can search a sector or total file for an ASCII string.
- WRITE – writes a sector previously read by READ or FILELOAD.

Subsidiary Functions

- CATALOGUE – similar to AMSDOS catalogue.
- MODE – Select 40 col./12 line display or 10/12, 40/24 or 80/24.
- INK – Select Background and Foreground Colours.
- ROM CHECK – lists all sideways Roms, giving Position, Foreground or Background, Name Version No. etc.

Other Disk Commands

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

ASSEMBLY PROGRAMMER'S AIDS

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

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 8104)	£39.95
SUPERPOWER ASSEMBLER, DISASSEMBLER & MACHINE-CODE MONITER (Ref B 105)	£39.95

'SUPERPOWER' PROGRAMMER'S TOOLBOX (REF B104)

This program provides a treasure chest of additional BASIC commands and functions, some of which are effectively provided in the CPC 664, as well as a large number of Programmer's Aids.

ADDITIONAL BASIC

- TURTLE – provides 'logo-like' turtle graphics when suffixed by a string of data in specified format.
- TUNE – plays music when suffixed by a string of data in specified format.
- CIRCLE – draws a circle from specified centre co-ordinates and radius.
- FILL – starting from a point, will fill an area surround by the foreground colour.
- GCOL – permits AND, OR and XOR with existing colours. (Simulates extra parameter of 664 graphics command).
- GRAPHICS PEN/GRAPHICS PAPER – select colour of graphics and background.
- COPYCHR – reads a screen character.
- ECHO ON/ECHO OFF – switches printer output on and off.
- PAGE ON/PAGE OFF – switches paged screen output on and off.
- CLEAR INPUT – clears input buffer.

PROGRAMMER'S AIDS

- EDITOR – provides 2 additional windows for the examination and editing of a Basic program.
- FIND «and REPLACE» – finds all occurrences of a string and optionally replaces with another string.
- XREF – lists all line nos. containing a particular line reference no. (GOTO'S, GOSUB'S etc.)
- COMPACT – removes lines starting with REM statements.
- FRAME – synchronises screen update with frame fly-back.
- PRINTER – selects Epson-compatible or Amstrad printer for dump routines.
- FDUMP – fast two-tone screen dump, with user-definition of colours to be treated as 'pen' and 'paper.'
- CDUMP – shaded screen dump depicting up to 9 colours.
- PLOAD – loads program saved under 'P' option.
- INFO – gives details on a specified disk file.
- MEDIT – comprehensive memory editor, displaying HEX and ASCII and enabling examinations of ROMS and RAM and also RAM modification.
- HELP – lists all commands and functions, together with their parameters.

In addition there are routines to design and generate characters, to convert screen characters into data for insertion into a Basic line and for defining a 'Mask' for the drawing of lines.

The program will be available in late July.

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

This suite of routines represents the complete Development Package for the Amstrad Z8Ø 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 Z8Ø 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.

The program will be available in early August.

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 Manager, Sheepscar House, Sheepscar Street South, LEEDS LS7 1AD. TELEPHONE (0532) 434006. State your NAME and ADDRESS and the REFERENCE NUMBERS of the products you wish to purchase.

Buy on Access/Visa card by stating your card number or write enclosing a cheque made payable to: MicroPower Ltd. (Please add 95p to your full order amount to cover post and packing.) (No cheques received will be banked before your order is despatched.)

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

ROM-BASED SOFTWARE FULFILS THE PROMISE OF YOUR AM

R AMSTRAD.

The following computer specialists stock the SuperPower Rom-card and Rom-based software for the Amstrad and are able to offer you advice and information on current products and the ever-expanding range:–

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

BEDFORDSHIRE
• Chilton Computers, Luton 0582 455684
• Hobbyte Limited, Luton 0582 457195

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
• Open Channel, Exeter 0392 218187
• Computerware, Exeter 0392 210507

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

HAMPSHIRE
• Electronequip, Fareham 0329 230671
• Alton Computers, Alton 0420 87998

HUMBERSIDE
• Computing World, Beverley 0482 881831

KENT
• The Village House of Computers, Bromley 01 460 7122
• Microway Computers, Rainham 0634 376702

LANCASHIRE
• Computer World, Bolton 0204 494304
• IMO Computers, Burnley 0282 54299

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 01580 3702

LINCOLNSHIRE
• MKD Computers, Lincoln 0522 25907

MANCHESTER
• Mighty Micro, Fallowfield 061 224 8117

MERSEYSIDE
• MicroByte, Wallasey 051 630 6933

MIDDLESEX
• JKL Computers, Uxbridge 0895 51815

NORFOLK
• Abacus, Norwich 0603 614441

NORTHERN IRELAND
• Botanic Computer Centre, Belfast 0232 240798
• Fun-cational Computers, Co. Armagh 07622 28543

NOTTINGHAM
• G A Computers, Ruddington 0602 211225
• Computer Market, Nottingham 0602 586454

STAFFORDSHIRE
• Leek Computer Centre, Leek 0538 387859

SURREY
• Computerama, Burgh Heath 073 73 54717

SUSSEX
• Gamer, Brighton 0273 698424
• Computermania, Chichester 0243 779263
• Computermania, Littlehampton 0903 716187

WALES
• Gwynedd Computers, Gwynedd 0286 2610
• Cardigan Electronics, Dyfed 0239 614483
• Bucon Limited, W Glamorgan 0792 467980

WILTSHIRE
• Barbury Computers, Swindon 0793 611487

YORKSHIRE
• MicroPower, Leeds 0532 458800
• Danum Computer Systems, Doncaster 0302 20355
• The Computer Room, Barnsley 0226 204000
• Just Micro, Sheffield 0742 752732

SUPER POWER

R AMSTRAD.

SERIOUS SOFTWARE
FROM THE MICROPOWER
ORGANISATION

SHEKHANA COMPUTER SERVICES

SIMPLY THE CHEAPEST FOR AMSTRAD GAMES

	RRP	Our Price		RRP	Our Price
Ghostbusters	10.99	8.50	Hunchback II	8.95	6.75
American Football	9.99	8.00	Daley Toms Decatalon	8.95	6.75
Sorcery	8.95	6.95	Kong Strikes Back	8.95	6.75
Master Chess	8.95	6.40	Steve Davis Snooker	7.95	5.90
Blagger	8.95	7.50	Classic Racing	8.95	6.95
Dark Star	7.95	6.40	Technician Ted	7.95	6.40
Gillians Gold	6.95	5.50	Combat Lynx	8.95	6.95
Flight Path 737	6.95	5.50	Forest at Worlds End	6.00	4.50
Football Manager	7.95	6.40	Message from Andromeda	6.00	4.50
Sir Lancelot	6.95	5.50	Jewels of Babylon	6.00	4.50
Fighter Pilot	8.95	• 6.95	Heroes of Karn	6.00	4.50
Jet Set Willy	8.95	6.95	Chopper Squad	6.00	4.50
Manic Minor	8.95	6.95	Azimuth (Head Alignment)	8.95	6.95
The Hobbit	14.95	11.95	Flight Sim (Myrddin)	11.95	9.99
Handicap Golf	6.95	5.95	Guide to Basic Part I	19.95	18.00
Pyjamarama	8.95	6.40	Artist (CRL)	9.95	7.50
Minder	9.95	7.50	One on One	8.95	7.50
Beachhead	9.95	7.95	Street Hawk	8.95	6.75
Raid Over Moscow	9.95	7.95	Superpipeline II	8.95	7.50
Star Avenger	6.95	5.95	*Herbert's Dummy Run	9.95	7.50
Smugglers Cove	5.95	4.99	Daley Toms Supertest	8.95	6.75
Every Ones A Wally	9.95	7.50	Mr Freeze		2.50
Alien 8	9.95	7.50	Arabian Knights	6.00	4.50
Airwolf	8.95	7.50	Warlords	6.00	4.50
*Frank Brunos Boxing	8.95	7.50	Pitman Typing Tutor	9.95	8.00
Dun Darach	9.95	7.50	Frankenstein	8.95	6.95
Hard Hat Mac	8.95	7.50	Forth (Interceptor)	12.99	9.99
Archon	10.95	9.99	*Pole Position	9.99	8.95
Bridge Player (Kuma)	6.95	5.95	Way of Exploding Fist	9.99	7.95
Chukie Egg	7.90	6.50	*Elite		POA
3D Star Strike	6.95	5.50	Knight Lore	9.95	7.50
Hypersports	8.95	6.95	Red Arrows	8.95	7.50
Emerald Isle	6.95	5.75	Roland in Time	8.95	6.95
Mini Office	5.95	4.90	Sherlock Holmes	14.95	11.99
Scrabble	9.95	8.95	Battle for Midway	9.95	8.00
Monopoly	9.95	8.95	Tank Busters	7.95	6.50
* Lords of Midnight	9.95	7.50	Millionaire	6.95	5.50
Hurricane Higgins Pool	9.95	7.50	Killer/Gorilla	9.95	7.99
Hurricane Higgins Snooker	9.95	7.50	Osprey	9.95	7.99
Fantasia Diamond	7.95	6.40	Test Match	6.95	5.50
Animated Strip Poker	8.95	7.25	Roland in Space	8.95	6.95
All Level 9 Games	9.95	7.95	Pinball Wizard (CP)	9.95	7.95
Zen Assembler (Kuma)	19.95	18.00	Database (Kuma)	14.95	13.95
* A View to Kill	10.99	8.50	Quickshot II - Joystick	12.95	8.99

Cheques/P.O. To: **SCS (DEPT CAM1) 653 GREEN LANES N8 0QY LONDON**
(MAIL ORDER ADDRESS ONLY). Tel: 01-800 3156 (S.A.E. FOR LIST)
Europe Add £1.00 per tape. Elsewhere at cost.
Or come and pick up your discount software direct from our shop at Marbles Shopping Centre Unit 11, 527-531 Oxford St. London W1R 1DD. (1 Min from Marble Arch Tube Stn.) Open 7 days a week from 10am-7pm. Please bring copy of advert to our shop for above discounts. *Please ring to confirm availability.

POOLSWINNER

THE ULTIMATE POOLS PREDICTION PROGRAM

Boxed, with detailed instruction booklet

- **MASSIVE DATABASE** Poolswinner is a sophisticated Pools prediction aid. It comes complete with the largest database available – 22000 matches over 10 years. The database updates automatically as results come in.
- **PREDICTS** Not just SCOREDRAWS, but AWAYS, HOMES and NO SCORES.
- **SUCCESSFUL** SELEC guarantee that Poolswinner performs significantly better than chance.
- **ADAPTABLE** Probabilities are given on every fixture – choose as many selections as you need for your bet. The precise prediction formula can be set by the user – you can develop and test your own unique method.
- **SIMPLE DATA ENTRY** All team names are in the program. Simply type in the reference numbers from the screen. Or use FIXGEN to produce fixture list automatically (see below).
- **DISC/MICRODRIVE COMPATIBLE** Tapes supplied with conversion instructions.
- **PRINTER SUPPORT** Full hard copy printout of data if you have a printer.

AVAILABLE FOR Spectrum (48K), Commodore 64, VIC 20 (+16K), AMSTRAD, BBC B, Atari (48K), ZX81 (16K), Dragon, Apple II, ELECTRON

PRICE £15.00 (all inclusive)

NOW AVAILABLE

FIXGEN 85/6

AT LAST: No more struggling for hours to get the fixture list into the computer. FIXGEN has been programmed with all English and Scottish fixtures for 1985/6. Simply type in the date, and the full fixture list is generated in seconds. Fully compatible with Poolswinner.

POOLSWINNER with FIXGEN £16.50 (all inclusive)

COURSEWINNER V3

THE PUNTERS COMPUTER PROGRAM

NOT JUST A TIPSTER PROGRAM, Coursewinner V3 can be used by experts and occasional punters alike.

You can develop and test your own unique winning system by adjusting the analysis formula, or use the program in simple mode. Coursewinner V3 uses statistical analysis of major factors including past form, speed ratings, course statistics, prize money, weight, going, trainer and jockey form etc, etc. It outputs most likely winners, good odds bets, forecasts, tricasts etc. The database includes vital course statistics for all British courses. You can update the database – never goes out of date.

AVAILABLE FOR Spectrum (48K), Commodore 64, BBC (B), AMSTRAD, Atari (48K), Apple II

PRICE £15.00 (all inclusive) includes Flat AND National Hunt versions.

Send Cheques/POs for return of post service to . . .

phone 24 hrs — selec SOFTWARE — phone 24 hrs

37 COUNCILLOR LANE, CHEADLE, CHESHIRE. ☎ 061-428 7425

(Send for full list of our software)

Amstrad Analysis

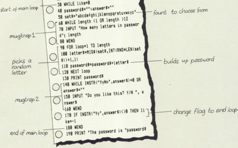
Password Generator

Analysed by Trevor Roberts

ONE of the first things you might like to try when you have created your first masterpiece is to protect it with a password routine. It sounds simple to think of a password but it can be surprisingly difficult.

You have to come up with a word that's not only memorable but also too hard for someone else to guess.

Password Generator is one way of overcoming the problem. Just run the program until you are presented with a word you can remember but which no one else is likely to guess.



```
10 REM PASSWORD GENERATOR
20 REM TREVOR ROBERTS
30 WHILE like=0
40 password$="":answer$=""
50 set$="abcdefghijklmnopqrstuvwxyz"
60 WHILE length <1 OR length >12
70 INPUT "How many letters in passwor
d"; length
80 WEND
90 FOR loop=1 TO length
100 letter$=MID$(set$,INT(RND*LEN(set
$))+1,1)
110 password$=password$+letter$
120 NEXT loop
130 PRINT password$
140 WHILE INSTR("YyNn",answer$)=0 OR
answer$=""
150 INPUT "Do you like this? Y/N ", a
nswer$
160 WEND
170 IF INSTR("Yy",answer$)<>0 THEN li
ke=-1
180 WEND
190 PRINT "The password is "password$
```

10,20 Title the listing and name the person responsible.

30-180 These lines form the major WHILE ... WEND loop of the program. Using the variable *like* as a flag, the loop keeps on cycling while *like* is false (0). Each time through a different password is produced.

40 This line initialises two variables, setting each to the null, or empty, string each time round the main loop. Leave it out and see what happens.

50 *set$* holds the complete set of letters that the program will choose from to make the password. In this case it's the alphabet but the more cryptically minded may like to use other combinations of characters.

60-80 These form a mugtrap. The user is prompted to enter the number of letters wanted in the password. The surrounding WHILE ... WEND loop only allows the answer to lie between 1 and 12.

90-120 This FOR ... NEXT loop cycles once for each letter of the password.

100 A mammoth line! All it does is select a letter at random from *set$* and store the result in *letter$*. Try doing the same thing with Ascii codes.

110 Each time round the loop the letter in *letter$* is added to the end of *password$*.

130 Displays the newborn password.

140-160 Another mugtrap. It asks whether the user accepts the password and stores the result in *answer$*. The conditions of the WHILE ... WEND loop ensure that only the prompted answers are accepted.

170 If the answer is yes (Y or y) then *like* is set to −1 and the main loop comes to an end. If the answer was N or n, *like* stays the same.

190 When the program drops out of the loop the chosen password is displayed.

An eventful way to get your kicks...

ROLAND WADDILOVE discusses the use of interrupts from machine code

AN interrupt is a signal sent to the Z80, the microprocessor at the heart of the Amstrad, informing it that its attention is required elsewhere immediately.

It stops what it is currently doing, carries out the process needing attention, then returns to its previous task, carrying on where it left off.

Interrupts are used for many things. Perhaps the most obvious one is the internal clock, read by the Basic pseudo-variable TIME. Every 300th of a second the clock is incremented. Interrupts also occur every 50th of a second so that the screen can be updated and the keyboard read, and every 100th of a second so sounds can be dealt with and enevelopes processed.

As you can see, interrupts enable the Amstrad to appear to be carrying out more than one task at once. We don't notice any of these background tasks being performed as they are dealt with quickly and efficiently and in such a manner so as not to disturb the foreground task the Amstrad is running.

One of the machine's most powerful features is its ability to handle interrupts from Basic. This is a major innovation, and is quite a rarity among home micros. For example, it is possible to arrange for a subroutine to be executed every five seconds.

If you haven't yet come across Basic interrupts then I would recommend that you look them up, they can be very useful. The associated Basic commands are AFTER, EVERY and REMAIN.

These are excellent, but how can the machine code programmer use the interrupt facility? It's not all that difficult, and once you get the hang of it you'll find the options available from machine code far more extensive than from Basic.

Handling interrupts can be quite a complicated process, but fortunately the operating system provides an easy-to-use prepackaged form of interrupt known as an event. Events are more flexible than hardware interrupts and there are fewer restrictions.

The time interrupts for the operating system provide three sources of "kicks" for events, and each source has an associated queue. When a particular event is kicked it is placed in the appropriate queue, so the event routine may not be called immediately.

The three sources of kicks are the fast ticker, ticker and frame flyback interrupts.

Fast ticker events are kicked every 1/300th of a second. Ticker events have a timer which is decremented every 1/50th of a second and when the timer reaches zero the event is called. It can then be automatically reset. Frame flyback events are kicked every 1/50th of a second.

Bytes	0/1	Tick chain
Bytes	2/3	Tick count
Bytes	4/5	Recharge count
Bytes	6/7	Chain
Byte	8	Count
Byte	9	Class
Bytes	10/11	Routine address
Byte	12	ROM select

Figure I: Workspace alteration

There are three classes of events. Express asynchronous events are called immediately during interrupt processing, but these are not normally used.

Normal asynchronous events are the most flexible type. When kicked they are placed in an interrupt queue to be processed when the operating system has finished its own interrupts. There are few restrictions, and the routine may take as long to run as is needed.

Synchronous events, when kicked, are placed on a separate queue according to a priority which they are given. The foreground program must poll the queue to see if there are any events outstanding, and process them if there are.

We're now going to look at how normal asynchronous events are processed and see what they can be used for. A couple of simple routines will be placed on the ticker list and we'll set them off at regular intervals.

The operating system requires a small block of memory as workspace for each event routine. We are going to use ticker events, and the block needed for each event is 13 bytes long.

The block is in two parts, the first six bytes being the ticker block and the last seven is the event block. (See Figure I.) The tick chain holds the address of the next tick block, if there is one. The tick count is the timer for the event and is decremented every 50th of a second. When it reaches zero the event routine is called.

The recharge count is the value the timer is reset to after the event has been kicked.

Chain is used to store the event's position in the queue and the count is the number of kicks received. The class I'll come back to. The routine address and the ROM select byte point to the entry address of the routine, which can be in ROM or RAM.

In order for our interrupt routine to be called we need to initialise a ticker block. &BCEF is called with the address of the event block in the HL register pair, the event address in the DE pair, the ROM select address in C and the class in B. We can ignore the ROM select address as the routine isn't in a ROM.

The class is bit significant – see Figure II. We are going to place the routines in the central 32k of RAM. This is classed as a near address, so bit 0 is set. The event is asynchronous, so bits 1-4 are meaningless. Most events will be normal and don't need urgent attention, so bit 6 is zero. It's an asynchronous event so bit 7 is set.

Bit 0	Near/far address
Bits 1-4	Synchronous event priority
Bit 5	Must be zero
Bit 6	Express event
Bit 7	Asynchronous event

Figure II: Class byte

The first four lines of Program I shows the code to initialise the event block. The next four lines show how the tick block may be added to the ticker list.

HL holds the tick block address, DE the initial timer value, BC the recharge value, the routine calls &BCE9 and returns. Now every 50th of a second the timer is decremented and the routine called when it is zero. As we set it to 50 initially this will take exactly one second.

The event routine itself is very simple. It just outputs CHR$(7) by loading the A register with 7 and jumping to &BB5A, the same as CALL1 and RETURN. So every second the micro will beep no matter what it is doing unless interrupts have been disabled for some reason.

Program II shows a slightly more complex event routine. As you can see the initialisation is identical, and, apart from the different timer values, so is the routine to add it to the ticker list.

The event routine reads the keyboard and freezes the program – Basic or machine code – if the Tab key is pressed. It waits until the Caps Lock key is pressed before continuing the program.

This is quite a useful little routine. It's quite handy to be able to freeze your favourite arcade game when the phone rings, for example, or just to have a breather.

If you want to use this do make sure it's tucked out of the way of your game. If you hide it in the function key buffer at about &B460 it will even freeze a fair proportion of commercial

```
RAW Assembler V.3

Pass... 2           ORG &8000

                    Initialise event
8000:21 28 80       LD HL,eventblock
8003:06 81          LD B,%10000001
8005:11 1D 80       LD DE,event
8008:C3 EF BC       JP &BCEF

                    Add to tick list
800B:21 22 80       LD HL,tickblock
800E:11 32 00       LD DE,50
8011:01 32 00       LD BC,50
8014:C3 E9 BC       JP &BCE9

                    Remove from list
8017:21 22 80       LD HL,tickblock
801A:C3 EC BC       JP &BCEC

                    Event routine
801D:               .event
801D:3E 07          LD A,7
801F:C3 5A BB       JP &BB5A

                    Workspace
8022:               .tickblock
8022:               DEFS 6
8028:               .eventblock
8028:               DEFS 7
802F:               END
```

Program I

```
RAW Assembler V.3

Pass... 2           ORG &8030

                    Initialise event
8030:21 61 80       LD HL,eventblock
8033:06 81          LD B,%10000001
8035:11 4D 80       LD DE,event
8038:C3 EF BC       JP &BCEF

                    Add to tick list
803B:21 5B 80       LD HL,tickblock
803E:11 05 00       LD DE,5
8041:01 05 00       LD BC,5
8044:C3 E9 BC       JP &BCE9

                    Remove from list
8047:21 5B 80       LD HL,tickblock
804A:C3 EC BC       JP &BCEC

                    Event routine
804D:               .event
804D:3E 44          LD A,68
804F:CD 1E BB       CALL &BB1E
8052:C8             RET Z
8053:               .loop
8053:3E 46          LD A,70
8055:CD 1E BB       CALL &BB1E
8058:28 F9          JR Z,loop
805A:C9             RET

                    Workspace
805B:               .tickblock
805B:               DEFS 6
8061:               .eventblock
8061:               DEFS 7

8068:               END
```

Program II

```
10 REM PROGRAM III
20 FOR i=0 TO 90
30 READ byte$
40 POKE &8000+i,VAL("&"+byte$)
60 NEXT
70 DATA 21,28,80,06,81,11,1D,80,C3
80 DATA EF,BC,21,22,80,11,32,00,01
90 DATA 32,00,C3,E9,BC,21,22,80,C3
100 DATA EC,BC,3E,07,C3,5A,BB,00,00
110 DATA 00,00,00,00,00,00,00,00,00
120 DATA 00,00,00
130 REM
140 DATA 21,61,80,06,81,11,4D,80,C3
150 DATA EF,BC,21,5B,80,11,05,00,01
160 DATA 05,00,C3,E9,BC,21,5B,80,C3
170 DATA EC,BC,3E,44,CD,1E,BB,C8,3E
180 DATA 46,CD,1E,BB,28,F9,C9
```

Program III

```
10 REM PROGRAM IV
20 MODE 1:INK 0,0:BORDER 0
30 PEN 1:PAPER 3:LOCATE 9,3
40 PRINT " TICKER AND EVENT BLOCK "
50 PAPER 0
60 i=&8022:j=i+2
70 LOCATE 1,7
80 PRINT TAB(9)"Tick chain";:GOSUB 19
0
90 PRINT TAB(9)"Tick count";:GOSUB 19
0
100 PRINT TAB(9)"Recharge Count";:GOS
UB 190
110 PRINT TAB(9)"Chain";:GOSUB 190
120 PRINT TAB(9)"Count";:GOSUB 180
130 PRINT TAB(9)"Class";:GOSUB 180
140 PRINT TAB(9)"Routine address";:GO
SUB 190
150 PRINT TAB(9)"ROM";:GOSUB 180
160 PEN 3
170 LOCATE 28,9:PRINT "&";HEX$(PEEK(j
)+256*PEEK(j+1),4):GOTO 170
180 GOSUB 200:PRINT "&";HEX$(PEEK(i),
2):PRINT:i=i+1:PEN 1:RETURN
190 GOSUB 200:PRINT "&";HEX$(PEEK(i)+
256*PEEK(i+1),4):PRINT:i=i+2:PEN 1:RE
TURN
200 PEN 2:WHILE POS(#0)<28:PRINT ".";
:WEND:PEN 3:RETURN
```

Program IV

software. It's then possible to step through the program little by little and see how it works.

If you haven't got an assembler to hand then Program III will poke the relevant data to &8000.

So far I've not covered how to remove an event from the ticker list. It's quite simple. Just load the HL register pair with the address of the tick block and CALL &BCEC.

After assembling the routines CALL &8000 and CALL &8030 to initialise the events. CALL &800B and CALL &803B to enable them, CALL &8017 and CALL &8047 to disable them.

Program IV can be used to study the tick blocks and event blocks, *i* in line 60 should point to the start of the block. It prints out the contents of the event and tick blocks so you can watch the operating system actually running the events.

Once you've got the hang of events, you'll be amazed how often they pop up as an ideal solution to programming problems. If you're not using interrupts, you're not using your Amstrad to the full!

Make sure those stompers don't scramble your eggs!

By ARAMELLO CHAPMAN

IN the Land of Block, to the North of Blob, lived a happy bunch of egg-farmers, who tend the eggs of the Wob-Wob birds.

However, one day, from the Eastern Land of Grog, came the evil Stompers. After many days of being stomped, the farmers gave up and went home to play with their Spectrums – they are a backward lot in Block.

Bod does not have a Spectrum and, for want of a better thing to do, he decides to save the eggs of the Wob-Wob birds.

Thus stands Bod against the mighty Stompers. Can you help him in his hellish quest?

Bod has some weird powers which can be of use on later levels. He can slide the row of blocks he is standing on, either left or right.

There are several hazards which cause hassle to Bod. Every time he moves, the blocks he was just standing on collapse. Also surrounding Bod are a lot of sleeping Stompers. And after two screens have been cleared, a mutant Stomper wakes up and starts stomping.

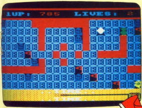
On screen seven and after, the Stomper starts leaving holes as it moves around. At least Bod can earn points by collecting eggs and flags.

ROUTINES

180	Main loop	Makes jumps to movement routine and checks to see if Bod is killed.
330	Time	Decreases time on fuse.
410	Move Bod	Includes rotation of Bod and movement in direction in which Bod is facing.
600	Slide left	Slides left the blocks which Bod is on.
700	Slide right	Slides right the blocks which Bod is on.
800	Lose life	Decreases lives, wobbles screen and checks if any more lives left.
960	Landed on egg	Increases score, checks to see if any more eggs on screen.
1070	Move enemy	If level < 7 moves and replaces scenery. If > 6 moves enemy and erases scenery.
1240	Set up screen	Prints screen with various characters.
1530	Puts egg on screen	Self-expanatory.
1610	U.D.G.	User defined graphics.
2010	Variables	Sets variables which change from game to game.
2180	Instructions	Prints instructions.
2650	Machine code	Scrolls row left and right, held above Basic memory.
2810	High score	Prints high score table. Enter name if high enough.

VARIABLES

x	X coordinates of Bod.
y	Y coordinates of Bod.
z	Y coordinates of Bod in set array.
bootx	X coordinates of Enemy.
booty	Y coordinates of Enemy.
bootz	Y coordinates of Enemy in set array.
mh$	Head of monsters (sleeping Stompers).
bh$	Head of Bod.
eh$	Head of enemy Stompers (moving Stompers).
mb$	Body of monster.
bb$	Body of Bod.
eb$	Body of enemy Stompers.
na$	Name of all high scorers.
hs	High scores.
sc	Scores.
level	Which screen Bod is on.
set (9,20)	Array of screen contents.
eggs	Number of eggs collected.
ti	Time left on TNT.
bod	Direction in which Bod is facing.
lives	Number of lives left.
p1	Which Stomper is present.
monster	Which monster (sleeping Stomper) is present.



KEYS:

- Z Rotates Bod left.
- X Rotates Bod right.
- / Moves Bod in direction he is facing.
- . Slides blocks left.
- \ Slides blocks right.

```
10 REM Ah Bod!!
20 REM By Aramello Chapman
30 REM(c)Computing With The Amstrad
40 REM
50 REM
60 DIM set(9,20)
70 DIM hs(8):DIM na$(8)
80 RESTORE 100
90 FOR f=1 TO 8:READ hs(f),na$(f):NEX
T f
100 DATA 5500,Mr,5000,Del Monte,4000,
Says,3500,Yes!!!,3000,Stomper,2500,Ah
 Bod!!,2000,Blob's,1500,Revenge
110 GOSUB 1610:REM U.D.G's
120 GOSUB 2650:REM Machine Code
130 GOSUB 2180:REM Instructions
140 GOSUB 2010:REM Variables
150 GOSUB 1240:REM Set Up Screen
160 LET eggs=0
170 LOCATE bootx,booty:PEN 1:PRINT MI
D$(eh$,p1,1):LOCATE bootx,booty+1:PRI
NT MID$(eb$,p1,1):LOCATE 18,2:PEN 3:P
RINT lives
180 REM***********Main Loop**********
*
210 LOCATE 6,2:PEN 3:PRINT sc
220 IF INKEY(71)=0 OR INKEY(63)=0 OR
INKEY(30)=0 THEN GOSUB 410
230 IF INKEY(31)=0 THEN GOSUB 600
240 IF INKEY(22)=0 THEN GOSUB 700
250 IF set(z,x)=2 THEN LET sc=sc+10:L
ET set(z,x)=5
260 IF set(z,x)=3 THEN GOTO 960
270 IF level>2 THEN GOSUB 1070
280 IF booty=y AND bootx=x THEN GOTO
800
290 IF set(z,x)=0 OR set(z,x)=4 THEN
GOTO 800
300 GOSUB 330:REM decrease time
310 IF ti<=2 THEN FOR f=1 TO 5:SOUND
1,0,10,7,0,0,1:INK 0,3:BORDER 3:SOUND
 1,0,25,5,0,0,10:INK 0,0:BORDER 0:NEX
T f:FOR f=1 TO 100:NEXT f:GOTO 800
315 LET fr=FRE("")
320 GOTO 180
330 REM***********TIME***************
360 LET ti=ti-(level/50)
380 LOCATE 19-ti,23:PAPER 3:PEN 1:PRI
NT CHR$(234);:PEN 0:PRINT CHR$(235):P
APER 0
400 RETURN
410 REM************Move Bod**********
*
440 IF INKEY(30)=0 THEN GOTO 520
450 IF INKEY(63)=0 THEN LET bod=bod-1
460 IF INKEY(71)=0 THEN LET bod=bod+1
470 IF bod=0 THEN LET bod=4
480 IF bod=5 THEN LET bod=1
490 LOCATE x,y:PEN 12:PRINT MID$(bh$,
bod,1)
500 LOCATE x,y+1:PRINT MID$(bb$,bod,1
)
510 RETURN
520 LOCATE x,y:PAPER 3:PRINT" ":LOCAT
E x,y+1:PRINT" ":PAPER 0
530 IF bod=1 AND x<20 THEN LET set(z,
x)=0:LET x=x+1:SC=SC+1
540 IF bod=2 AND y<20 THEN LET set(z,
x)=0:LET y=y+2:LET z=z+1:SC=SC+1
550 IF bod=3 AND x>1 THEN LET set(z,x
)=0:LET x=x-1:SC=SC+1
560 IF bod=4 AND y>4 THEN LET set(z,x
)=0:y=y-2:LET z=z-1:SC=SC+1
570 LOCATE x,y:PEN 12:PRINT MID$(bh$,
bod,1):LOCATE x,y+1:PRINT MID$(bb$,bo
d,1)
580 SOUND 1,2900,20,5,0,0,1
590 RETURN
600 REM**********Rotate Left*********
*
630 FOR f=1 TO 4:CALL high,y-1:CALL h
igh,y:NEXT f
640 LET p3=set(z,1)
650 FOR f=1 TO 19:LET set(z,f)=set(z,
f+1):NEXT f:LET set(z,20)=p3
660 LET x=x-1:IF x=0 THEN LET x=20
670 IF booty=y THEN LET bootx=bootx-1
680 IF bootx=0 THEN LET bootx=20
690 RETURN
700 REM**********Rotate Right********
*
730 FOR f=1 TO 4:CALL high+50,y-1:CAL
L high+50,y:NEXT f
740 LET p3=set(z,20)
750 FOR f=20 TO 2 STEP-1:LET set(z,f)
=set(z,f-1):NEXT f:LET set(z,1)=p3
760 LET x=x+1:IF x=21 THEN LET x=1
770 IF booty=y THEN LET bootx=bootx+1
780 IF bootx=21 THEN LET bootx=1
790 RETURN
800 REM***********Lose Life**********
*
830 LOCATE bootx,booty:PAPER 3:PRINT"
 ":LOCATE bootx,booty+1:PRINT" ":PAPE
R 0:LOCATE x,y:PEN 12:PRINT MID$(bh$,
bod,1):LOCATE x,y+1:PRINT MID$(bb$,bo
d,1)
834 IF bootz<0 THEN LET bootz=1
835 IF set(bootz,bootx)<>3 THEN LET s
et(bootz,bootx)=0 ELSE IF set(bootz,b
ootx)=3 THEN PEN 4:LOCATE bootx,booty
:PRINT CHR$(222):LOCATE bootx,booty+1
:PRINT CHR$(223)
836 LOCATE 1,23:PAPER 3:PEN 1:PRINT S
TRING$(18,CHR$(234)):PAPER 0:PEN 1
840 LET lives=lives-1:LET ti=18
850 IF lives=0 THEN GOTO 910
860 LET set(z,x)=1
870 FOR f=1 TO 100:OUT &BC00,8:OUT &B
D00,1:SOUND 130,0,50,15,0,0,1:NEXT f:
OUT &BC00,8:OUT &BD00,0:LOCATE 18,23:
PAPER 3:PEN 1:PRINT CHR$(234):PRINT C
HR$(22)+CHR$(1):LOCATE 2,23:PAPER 3:P
EN 5:PRINT STRING$(17,CHR$(154)):PAPE
```

```
R 0:PRINT CHR$(22)+CHR$(0)
880 IF booty<>y AND bootx<>x THEN GOT
O 170
890 IF x=20 AND y=20 THEN LET booty=4
:LET bootx=1:GOTO 170
900 LET booty=20:LET bootx=20:GOTO 17
0
910 FOR f=1 TO 2:OUT &BC00,8:OUT &BD0
0,1:RESTORE 940:FOR g=1 TO 11:READ d,
n:SOUND 1,n,d,7:SOUND 4,n+2,d,7:SOUND
 5,0,3,0:NEXT g,f
920 OUT &BC00,8:OUT &BD00,0
930 LOCATE 1,1:SOUND 130,0,50,3,0,0,1
::PRINT STRING$(26,11)
940 DATA 50,1016,37,1016,12,1016,50,1
016,25,850,25,899,25,899,25,1016,25,1
016,25,1136,100,1016
950 GOTO 2820
960 REM*********Landed on Egg********
*
990 LET eggs=eggs+1
1000 SOUND 2,200,25,7,0,4:SOUND 2,200
,25,5,0,1:LET sc=sc+100:LET set(z,x)=
1
1010 IF eggs=level+3 THEN GOTO 1040
1020 GOSUB 1530
1030 GOTO 100
1040 LET sc=sc+INT(ti)*10:LET level=l
evel+1
1050 LET ti=18
1060 GOTO 150
1070 REM***********Move Enemy*********
**
1091 IF bootz<0 THEN LET bootz=1
1100 IF level>6 AND set(bootz,bootx)=
1 THEN LOCATE bootx,booty:PAPER 3:PRI
NT" ":LOCATE bootx,booty+1:PRINT" ":L
ET set(bootz,bootx)=0:PAPER 0:GOTO 11
60
1110 IF set(bootz,bootx)=0 THEN LOCAT
E bootx,booty:PAPER 3:PRINT" ":LOCATE
 bootx,booty+1:PRINT" ":PAPER 0
1120 IF set(bootz,bootx)=1 THEN LOCAT
E bootx,booty:PEN 10:PRINT CHR$(200):
LOCATE bootx,booty+1:PRINT CHR$(201)
1130 IF set(bootz,bootx)=2 THEN PEN 3
:LOCATE bootx,booty:PRINT CHR$(212):L
OCATE bootx,booty+1:PRINT CHR$(213)
1140 IF set(bootz,bootx)=3 THEN PEN 4
:LOCATE bootx,booty:PRINT CHR$(222):L
OCATE bootx,booty+1:PRINT CHR$(223)
1150 IF set(bootz,bootx)=4 THEN PEN m
onster+5:LOCATE bootx,booty:PRINT MID
$(mh$,monster,1):LOCATE bootx,booty+1
:PRINT MID$(mb$,monster,1)
1160 LET MOV=INT(RND*4)+1
1170 IF MOV=1 AND bootx<20 THEN LET b
ootx=bootx+1
```

```
1180 IF MOV=2 AND booty<20 THEN LET b
ooty=booty+2:LET bootz=bootz+1
1190 IF MOV=3 AND bootx>1 THEN LET bo
otx=bootx-1
1200 IF MOV=4 AND booty>4 THEN LET bo
oty=booty-2:LET bootz=bootz-1
1210 LOCATE bootx,booty:PEN 1:PRINT M
ID$(eh$,p1,1):LOCATE bootx,booty+1:PR
INT MID$(eb$,p1,1)
1220 SOUND 1,266,5,7:SOUND 1,399,5,7
1230 RETURN
1240 REM*********Set Up Screen*******
**
1270 LET y=10:LET x=10:LET z=4:LET bo
otx=20:LET booty=20:LET bootz=9
1280 PAPER 0:BORDER 0:CLS
1290 LET monster=INT(RND*4)+1:LET p1=
INT(RND*4)+1
1300 FOR f=1 TO 9:FOR g=1 TO 20:LET s
et(f,g)=1:NEXT g,f
1310 FOR f=1 TO 6+level
1320 LET x1=INT(RND*20)+1:LET y1=INT(
RND*9)+1
1330 IF x1=10 THEN GOTO 1320
1340 LET set(y1,x1)=4
1350 NEXT f
1360 FOR f=1 TO 3+level
1370 LET x1=INT(RND*20)+1:LET y1=INT(
RND*9)+1
1380 IF set(y1,x1)=1 THEN LET set(y1,
x1)=2
1390 NEXT f
1400 LET count=1
1410 FOR f=4 TO 20 STEP 2:FOR g=1 TO
20
1420 IF set(count,g)=1 THEN LOCATE g,
f:PEN 10:PRINT CHR$(200):LOCATE g,f+1
:PRINT CHR$(201)
1430 IF set(count,g)=2 THEN LOCATE g,
f:PEN 3:PRINT CHR$(212):LOCATE g,f+1:
PRINT CHR$(213)
1440 IF set(count,g)=4 THEN PEN monst
er+5:LOCATE g,f:PRINT MID$(mh$,monste
r,1):LOCATE g,f+1:PRINT MID$(mb$,mons
ter,1)
1450 NEXT g:LET count=count+1:NEXT f
1460 LOCATE x,y:PEN 12:PRINT MID$(bh$
,bod,1):LOCATE x,y+1:PRINT MID$(bb$,b
od,1)
1470 LOCATE bootx,booty:PEN 1:PRINT M
ID$(eh$,p1,1):LOCATE bootx,booty+1:PR
INT MID$(eb$,p1,1)
1480 LOCATE 1,2:PEN 2:PRINT" 1UP:";TA
B(12);"LIVES:"
1490 LOCATE 1,1:PEN 3:PRINT STRING$(2
0,CHR$(131)):LOCATE 1,3:PRINT STRING$
(20,CHR$(140))
1500 LOCATE 1,22:PAPER 3:PEN 1:PRINT
```

```
STRING$(59,CHR$(234)):LOCATE 20,24:PR
INT CHR$(234):PAPER 0
1510 LOCATE 1,23:PRINT CHR$(22)+CHR$(
1):LOCATE 2,23:PEN 5:PRINT STRING$(17
,CHR$(154));:LOCATE 19,23:PRINT CHR$(
230);CHR$(232):LOCATE 19,24:PRINT CHR
$(231);CHR$(233):LOCATE 7,24:PEN 4:PR
INT"..TIME..":LOCATE 7,22:PRINT"..TIM
E.."
1520 LOCATE 1,23:PRINT CHR$(22)+CHR$(
0)
1530 REM*******Put Egg On Screen*****
**
1560 LET y1=INT(RND*9)+1:LET x1=INT(R
ND*20)+1:IF y1=20 AND x1=20 THEN GOTO
 1560
1570 IF set(y1,x1)=1 THEN LET set(y1,
x1)=3 ELSE GOTO 1560
1580 LET count=1:FOR f=4 TO 20 STEP 2
:FOR g=1 TO 20
1590 IF set(count,g)=3 THEN PEN 4:LOC
ATE g,f:PRINT CHR$(222):LOCATE g,f+1:
PRINT CHR$(223) ELSE NEXT g:LET count
=count+1:NEXT f
1600 RETURN
1610 REM***********U.D.G************
**
1640 SYMBOL AFTER 199
1650 SYMBOL 200,126,255,129,185,165,1
65,165,185
1660 SYMBOL 201,185,165,165,165,185,1
29,255,126
1670 SYMBOL 202,0,240,240,80,112,80,1
12,80
1680 SYMBOL 203,112,80,112,80,254,255
,255,126
1690 SYMBOL 204,24,24,126,44,34,36,24
,60
1700 SYMBOL 205,110,111,103,119,62,24
,24,28
1710 SYMBOL 206,28,28,127,62,62,28,8,
28
1720 SYMBOL 207,62,127,127,127,62,28,
20,54
1730 SYMBOL 208,28,28,127,54,42,34,28
,8
1740 SYMBOL 209,28,62,93,93,62,8,20,5
4
1750 SYMBOL 210,24,24,126,52,68,36,24
,60
1760 SYMBOL 211,118,246,230,238,124,2
4,24,48
1770 SYMBOL 212,3,15,63,115,247,115,5
5,15
1780 SYMBOL 213,3,1,1,1,1,1,3,7
1790 SYMBOL 214,24,60,60,126,94,74,82
,122
1800 SYMBOL 215,126,126,153,195,255,2
55,255,155
1810 SYMBOL 216,0,195,66,66,90,60,90,
153
1820 SYMBOL 217,219,126,60,24,24,36,3
6,231
1830 SYMBOL 218,129,126,153,24,60,90,
189,255
1840 SYMBOL 219,219,102,60,24,102,36,
36,231
1850 SYMBOL 220,24,60,126,219,102,60,
24,255
1860 SYMBOL 221,189,189,153,153,60,12
6,255,255
1870 SYMBOL 222,24,60,60,126,126,255,
255,255
1880 SYMBOL 223,126,126,60,60,24,0,0,
0
1890 SYMBOL 224,24,24,126,255,153,255
,219,102
1900 SYMBOL 225,189,195,126,24,24,24,
24,126
1910 SYMBOL 226,153,36,46,69,210,146,
74,131
1920 SYMBOL 227,149,210,97,45,42,18,1
56,73
1930 SYMBOL 228,28,62,107,255,247,91,
99,254
1940 SYMBOL 229,195,255,255,126,60,28
,12,6
1950 SYMBOL 230,0,31,32,223,192,64,92
,72
1960 SYMBOL 231,74,75,66,66,32,31,0,0
1970 SYMBOL 232,0,248,4,250,2,58,18,1
0
1980 SYMBOL 233,18,66,194,66,4,248,0,
0
1985 SYMBOL 234,146,255,73,255,146,25
5,73,255
1990 SYMBOL 235,132,74,40,151,95,40,6
8,146
1995 SYMBOL 236,198,165,198,165,6,40,
40,16
2000 RETURN
2010 REM***********Variables*********
**
2040 ENT 1,5,1,1,10,-1,1,10,1,1,10,1,
5,1,1,5:ENV 3,3,2,2,3,-2,2:ENT 4,5,-1
0,10,1,0,5
2050 LET mh$=CHR$(214)+CHR$(216)+CHR$
(218)+CHR$(220)
2060 LET mb$=CHR$(215)+CHR$(217)+CHR$
(219)+CHR$(221)
2070 LET bh$=CHR$(204)+CHR$(208)+CHR$
(210)+CHR$(206)
2080 LET bb$=CHR$(205)+CHR$(209)+CHR$
(211)+CHR$(207)
2090 LET bod=1
2100 LET eh$=CHR$(202)+CHR$(224)+CHR$
(226)+CHR$(228)
2110 LET eb$=CHR$(203)+CHR$(225)+CHR$
(227)+CHR$(229)
2120 LET lives=3
2130 LET sc=0
2140 LET level=1
2150 MODE 0
2160 LET ti=10
2170 RETURN
2180 REM********Instructions*********
**
2210 BORDER 0:GOTO 2290
2220 MODE 1:PEN 3:PRINT STRING$(40,CH
R$(200)):LOCATE 1,2:PRINT STRING$(40,
CHR$(201))
2230 LOCATE 1,3:PRINT CHR$(200);TAB (
16);:PEN 2:PRINT"*AH BOD!!*";TAB(40):
PEN 3:PRINT CHR$(200)
2240 LOCATE 1,4:PRINT CHR$(201);STRIN
G$(38,CHR$(200));CHR$(201):LOCATE 2,5
:PRINT STRING$(38,CHR$(201))
2245 IF f1=1 THEN RETURN
2250 LOCATE 1,20:PRINT STRING$(40,CHR
$(200)):LOCATE 1,21:PRINT STRING$(40,
CHR$(201))
2260 LOCATE 1,22:PRINT CHR$(200);TAB
(8);:PEN 1:PRINT"Press <SPACE> for ne
xt page.";TAB(40):PEN 3:PRINT CHR$(20
0)
2270 LOCATE 1,23:PRINT CHR$(201);STRI
NG$(38,CHR$(200));CHR$(201):LOCATE 2,
24:PRINT STRING$(38,CHR$(201))
2280 RETURN
2290 INK 0,0:GOSUB 2220
2300 LOCATE 1,7:PEN 2
2310 PRINT"  In the land of Block to
the north of  Blob the evil Stomper's
 are causing     havoc with the egg f
armers who tend the eggs of the Wob-W
ob Birds."
2320 PRINT"  After many days of being
 Stomped the  farmers were a bit unha
ppy and decided  to go home and play
with their Spectrums(They are a backw
ards race of farmers)."
2330 PRINT"  However Bod does not hav
e a Spectrum  and thus for want of a
better thing to  do decides to save t
he Wob-Wob Birds."
2340 PRINT"It is your task to help hi
m with his    Hellish Quest."
2350 IF INKEY$<>"" THEN GOTO 2350
2360 WHILE INKEY$<>" ":WEND
2370 LOCATE 1,1:PRINT STRING$(26,11)
2380 GOSUB 2220
2390 LOCATE 1,7:PEN 2
2400 PRINT"  Bods task is to collect
all the eggs  on the screen.The eggs
```

```
appear one at a  time and you have to
 get all of them    before the fuse o
n the TNT burns down."
2410 PRINT"  On the screen are flags
which gain    extra points if collect
ed.There are alsoa few hazards in the
 land of Block.      Resting Stomper
s are all around Bod   and after he h
as cleared two screens a  mutant Stom
per starts Stomping."
2420 PRINT"  Every time Bod moves,the
 block he was just standing on crumbl
es away leaving  a gaping hole which
must be avoided."
2430 IF INKEY$<>"" THEN GOTO 2430
2440 WHILE INKEY$<>" ":WEND
2450 LOCATE 1,1:PRINT STRING$(26,11)
2460 GOSUB 2220
2470 LOCATE 1,7:PEN 2
2480 PRINT"  Bod however is not your
every day,run of the mill egg farmer.
He has strange   powers only to be fo
und in the land of  Block.He can rota
te the row of blocks   on which he is
 standing either left or  right."
2490 PRINT"  Bod can not walk off the
 land of block.However he can rotate
himself off one  side and appear on t
he other."
2500 PRINT"  Summary-collect eggs & f
lags.Avoid allmonters be they moving
or standing still.Don't hang around t
o long and don't tryto cross the hole
s in Block land."
2510 IF INKEY$<>"" THEN GOTO 2510
2520 WHILE INKEY$<>" ":WEND
2530 LOCATE 1,1:PRINT STRING$(26,11)
2540 GOSUB 2220
2550 LOCATE 16,7:PEN 2:PRINT"THE KEYS
":LOCATE 16,8:PEN 1:PRINT STRING$(8,C
HR$(131))
2560 PEN 2:PRINT TAB(17):PRINT"Rotate
":PRINT"       Left-'Z'";TAB(26);"Righ
t-'X'"
2570 PEN 3:PRINT TAB(16);"Move-'/'"
2580 PEN 1:PRINT:PRINT TAB(8);"  Rota
te Blocks Left-'.'"
2590 PEN 2:PRINT:PRINT TAB(8);"  Rota
te Blocks Right-'\'"
2600 LOCATE 15,19:PEN 1:PRINT"*GOOD-L
UCK*
2610 IF INKEY$<>"" THEN GOTO 2610
2620 WHILE INKEY$<>" ":WEND
2630 LOCATE 1,1:PRINT STRING$(26,11)
2640 GOTO 2820
2650 REM########Machine Code#########
#
2680 LET hig=HIMEM:MEMORY hig:LET add
```

```
ress=hig+1500:LET high=hig+1500:RESTO
RE 2740
2690 READ n
2700 LET sc=sc+n:IF n=999 THEN READ n
:IF n<>sc THEN PRINT CHR$(7):PRINT ST
RING$(6,CHR$(7)):CLS:LOCATE 11,10:PEN
 3:PRINT"ERROR IN MACHINE CODE":END E
LSE IF n=16121 THEN LET sc=0:RETURN
2710 POKE address,n
2720 LET address=address+1
2730 GOTO 2690
2740 REM############LEFT SCROLL#######
2750 DATA &dd,&6e,&00,&26,&00,&cd,&1a
,&bc,&e5,&7e,&f5,&54,&5d,&eb,&23,&cb,
&f4,&cb,&fc,&0e,&4f,&7e,&12,&23,&cb,&
f4,&cb,&fc,&eb,&23,&cb,&f4,&cb,&fc,&e
b,&0d
2760 DATA &20,&ef,&f1,&12,&e1,&7c,&c6
,&00,&67,&e6,&38,&20,&d7,&c9
2770 REM###########RIGHT SCROLL#######
2780 DATA &dd,&6e,&00,&26,&00,&cd,&1a
,&bc,&01,&4f,&00,&09,&cb,&4f,&cb,&fc,
&e5,&7e,&f5,&54,&5d,&2b,&cb,&f4,&cb,&
fc,&0e,&4f,&7e,&12,&2b,&cb,&f4,&cb,&f
c,&eb,&2b,&cb,&f4
2790 DATA &cb,&fc,&eb,&0d,&20,&ef,&f1
,&12,&e1,&7c,&c6,&00,&67,&e6,&38,&20,
&d8,&c9
2800 DATA 999,16121
2810 REM##########High Score#########
2820 MODE 1
2830 LET f1=1:GOSUB 2220:f1=0
2840 LOCATE 16,3:PEN 2:PRINT"HIGH SCO
RE":PEN 3
2850 FOR f=6 TO 14 STEP 2::LOCATE 4,f
:PRINT CHR$(200):LOCATE 4,f+1:PRINT C
HR$(201):LOCATE 37,f:PRINT CHR$(200):
LOCATE 37,f+1:PRINT CHR$(201):NEXT f
2860 LOCATE 5,16:PRINT STRING$(32,CHR
$(201))
2870 FOR f=1 TO 8
2880 IF sc>hs(f) THEN GOSUB 2980:f=10
2890 NEXT f
2900 FOR f=1 TO 8:PEN 1:LOCATE 8,f+6:
PRINT na$(f):LOCATE 18,f+6:PEN 3:PRIN
T"..........";hs(f):NEXT f
2910 LOCATE 1,17:PRINT STRING$(220,"
")
2920 IF INKEY$<>"" THEN GOTO 2920
2930 PEN 2:LOCATE 1,20:PRINT STRING$(
40,CHR$(154)):LOCATE 1,22:PRINT STRIN
G$(40,CHR$(154)):PEN 1:
2940 LOCATE 1,23:PRINT STRING$(40," "
)
2950 LOCATE 11,21:PEN 3:PRINT"PRESS";
:PEN 1:PRINT"< SPACE >";:PEN 3:PRINT"
TO PLAY.":PEN 1
2960 IF INKEY(47)<>0 THEN GOTO 2960
```

```
2970 LOCATE 1,1:PRINT STRING$(26,11):
GOTO 140
2980 LET a$="ABCDEFGHIJKLMNOPQRSTUVWX
YZ .0%&()!()?*+"+CHR$(236)
2990 LET c=19:LOCATE 1,20:PEN 1:PRINT
 a$
3000 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))
3010 LOCATE 12,23:PEN 3:PRINT"PRESS";
:PEN 1:PRINT"< 'X' >";:PEN 3:PRINT"TO
 EXIT.":PEN 1
3020 LET x$=""
3030 FOR z=1 TO 10
3040 LOCATE c,21:PEN 2:PRINT" "
3050 IF INKEY(1)=0 AND c<40 THEN LET
c=c+1
3060 IF INKEY(8)=0 AND c>1 THEN LET c
=c-1
3070 IF INKEY(9)=0 AND c=40 THEN LOCA
TE 7,f+6:PRINT"           ":LET z=100
:GOTO 3120
3080 IF INKEY(63)=0 THEN LET z=11:GOT
O 3120
3090 IF INKEY(9)<>0 THEN LOCATE c,21:
PRINT"*":FOR a=1 TO 50:NEXT a:GOTO 30
40
3100 LET x$=x$+MID$(a$,c,1):LOCATE 7+
z,f+6:PEN 1:PRINT MID$(a$,c,1)
3110 FOR a=1 TO 200:NEXT a
3120 NEXT z
3130 IF z=101 THEN GOTO 3020
3140 LET hs(8)=sc:LET na$(8)=x$
3150 LET f=0
3160 FOR z=1 TO 7
3170 IF hs(z)<hs(z+1) THEN LET t=hs(z
+1):LET hs(z+1)=hs(z):LET hs(z)=t:LET
 a$=na$(z+1):LET na$(z+1)=na$(z):LET
na$(z)=a$:LET f=1
3180 NEXT z
3190 IF f=1 THEN GOTO 3150
3200 LET fr=FRE("")
3210 RETURN
```

Let machine code make your graphics really flow

ROLAND WADDILOVE introduces the basic techniques of moving coloured characters around your Amstrad's screen

THE graphics of some of the latest software for the Amstrad are absolutely amazing, being packed full of incredibly fast, super smooth, multi-colour, sprite-like characters.

Roland in Time (no relation) is an excellent example. Have you ever wondered how it is done?

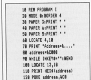
While it's impossible to cover everything in the course of a magazine series, over the next few months I shall be showing you some of the basic techniques involved in moving multi-coloured characters of any size smoothly round the screen.

Although Amstrad Basic is pretty fast, really the only way to achieve such animation is through the use of machine code, as it runs many times faster than Basic.

So to make the most of these articles you will need a fair knowledge of Z80 machine code. Even if you haven't, you should still be able to follow the first section, which looks at how the screen memory is organised, and you will have till next month to swot up on the subject. Mike Bibby's machine code series, which started in our January issue, should get you off to a flying start.

Mode 0 is the (relatively) low resolution, multi-colour mode that most arcade games are written in, so this is the one we shall concentrate on. But if you wish to experiment, the techniques involved can quite easily be transferred to another mode with minor adaptations.

The secret of high speed multi-colour graphics is to access the screen memory directly and to use the operating system as little as possible. If you disassemble almost any commercial program you will find few firmware calls.



```
10 REM PROGRAM I
20 MODE 0:BORDER 4
30 PAPER 3:PRINT " "
40 PAPER 6:PRINT " "
50 PAPER 5:PRINT " "
60 LOCATE 4,10
70 PRINT "Address=&...."
80 address=&C000
90 WHILE INKEY$="":WEND
100 LOCATE 13,10
110 PRINT HEX$(address)
120 POKE address,&C0
130 address=address+1
140 GOTO 90
```

Program I

It's not that there is anything wrong with using the firmware – quite the contrary, it's excellent – but it wasn't designed specifically to run arcade games in Mode 0, being intended to perform a much wider range of tasks.

For example, in the OS ROM there's a superb routine which will print any character you care to define, in any colour and in any mode at any pixel. It gives me a headache just thinking about the calculations it must perform.

To detail just a few of the problems, the bit pattern must be fetched, the foreground and background colour found and the bytes required to produce the pattern calculated, which depends on the mode, and whether you are printing at the text or the graphics cursor using TAG. The correct addresses in the screen memory must then be found and the data poked in.

As you can see, quite a lot of work is involved.

However if we know in advance what the bit pattern, colour and mode will be, the data can be worked out before the program is run and a fantastic amount of time can be saved. Then all that is necessary is to poke the data into the correct location, using a greatly simplified routine.

It's not as versatile as the firmware routines, but is far faster.

The first thing to do is to try to find out how the Amstrad organises the screen memory. You will need Program I for this.

The memory map in the Amstrad manual doesn't tell us much except that the screen RAM is between &C000 and &FFFF – the top 16k, underneath the Basic ROM.

Program I should give you a clearer idea how things work. It prints three blocks of colour at the top left of the screen and then waits for a key to be pressed. The variable *address* is initially set to &C000 and whenever a key is pressed its value is printed and &C0 stored in the screen memory *address* is then incremented by 1.

Run the program, holding down a key. You will see that each text character is four bytes wide and that the whole top row of pixels on the top line of the screen is poked first, then the whole of the top row on the second line, the top row of the third line, then the fourth and so on.

The actual byte being poked is displayed on the screen.

Keep a key pressed when it runs off the bottom of the screen and watch it reappear at the top &40

```
10 REM Screen Display
20 REM By R.A.Waddilove
30 REM (c)Computing With The Amstrad
40 REM -----------------------------
50 MODE 0
60 GOSUB 210:REM initialise
70 WHILE UPPER$(INKEY$)<>"E"
80 PLOT 0,398,colour1:PLOT 4,398,colo
ur2
90 LOCATE 14,4:PEN colour1:PRINT CHR$
(233);CHR$(9);CHR$(9);:PEN colour2:PR
INT CHR$(233)
100 PEN 2:LOCATE 13,6:PRINT USING "00
";colour1;:PRINT CHR$(9);USING "00";c
olour2
110 byte=PEEK(&C000):LOCATE 4,5:PRINT
 "&";HEX$(byte,2)
120 PAPER 8:LOCATE 7,16:FOR i=7 TO 0
STEP -1:GOSUB 330:NEXT
130 LOCATE 5,20:i=1:GOSUB 330:i=5:GOS
UB 330:i=3:GOSUB 330:i=7:GOSUB 330
140 LOCATE 13,20::i=0:GOSUB 330:i=4:G
OSUB 330:i=2:GOSUB 330:i=6:GOSUB 330
150 PAPER 5:PEN 1
160 IF INKEY(8)>-1 THEN colour1=(colo
ur1+1) MOD 16
170 IF INKEY(1)>-1 THEN colour2=(colo
ur2+1) MOD 16
180 WEND
190 END
200 REM -----------------------
210 REM initialise
220 DEFINT a-z
230 BORDER 1:PAPER 5:CLS
240 MOVE 0,0:DRAW 0,398,4:DRAW 636,39
8:DRAW 636,0:DRAW 0,0
250 MOVE 0,46:DRAW 640,46:MOVE 0,270:
DRAW 640,270:MOVE 320,270:DRAW 320,40
0
260 PAPER 3:PEN 12:LOCATE 2,24:PRINT
" ";CHR$(242);" ";CHR$(243);" change
pixel ":PAPER 5
270 PEN 1:LOCATE 4,3:PRINT "Byte":LOC
ATE 13,2:PRINT "Pixels"
280 LOCATE 6,12:PRINT "Bit Pattern"
290 PEN 2:LOCATE 7,15:PRINT "76543210"
300 LOCATE 5,19:PRINT "1537    0426"
310 colour1=0:colour2=0
320 RETURN
330 REM -----------------------
340 IF (byte AND 2^i) THEN PEN 1 ELSE
 PEN 5
350 PRINT CHR$(233);
360 RETURN
```

Program II

80	Sets pixels.
90-100	Print large pixels and pen numbers.
110	Gets byte and print hex value.
120-140	Print bit patterns.
160-170	Change pens.
210-320	Set up display, initialise variables.
340	Prints block if bit is set in byte.

30	Dims array for data.
50-100	Plot points and get hex values for pens.
170-230	Print values.

bytes later. There seems to be a part of the memory that is not displayed.

As you'll see, the poking starts at the top of the screen again – this time filling the second row of pixels on the first line, followed by the second row of pixels on the second line and so on.

Keep an eye on the address printed and note when it runs on to the next line – it's &C000 at the start of the first line, &C050 at the start of the second line, &C0A0 on the next and so on.

Each line starts &50 lower than the previous one. That is, if you know the address of a particular pixel in a character cell, the corresponding pixel in the cell below will have an address exactly &50 higher.

When it disappears off the bottom and reappears at the top, you'll see that the second row of pixels is at &C800, the third at &D000 and so on. Each pixel is separated by &800 in the vertical direction – except for the bottom pixel of each character cell and the top pixel of the character cell immediately below (we'll come to this in a later article).

As you'll see from Program I, a Mode 0 character is stored in 32 bytes, 8 rows of 4 bytes with each row separated by &800. Figure I shows the pattern. As each character is 8 pixels wide – this holds for all modes – a single byte must contain the colour information for two pixels.

It's easy to work out why. We saw earlier that each row of pixels is &50 bytes in length. Since there are 160 pixels across the Mode 0 screen, and as 160 divided by &50 is 2, we have two pixels per byte.

How is the information coded? Program II will help here. The two pixels in the first byte of the screen memory can be set to any of the sixteen pens by pressing the left or right cursor keys.

They can just be seen in the top left corner of the screen, but in case you find these hard to see they are repeated eight times normal size on the right, with the pen number printed below. The value of this byte is printed in hex and its binary bit pattern is shown.

Try altering the colour of the pixels and look for a pattern in the hex value or binary pattern. It doesn't seem to make sense does it?

Four bits can be used to store the numbers 0 to 15 – %0000, %0001, %0010 . . . %1111 in binary.

So a byte, consisting of 8 bits, can store the pens (0-15) for two pixels. It

pixel row	pixels 0, 1	pixels 2, 3	pixels 4, 5	pixels 6, 7
0	&C000	&C001	&C002	&C004
1	&C800	&C801	&C802	&C803
2	&D000	&D001	&D002	&D003
3	&D800	&D801	&D802	&D803
4	&E000	&E001	&E002	&E003
5	&E800	&E801	&E802	&E803
6	&F000	&F001	&F002	&F003
7	&F800	&F801	&F802	&F803

Figure I: Memory locations corresponding to the top left screen character in Mode 0

would be logical to use the first four bits for the first pixel and the second four for the second pixel.

However it's not quite so simple. Bits 1, 5, 3 and 7 store the pen for the left pixel, and bits 0, 4, 2 and 6 store the pen for the right pixel. Program II prints the two nybbles (4 bits or half a byte), for each pixel near the bottom of the screen.

Press the left cursor key and the left nybble will cycle through the 16 pens %0000, %0001, %0010 and so on. Similarly the right nybble can be changed by pressing the right cursor key.

A multicoloured character could be designed on paper, and each horizontal pair or pixels could be set using this program and the data noted. It could then be stored at any position on the screen. But this would be a very clumsy method to use, so in a later article in this series we will employ a sprite designer to make it a bit easier.

You should now be able to see why in Program I storing &C0 in the screen memory, line 120, coloured it yellow. Use Program II to set both pixels to yellow and look at the hex value and bit patterns – &C0 and %0001, %0001.

Program III prints a complete table of hex values for all combinations of left and right pixels. Look down the left column for the left pixel pen, then along to the right pixel pen and read off the hex value. If this number is stored in the screen memory the two pixels will be displayed in the pens chosen.

● *That's all for now. Next month we will be starting with a few simple machine code routines involving the screen.*

```
10 REM PROGRAM III
20 a$=" Left Pixel Pen "
30 DIM byte(15,15)
40 MODE 0
50 FOR i=0 TO 15
60 FOR j=0 TO 15
70 PLOT 0,398,i:PLOT 4,398,j
80 byte(i,j)=PEEK(&C000)
90 NEXT
100 NEXT
110 MODE 2:PRINT
120 PRINT TAB(15)"Hexadecimal values
for left and right pixels in mode 0"
130 PRINT
140 PRINT TAB(35)"Right Pixel Pen..."
150 PRINT
160 PRINT TAB(21)"0  1  2  3  4  5  6
  7  8  9 10 11 12 13 14 15"
170 FOR i=0 TO 15
180 PRINT
190 PRINT TAB(13)MID$(a$,i+1,1);" ";i
;TAB(19);
200 FOR j=0 TO 15
210 PRINT " ";HEX$(byte(i,j),2);
220 NEXT
230 NEXT
240 BORDER 3
250 MOVE 0,0:DRAW 0,398,1:DRAW 638,39
8:DRAW 638,0:DRAW 0,0
```

Program III

Let your fingers do the talking...with this specia

Now you can teach your Amstrad to talk!

How it works

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

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

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

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

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

In fact this system supports four different modes of use.

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

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

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

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

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

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

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

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

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

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

A	AE	26	fat	F	FF	40	fire	NG	NG	44	bans	TH	TH	29	thin
A	AY	20	great	G	GG2	61	go	O	O	23	cot	TH	DH1	18	they
AI	AIR	47	hair	IG	GG3	34	wig	O	OW	53	snow	TH	DH2	54	bathe
AR	AR	59	farm	GU	GG1	36	guest	O	UW1	22	do	U	AX	15	succeed
AU	AO	24	aught	GE	ZH	38	beige	OO	UW2	31	food	UH	OO	30	cook
B	BB1	28	rib	H	HH1	27	he	OR	OR	58	store	U	YY1	49	compute
B	BB2	63	big	H	HH2	57	hoe	OU	OW	32	ouch	V	VV	35	even
C	KK1	42	common	I	IH	12	fitting	OY	OY	5	toy	W	WW	46	wool
´C	C	8	uncle	I	Y	6	sky	P	PP	9	pub	WH	WH	48	whig
K	KK2	41	sky	IR	ER2	52	bird	R	RR1	14	real	Y	YY2	25	yes
CH	CH	50	church	J	JH	10	jury	R	RR2	39	brain	Z	ZZ	43	zoo
D	DD1	21	could	L	LL	45	luck	R	YR	60	fear	PA1	PA1	0	10 mS
D	DD2	33	do	L	EL	62	angle	S	SS	55	sat	PA2	PA2	1	30 mS
E	EH	7	bend	M	MM	16	milk	SH	SH	37	shirt	PA3	PA3	2	50 mS
E	EE	19	see	N	NN1	11	earn	T	TT1	17	its	PA4	PA4	3	100 mS
ER	ER	51	cater	N	NN2	56	no	T	TT2	13	top	PA5	PA5	5	200 mS

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

cial Computing with the Amstrad reader offer!

Save £5 (plus FREE post and packing)

Offer price only £34.95

Look at what this package offers you:

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

Eight additional Basic commands

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

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

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

I wish to pay by

□ Access Card No. ____________________

□ Visa Card No. ____________________

Signed ____________________

Name ____________________

Address ____________________

____

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

CA8

Allow 28 days for delivery

Steam along and build up your character the easy way

Part VII of the Amstrad graphics series by GEOFF TURNER and MICHAEL NOELS

IF you can cast your mind back that far, you'll remember that last month we entered the world of user defined graphics. I hope by now you've had some practice in designing your own characters.

However if you're struggling with the calculations involved don't worry, you'll find our first program will help.

It is a simple version of a character generator, which takes all the hard work out of designing characters. The May 1985 issue of *Computing with the Amstrad* contains an all-singing, all dancing character generator if you're feeling ambitious. For the present, though, Program I will suffice.

When you run it you'll find that it displays a large 8 by 8 character grid with a movable cursor. The Amstrad's cursor keys are used to guide this around the grid to any one of the 64 squares.

When you wish to fill or plot a square simply press the Copy key, and the currently selected square will be filled. The square is unplotted by pressing the Copy key once more. Designing your own characters now becomes easy. You just fill in the squares as needed to create your own character.

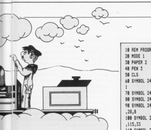
As you use it you'll see that the program takes care of all the necessary calculations. The current value of each row is displayed alongside the grid. Just for reference, the binary values are also displayed.

Notice how the ones and zeros of the binary numbers correspond to the squares of the grid. A one means that that square is filled, a zero that it is left empty.

The character is printed below the grid in its actual size, so that you can see how it will look in your programs. Once you're satisfied with a character you should make a note of the eight values which make up the design. These can then be used with the

```
10 REM PROGRAM I
20 MODE 1
30 SYMBOL 240,0,0,0,0,0,0,0,0
40 SYMBOL 252,255,129,129,129,129,129
,129,255
50 SYMBOL 253,255,255,255,255,255,255
,255,255
60 SYMBOL 254,255,129,129,153,153,129
,129,255
70 SYMBOL 255,255,255,255,231,231,255
,255,255
80 FOR row=1 TO 8
90 FOR column=1 TO 8
100 LOCATE column,row
110 PRINT CHR$(252)
120 NEXT
130 LOCATE 12,row
140 PRINT USING "###";value(down)
150 LOCATE 18,row
160 PRINT BIN$(value(down),8)
170 NEXT
180 column=1:row=1
190 LOCATE column,row
200 PRINT CHR$(254)
210 WHILE -1
220 oldcolumn=column
230 oldrow=row
240 GOSUB 460 :REM update numbers
250 GOSUB 570 :REM print character
260 IF INKEY(0)=0 THEN row=row-1
270 IF row=0 THEN row=8
280 IF INKEY(2)=0 THEN row=row+1
290 IF row=9 THEN row=1
300 IF INKEY(8)=0 THEN column=column-
1
310 IF column=0 THEN column=8
320 IF INKEY(1)=0 THEN column=column+
1
330 IF column=9 THEN column=1
340 LOCATE oldcolumn,oldrow
350 PRINT CHR$(252+status(oldcolumn,o
ldrow))
360 LOCATE column,row
370 PRINT CHR$(254+status(column,row)
)
380 IF INKEY(9)=0 THEN GOSUB 400:REM
change pixel
390 WEND
400 REM plot / unplot pixel
410 PRINT CHR$(7)
420 status(column,row)=ABS(status(col
umn,row)-1)
430 LOCATE column,row
440 PRINT CHR$(254+status(column,row)
)
450 RETURN
460 REM update numbers
470 down=row
480 value(down)=0
490 FOR across=1 TO 8
500 IF status(across,down)=1 THEN val
ue(down)=value(down)+(2^(8-across))
510 NEXT
520 LOCATE 12,down
530 PRINT USING "###";value(down)
540 LOCATE 18,down
550 PRINT BIN$(value(down),8)
560 RETURN
570 REM print character
580 SYMBOL 240,value(1),value(2),valu
e(3),value(4),value(5),value(6),value
(7),value(8)
590 LOCATE 5,12
600 PRINT CHR$(240)
610 RETURN
```

Program I



```
10 REM PROGRAM III
20 MODE 1
30 PAPER 2
40 PEN 3
50 CLS
60 SYMBOL 240,0,127,127,17,17,17,31,3
1
70 SYMBOL 241,0,0,0,0,24,60,60,255
80 SYMBOL 242,0,0,30,12,12,12,12,254
90 SYMBOL 243,127,127,127,127,127,127
,28,8
100 SYMBOL 244,255,255,255,255,33,115
,115,33
110 SYMBOL 245,255,255,255,255,8,156,
156,8
120 Top$=CHR$(240)+CHR$(241)+CHR$(242
)
130 Bottom$=CHR$(243)+CHR$(244)+CHR$(
245)
140 Engine$=Top$+CHR$(10)+CHR$(8)+CHR
$(8)+CHR$(8)+Bottom$
150 LOCATE 18,12
160 PRINT Engine$
170 LOCATE 18,13
180 PRINT CHR$(243);CHR$(244);CHR$(24
5)
190 WHILE NOT true:WEND
```

Program III

SYMBOL command to produce the same character in your own programs.

Now we'll take a look at a few techniques which you may find useful when using your characters in programs.

In Program II, we've designed a steam engine. Reset the Amstrad to clear Program I before you run Program II. We wanted the character to be quite large, so we used six individual characters to make up the finished display. The overall size of the design is three characters across by two characters down.

```
10 REM PROGRAM II
20 MODE 1
30 PAPER 2
40 PEN 3
50 CLS
60 SYMBOL 240,0,127,127,17,17,17,31,3
1
70 SYMBOL 241,0,0,0,0,24,60,60,255
80 SYMBOL 242,0,0,30,12,12,12,12,254
90 SYMBOL 243,127,127,127,127,127,127
,28,8
100 SYMBOL 244,255,255,255,255,33,115
,115,33
110 SYMBOL 245,255,255,255,255,8,156,
156,8
120 LOCATE 18,12
130 PRINT CHR$(240);CHR$(241);CHR$(24
2)
140 LOCATE 18,13
150 PRINT CHR$(243);CHR$(244);CHR$(24
5)
160 WHILE NOT true:WEND
```

Program II

Having designed the shapes, the next problem was how to get them on to the screen in the correct positions. The top row of the steam engine consists of characters 240, 241 and 242, while the bottom row is made up from 243, 244 and 245.

There are several ways of printing this character. Probably the most obvious is to move the text cursor to the required position, then print the first three characters next to each other. The cursor can then be moved to the correct position on the next row down with a cunning LOCATE, and the final three characters printed.

Program II uses this method. While it obviously works, it's not very clear what the program is doing. This would be particularly so if it were part of a longer listing.

A much better way is to combine the six separate characters into a string variable, and then print the string all in one line. Program III demonstrates how this is done.

At line 120 we've concatenated or linked the top three characters into one string and called it *Top$*. The remaining three characters are combined into *Bottom$* at line 130. Finally *Top$* and *Bottom$* are joined in line 140.

Notice that we have included some new characters in the finished string. Can you remember what CHR$(10) and CHR$(8) do? These are the cursor control codes. Here they are used to place the cursor in the correct position ready to print *Bottom$*. We are moving the cursor one row down (CHR$(10)), and three positions back (CHR$(8)). This places it directly under the beginning of *Top$*.

Finally we've given a meaningful name to the completed string and called it *Engine$*. Now whenever we wish to print the complete character, it's simply a matter of using the command.

PRINT Engine$

using LOCATE to give the position required.

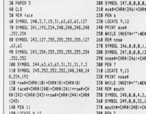
On the face of it, it does seem a lot of work to produce the same result as Program II. However you will find as your programs become longer and more complex that it will be much easier to use this method.

One of the drawbacks with printing characters is that they can normally be printed in the current pen and paper colours. The foreground is printed in the pen colour, while the background is printed in the paper colour.

It might, however, be that we want to produce a character consisting of several different colours. We may want to print a pink face, with blue eyes and red lips. The trouble is that if we design the whole character in Basic using the SYMBOL command there's no facility to plot it in several colours.

To get around this we need to design several characters, each making up a different part of the face. The main symbol would be the head shape, other symbols representing the eyes, nose and mouth.

It seems fairly plausible that if we first select a suitable ink colour to ▶



```
10 REM PROGRAM IV
20 MODE 0
30 PAPER 5
40 CLS
50 REM face
60 SYMBOL 240,3,7,15,31,63,63,63,127
70 SYMBOL 241,192,224,240,248,248,248
,252,254
80 SYMBOL 242,127,255,255,255,255,127
,63,63
90 SYMBOL 243,254,255,255,255,255,254
,252,252
100 SYMBOL 244,63,63,63,31,31,31,7,3
110 SYMBOL 245,252,252,252,248,248,24
8,224,192
120 row$=CHR$(10)+CHR$(8)+CHR$(8)
130 face$=CHR$(240)+CHR$(241)+row$+CH
R$(242)+CHR$(243)+row$+CHR$(244)+CHR$
(245)
140 PEN 11
150 LOCATE 9,12
160 PRINT face$
170 WHILE INKEY$="":WEND
180 REM eyes
190 SYMBOL 246,0,0,0,0,4,14,0,0
200 SYMBOL 247,0,0,0,0,32,112
210 eye$=CHR$(246)+CHR$(247)
220 PEN 6
230 LOCATE 9,12
240 PRINT eye$
250 WHILE INKEY$="":WEND
260 REM nose
270 SYMBOL 246,0,0,0,1,1,0,0,0
280 SYMBOL 247,0,0,0,128,128,0,0,0
290 nose$=CHR$(246)+CHR$(247)
300 PEN 7
310 LOCATE 9,13
320 PRINT nose$
330 WHILE INKEY$="":WEND
340 REM mouth
350 SYMBOL 248,0,0,4,3,0,0,0,0
360 SYMBOL 249,0,0,32,192,0,0,0,0
370 mouth$=CHR$(248)+CHR$(249)
380 PEN 3
390 LOCATE 9,14
400 PRINT mouth$
```

Program IV

print the face, then switch to blue ink and move the cursor, we can overprint the eyes in blue over the face. Unfortunately, as you'll see in Program IV, this method doesn't work satisfactorily.

The program works in stages, each requiring a key to be pressed before it moves on to the next part. First it prints the overall face shape. So far, so good. However when the eyes are printed the black background of the eye characters overwrites the previously printed pink face.

This is because characters printed at the text cursor always print a character square whose foreground and background is made up of the currently selected pen and paper. The same problem occurs with the nose and the mouth.

Happily there is a way of overcoming this. We can print characters in what's known as transparent mode, in which the background or paper colour of a character isn't printed. It is, in effect, transparent, so anything that was on the screen previously shows through.

This means that if we put the Amstrad into transparent mode we could put our blue eyes on to the face but still have the pink of the face showing through the transparent background of the eye character.

There is no Basic keyword for transparent printing. We have to use control code 22 followed by a second number which turns it on or off. Transparent mode is turned on by:

```
PRINT CHR$(22);CHR$(1)
```

and turned off again by:

```
PRINT CHR$(22);CHR$(0)
```

When using transparent mode you should always remember to turn it off again at a suitable point in your program or strange things may happen to your displays.

To modify Program IV, add the following new lines:

```
175 PRINT CHR$(22);CHR$(1)
410 PRINT CHR$(22);CHR$(0)
```

The eyes, nose and mouth will now be printed without destroying any part of the face. This is because the paper they are printed on is see-through, letting the previous background show. So using this method we can build up multi-coloured characters.

Incidentally, the character could have been linked together into one long compilation of a string variable, as we did with our steam engine.

If you examine the list of control codes in the User Manual you will see that there are also control characters to select pen and paper inks.

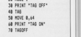
Perhaps you might like to produce a single string variable which will print the multicolour face all in one go. It's possible and, if you were using a lot of faces on the screen, could make things easier.

Another useful facility when printing characters on the screen is the ability to print at the position of the graphics cursor instead of the text cursor. The advantage of this is that characters may be then printed at any graphics coordinate and we are not limited to the usual, rather clumsy, text cursor locations.

This greater definition can be useful when labelling diagrams and graphs and so on.

To achieve printing at the graphics cursor we use the TAG command. Program V demonstrates how it affects the position of the printed text.

The first print command at line 30 occurs at the position of the text cursor. Having issued the TAG command at line 40 you'll see that



```
10 REM PROGRAM V
20 MODE 1
30 PRINT "TAG OFF"
40 TAG
50 MOVE 0,64
60 PRINT "TAG ON"
70 TAGOFF
```

Program V

the next print command is obeyed down at the bottom of the screen where the graphics cursor has been positioned by line 50.

It was necessary to move the graphics cursor slightly up the screen because if it was left at its home position, (0,0), the printed text would be below the bottom of the screen. Leave out line 50 and see what happens.

Notice that when printing at the graphics cursor the ink used belongs to the current graphics pen and not the text pen. This presents a problem if we wish to change the colour of text printed at the graphics cursor.

There's no direct way of changing the graphics pen except by plotting or drawing. This can be a nuisance as you don't always want to draw something over your nice display just to change the graphics colour.

One solution is to perform a dummy PLOT at a point off the screen. If you add a line like:

```
45 PLOT 1000,1000,3
```

to Program V then the graphics pen will be changed to number 3, and all text printed at the graphics cursor will now be in pen 3.

Observant readers will have noticed another effect of printing at the graphics cursor. After printing any characters the arrow symbols representing line feed and carriage return are also displayed. To suppress these control symbols we must place a semicolon at the end of any print statements followed by a TAG. Try doing this in Program V.

In the last article we mentioned that it was unlikely that you would need to redesign all the alphanumeric character set. But there is one application where this could be useful.

By now you should be familiar with the different screen modes. You're probably well aware that if you want a large selection of colours you need to use Mode 0. Unfortunately this suffers from only allowing 20 characters per line.

Wouldn't it be nice to have the possibility of 16 colours and 40 characters per line? Well you may be surprised to learn that it can be done (well, almost).

In Mode 0 the characters are displayed twice their normal width, where "normal" means Mode 1. If we could redesign any character to occupy only the left hand side of a character cell then it would appear normal size when printed in Mode 0. We could do this for each letter of the alphabet, using the character generator program, and we would then have a set of characters which were only half normal width.

We could therefore print a lot more than 20 of these characters on one line in Mode 0.

There is, of course, a snag (there always is). If we print the characters at the text cursor they will still be printed at intervals relative to Mode 0 character cells. This means that we're still stuck with 20 per line.

If we redefined our characters in this way and used the command:

```
PRINT"ABC"
```

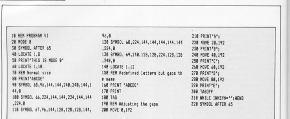
in Mode 0, it would appear as:

```
A B C
```

with a gap between each letter.

Have you worked out how to overcome this problem? Remember that we can use the TAG command and then print at graphics X and Y coordinates. So instead of the above we could first print A, MOVE the graphics cursor just to the right of the A and then print B. In effect, we close up the gaps between the characters.

Program VI demonstrates this. We've only redesigned a few charac-



```
10 REM PROGRAM VI
20 MODE 0
30 SYMBOL AFTER 65
40 LOCATE 1,8
50 PRINT"THIS IS MODE 0"
60 LOCATE 1,10
70 REM Normal size
80 PRINT"ABCDE"
90 SYMBOL 65,96,144,144,240,240,144,1
44,0
100 SYMBOL 66,224,144,144,224,144,144
,224,0
110 SYMBOL 67,96,144,128,128,128,144,
96,0
120 SYMBOL 68,224,144,144,144,144,144
,224,0
130 SYMBOL 69,240,128,128,224,128,128
,240,0
140 LOCATE 1,12
150 REM Redefined letters but gaps th
e same
160 PRINT "ABCDE"
170 PRINT
180 TAG
190 REM Adjusting the gaps
200 MOVE 0,192
210 PRINT"A";
220 MOVE 20,192
230 PRINT"B";
240 MOVE 40,192
250 PRINT"C";
260 MOVE 60,192
270 PRINT"D";
280 MOVE 80,192
290 PRINT"E";
300 TAGOFF
310 WHILE INKEY$="":WEND
320 SYMBOL AFTER 65
```

Program VI

ters to illustrate the technique. Obviously it's a lot harder to print text in this way, but it can be very useful when printing just a few words such as SCORE, BONUS or LIVES on the screen.

Unfortunately some characters such as M and W are difficult to portray in half normal width, so these will need to occupy a slightly wider space. The movement of the graphics cursor has to be adjusted to take account of this.

In Program VI there are 20 pixels per character, which will actually allow us 32 characters per line. Although we can't manage the full 40 characters per line we can at least improve on the standard 20.

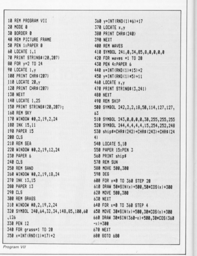
A variation on this technique could be used to produce larger characters in Mode 2 – normally 80 characters per line – which would be useful for producing large headings or titles. In this case we would need to spread each character over two character positions and print the two new characters side by side.

That's something for you to experiment with.

To finish for this month, Program VII brings together many of the topics covered in all the previous chapters. It's a sort of refresher course.

See if you can work through the program and understand all the techniques used. The program is broken down into sections for clarity, with appropriate REM statements so it shouldn't be too hard.

And by the time you've finished you should be ready for next month's discussion of logical colours.



```
10 REM PROGRAM VII
20 MODE 0
30 BORDER 0
40 REM PICTURE FRAME
50 PEN 1:PAPER 0
60 LOCATE 1,1
70 PRINT STRING$(20,207)
80 FOR y=2 TO 24
90 LOCATE 1,y
100 PRINT CHR$(207)
110 LOCATE 20,y
120 PRINT CHR$(207)
130 NEXT
140 LOCATE 1,25
150 PRINT STRING$(20,207);
160 REM SKY
170 WINDOW #0,2,19,2,24
180 INK 15,11
190 PAPER 15
200 CLS
210 REM SEA
220 WINDOW #0,2,19,12,24
230 PAPER 6
240 CLS
250 REM SAND
260 WINDOW #0,2,19,18,24
270 INK 13,15
280 PAPER 13
290 CLS
300 REM GRASS
310 WINDOW #0,2,19,2,24
320 SYMBOL 240,64,32,34,148,85,100,60
,126
330 PEN 12
340 FOR grass=1 TO 20
350 x=INT(RND(1)*17)+2
360 y=INT(RND(1)*6)+17
370 LOCATE x,y
380 PRINT CHR$(240)
390 NEXT
400 REM WAVES
410 SYMBOL 241,0,34,85,8,0,0,0,0
420 FOR waves =1 TO 20
430 PEN 4:PAPER 6
440 x=INT(RND(1)*15)+2
450 y=INT(RND(1)*5)+11
460 LOCATE x,y
470 PRINT STRING$(3,241)
480 NEXT
490 REM SHIP
500 SYMBOL 242,2,2,18,50,114,127,127,
63
510 SYMBOL 243,0,0,0,0,30,255,255,255
520 SYMBOL 244,4,4,4,4,15,254,252,248
530 ship$=CHR$(242)+CHR$(243)+CHR$(24
4)
540 LOCATE 5,10
550 PAPER 15:PEN 3
560 PRINT ship$
570 REM SUN
580 MOVE 500,300
590 DEG
600 FOR x=0 TO 360 STEP 20
610 DRAW 50*SIN(x)+500,50*COS(x)+300
620 MOVE 500,300
630 NEXT
640 FOR x=0 TO 360 STEP 4
650 MOVE 30*SIN(x)+500,30*COS(x)+300
660 DRAW 30*SIN(360-x)+500,30*COS(360
-x)+300
670 NEXT
680 GOTO 680
```

Program VII

More than ten million places set for pi ...

Aleatoire goes back to the roots of the pi calculation problem

LAST year it was announced that pi had been calculated to over 10,013,395 places by a Japanese team at the University of Tokyo using 24 hours on an Hitachi computer. That's 450 megaflops a second.

Pi is an irrational number (i.e. infinitely long and random) because it cannot be described by the ratio between two finite integers.

To give a simple example of an irrational number, consider the square root of 2.

Assume that there exist two integers, A and B, with NO common factor such that A/B=SQRT(2) or A*A=2*B*B therefore A must be an even integer.

Therefore let A=2*a but this means that B*B=2*a*a which means that B must ALSO be an even number which contradicts the assumption that A and B had no common factor THEREFORE A and B cannot exist THEREFORE the square root of 2 is infinitely long and random.

The Greeks (well some of them) were aware of irrational numbers and disliked them intensely.

Nevertheless they, and particularly Archimedes, used geometical methods to calculate the famous 22/7 approximation we are taught in school, which is accurate to only three places.

The first man to achieve an accuracy of 100 places was John Machin, a British mathematician, in 1706.

In the language of mathematics, Machin discovered the following equation:

```
45 degrees = pi/4 = 4*ARCTAN(1/5) -
ARCTAN(1/239)
```

It is possible to prove this relationship using only the simple geometry of similar triangles applied to the construction in Figure I.

The approximate values for ARCTAN can be obtained by the expansion of Gregory's series thus:

```
Pi/4 = 4*(1/5 - 1/(3*5**3) + 1/(5*5
** 5) - ...) - (1/239 - 1/(3*239**3)
+ 1/(5*239**5) - ...)
```

which is an expression you can try immediately on your computer or calculator giving the answer pi=3.14162 which is accurate to five places.

Machin apparently treated the calculation as a hobby and happily whiled away his time for four years before publishing the result.

In 1853, William Shanks, using Machin's formula, published his value for pi to 607 decimal places.

Unfortunately he had made an error at the 459th place, so he returned to his calculations and 20 years later published a new result to 707 places – it was the last number on the subject until 1946.

D.F. Ferguson, of the Royal Naval College, had spent much of his spare time during the war calculating pi to 620 places using the formula:

```
Pi/4 = 3*ARCTAN(1/4) + TAN(1/20) +
TAN(1/1985)
```

and found that Shanks' 707 approximation was incorrect beyond 527 places.

In order to verify Ferguson's new value, two American mathematicians, John Wrench and Levi Smith, did an independent check using Machin's formula.

The result was compared with Ferguson's new calculations (to 710 places) and found to agree.

Shanks had made at least two errors. One of them was the omission of a zero in evaluating the term

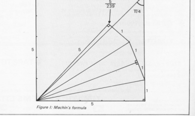


Figure I: Machin's formula

[497*5**497]. Also, in the 531st decimal place, instead of .0084 . . . 82897, Shanks had "carelessly" written .0848 . . . 28973.

Careless is hardly the right word, but this marked the end of human calculation as the computer entered the competition.

To quote from a paper by George Reitwiesner:

"Early in June, 1949, Professor John Von Neumann expressed an interest in the possibility that the ENIAC might sometime be employed to determine the value of pi and e to many decimal places with a view toward obtaining a statistical measure of the randomness of the distribution of the digits, suggesting the employment of Machin's formula.

"Since the possibility of official time was too remote for consideration, permission was obtained to execute these projects during two summer holiday weekends when the ENIAC would otherwise stand idle, and the planning and programming of the projects was undertaken on an extra-curricular basis . . ."

The computation took place in July, 1949, and took 70 hours' machine time.

The result, rounded to 2,035 places, was published, and the newspapers quickly picked up the story – "Electronic Brain first through 1,000 digit barrier" . . . "Electronic Brain performs in a few hours calculations that would take a mathematician 100 years".

The Electronic Brain had arrived.

By 1958, a Frenchman, F. Genuys, had taken pi to 10,000 places on an IBM 704 and in 1961 Daniel Shanks and John Wrench reached 100,000 places.

In their report, Shanks and Wrench discussed the 1,000,000-place calculation and predicted that within five to seven years a computer would become available to perform such a calculation – a machine 100 times faster and more reliable than the IBM 7090 they had used.

Their prediction was fairly accurate.

In May, 1973, the French mathematicians Jean Guilloud and Mlle Martine Bouyer of the Commissariat à l'Energie Atomique achieved the million mark on an American computer – a CDC7600 – of the required power.

You may find it interesting to compare the power of your Amstrad with the giant ENIAC of 35 years ago.

Next month I will give an example in Basic to calculate pi to more than 2,000 places and meanwhile invite any budding programmers to try to beat three hours.

Aleatoire

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

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

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

£8.95! post free

Read the book – then play the game!

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

Name ______________________________

Address ______________________________

____

____

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

No. ______________________

Signed ______________________

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

CA8

The masked bytes are taking control

MIKE BIBBY continues his series on how your CPC464 works

IN the past months we learned a lot about the binary system – the numbers our micro works in.

We have seen that its memory is divided up into bytes – a set of eight two-state, binary units called bits. Each bit can have the value 1 or 0.

If a bit has the value 1 we say it is set. If a bit has the value 0 we say it is clear.

As we're dealing with eight bits at a time, we can use various combinations of the bits in a byte to code any whole number (integer) in the range 0 to 255.

To do this we associate a code number with each bit. Figure I shows the scheme.

Our eight bits are labelled b7 . . . b0 and the numbers associated with each number are shown above each bit (the more mathematical among you will see that they're in ascending powers of two).

To discover the value coded in a byte we simply add the numbers associated with every bit that is set (1), ignoring all clear bits (0).

So:

```
%10101000
```

codes the number:

```
128 + 32 + 8 = 168
```

We also learned to do tricks with, or to put it more properly, manipulate, binary numbers. We could create the complement of a number – a sort of binary opposite – by changing every clear bit to set ("setting" the bit) and changing every set bit to clear ("clearing" the bit).

So the complement of the above number:

```
%10101000
```

gives us:

```
%01010111
```

We can add and subtract binary numbers, as well as multiply and divide. We learned other ways of combining them too, with the logical operators AND, OR, EOR.

EOR, which stands for Exclusive OR, is also called XOR.

When combining two binary numbers under the influence of these operators we compare each bit in one number with the corresponding bit of the other.

Then, according to a rule which depends on the operator we're using, we decide whether that particular bit (the result bit) in the "answer" byte is set or clear. Table I shows the rules for the operators.

As we've said, a micro's memory is divided into byte-sized compartments, called memory locations. Each location has a number associated with it so we know which one we're talking about.

These numbers are known as memory addresses.

Much of what a microprocessor does involves moving information – in the form of binary numbers – from one location to another.

AND Sets the result bit only if both bits compared are set, otherwise the result bit is clear.

OR Sets the result bit if either or both the bits compared are set. Only if both bits compared are clear is the result bit clear.

EOR Sets the result bit if the bits being compared differ in value. If the XOR bits compared are identical, the result bit is cleared.

Table I: Rules for logical operators.

If you cast your mind back to earlier articles, I said that each bit was like a switch – its two values 1 and 0 could be used to signify that the switch was on or off respectively.

Imagine that we could wire up one of our bits to a machine's on/off switch. Then by setting that bit we could switch the machine on, and by clearing it we could switch it off.

This sort of thing is possible, though we'd need to use some clever electronics. In fact, since we deal with eight bits at a time, we could arrange things so that a single byte controlled the on/off status of eight separate machines – each machine m7,m6 . . . ▶

128	64	32	16	8	4	2	1
b7	b6	b5	b4	b3	b2	b1	b0

Figure I: Values associated with bit positions

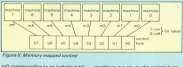


Figure II: Memory mapped control

m0 corresponding to an individual bit of that byte, b7,b6 . . . b0. We'll term that byte the control byte.

We call such arrangements memory-mapped output, since what we put in memory maps, or sets the pattern for, what happens in the outside world. Most microprocessors support this or some similar sort of output. Figure II shows the type of scheme we mean.

Assuming we've got things connected up properly, if we then load the control byte with:

%11111111

all the machines would be on. Remember that if a bit is set the corresponding machine is on. If we want to switch all the machines off, we can load the control byte with:

%00000000

And, of course, we can have any on/off pattern of machines, setting or clearing the relevant bits by loading the control byte with new numbers.

Loading it with:

%11110000

is one way of switching off half the machines.

Sometimes, though, we might want to switch a particular machine or two on or off without knowing (or caring) whether the others are on or off.

This means we need some way of affecting only the bits controlling those machines, while leaving the others unchanged.

Suppose we wanted to switch off a machine – say m6. We can do this by making b6 of the control byte zero.

To clear that one bit to zero we AND the control byte with another byte – called the mask – the bits of which are set (1) except for b6, which will be 0. That is, we AND the control byte with:

%10111111

We then make this result our new control byte, and off the machine goes.

To see how it works in practice, let's assume that initially all the machines are on, so the control byte is:

%11111111

To switch machine m6 off we must AND it with:

%1011111

The sum is:

%11111111 control byte
AND %10111111 mask
%10111111 new control byte

As you can see, the outcome is that when we update the control byte with the result, m6 is switched off while the others remain on.

The trick isn't hard to see. Let's consider things from the point of view of bits in the mask. If the bit is a 1, when you AND it with the relevant control bit the resulting bit is the same as the control bit. That is, ANDing a bit with 1 leaves that bit unchanged.

Think about it. If the control bit were 1, then as 1 AND 1 = 1, you're left with 1. The bit's unchanged.

If, on the other hand, the control bit were 0 then, as 0 AND 1 = 0, the bit remains unchanged as 0.

In other words bits in the mask with 1 in them leave the corresponding control bit unchanged.

So for machines whose on/off status we don't want to alter – we may not even know if they're on or off – we set the corresponding bit in the mask to 1.

However if the bit in the mask were clear (0) it wouldn't matter what the state of the original control bit was – the result would still be 0.

Say the control bit was 1, then as 1 AND 0 = 0 the resulting bit is a 0.

Alternatively, if it were 0, since 0 AND 0 = 0 the resulting bit is again 0.

So bits in the mask with 0 in them set the corresponding bits in the result byte to 0.

This means to switch specific machines off we construct a mask consisting of 1s for the machines we wish to leave unchanged and 0s for the machines we want off – in the appropriate bit positions.

We then AND the mask with the control byte and then make the resulting byte the new control byte.

Fine, but how do we switch on specific machines?

Well, we update the control byte by ORing it with another mask. This time we put 1 in the bits corresponding to the machine we want on, and 0 in the bits corresponding to the machines whose on/off status we wish to leave unchanged.

This works, since when you OR a bit (whether 0 or 1) with another bit whose value is 1, the answer is 1. That is **0 OR 1 = 1** and **1 OR 1 = 1.**

So using a 1 in the relevant bit of an OR mask will set the corresponding result bit. When this becomes the new control byte the corresponding machine will be turned or left on.

On the other hand, ORing a bit in the control byte (no matter what value) with 0 leaves that bit totally unchanged since **1 OR 0 = 1** and **0 OR 0 = 0.**

So when we OR the bits of the mask that are 0 leave the corresponding bits of the control byte unchanged.

This means, to switch specific machines on we use a mask consisting of 0s for the machines we wish to leave unchanged, and 1s for the machines we want on – in the appropriate bit positions.

We then OR that mask with the control byte and make the resulting byte the new control byte.

Hence, to ensure that m6 is definitely on, we OR the control byte with:

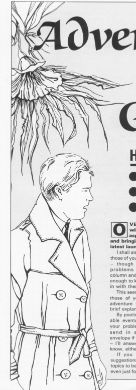
%01000000

For example, if m6 is off, and all the rest on, to switch m6 on we do the following:

%10111111 control byte
AND %01000000 mask
%11111111 new control byte

Of course, both AND and OR have uses for the micro enthusiast other than controlling machines.

● *Next month we'll have a look at some, as well as uses of XOR/EOR.*



Adventuring with Gandalf

He will . . .

- Bring you the latest news
- Answer your problems
- Let you have YOUR say

OVER the coming months I will be discussing various aspects of adventures – and bringing you news of all the latest launches.

I shall also be endeavouring to help those of you who are having problems – though initially I will put any problems you send me into the column and invite those of you clever enough to know the answers to write in with them.

This seems like a good time to ask those of you who have solved an adventure to send me a map and brief explanation of how you did it.

By pooling our efforts I hope to be able eventually to answer most of your problems immediately. Please send in a stamped addressed envelope if you want a personal reply – I'll answer, if only to say I don't know, either!

If you want to write in with suggestions for future articles or for topics to be covered in the column, or even just for a chat, then please do so. Only please stick to adventures – arcade games are not welcome in this part of the magazine!

Above all, let me stress that this section is for you and I won't know what you want unless you write in and tell me.

If, on the other hand, you haven't tried an adventure yet, allow me to suggest that the next time you are in a bookshop you try to find a book, published by Penguin, called The Soul of a New Machine, by Tracy Kidder.

It's about the design and development of a new computer back in 1979 and describes the author's introduction to "Adventure".

If you want to find out why adventures are so compulsive then turn to page 82 of the book and start reading. Old hands will recognise the scene immediately. You can almost cut the atmosphere with a knife.

One thing I will be doing is compiling an Adventures Top Ten. So if you write in please send me your marks out of a hundred for the adventures that you have got. I have given some example marks for Level 9's Dungeon Adventure, surely one of the most complex yet to be released for the Amstrad.

Presentation	*9 out of 10*
Content	*29 out of 30*
Frustration factor	*28 out of 30*
Value for money	*30 out of 30*
TOTAL	*96 out of 100*

The Top Ten, plus our software reviews, will give you the fairest possible evaluation of the games available, though if an adventure doesn't reach the list please feel free to write in and ask for my opinion on it.

One of this year's sensations is likely to be the release of The Hobbit, from Melbourne House. I have managed to get a sneak preview of it and I suspect that my mailbag is going to contain a lot of correspondence about it.

On the subject of graphical adventures, do you prefer them, or would you rather have the memory used for more text and puzzles? I have to confess that I'm a confirmed text man.

One program I haven't yet seen is Forest at World's End. D. Brown is having problems with it and says he has got the chest and the key but cannot open the chest. Can anyone help?

I'm having problems in Emerald Isle. I've made the canoe but can not get it to float. How do you get across to the island?

David Bell has produced a help list for anyone having problems with Sorcery. If you would like a copy, please write in, enclosing an sae.

Before I go, I'd like to invite software houses to send their adventures in for review. A list of adventures and suppliers will be compiled in the near future for readers of this column.

Finally, I'd like to give you a warning. I shall not be adopting any fancy letter-transposition codes or in any way encrypting my answers. If you don't want to know the answers, then don't read the section on the right.

Well, I'd better get off now. Baggins wants me to try to read that blasted map again!

Problem corner

OVER the past few months we have had a number of pleas for help about Level 9 adventures. D. Haywood wants to know how to get out of Witts End in Colossal Adventure and where to find the jack to open the clam. *Keep going on in one direction and there isn't one, so Neptune tells me.*

L. Richfield wants to know who the figure is that waves to you and how to get back across the troll's bridge without losing a treasure. *You are looking in a mirror and you'll have to grin and bear it.*

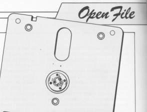
Maureen Ingle wants to know how to get past the giant in Adventure Quest and what to use the onion for. *If you play the part of David to his Goliath you'll find he'll SHOW willing. The onion isn't, FANGS ain't what they used to be!*

C. Rosenberg wants to know how to get past the Djinn and the snakes. *The Djinn is a windbag and you'll have to charm the snakes.*

M. Adams is having trouble with Dungeon Adventure. He wants to know how to get past the killer willow and the skeletons. *The willow should have its hands full with you and the skeletons will present no problem if you find something suitable to wave.*

Finally, Mrs L. Teece has written in for some advice on Fantasia Diamond. *Use a weapon on the guardian though you'll have to put some time in first and if the chest seems to reveal very little, futher examination may prove rewarding.*

See you next month.



Edit your discs sector by sector and recover accidentally deleted programs with CHES JESKE's

DEDIT

DEDIT is a disc editor which will enable you to edit any sector on any track of a disc. If you're careful you'll be able to recover programs that have been accidentally deleted.

To use it to full advantage, we have to consider how the Amstrad disc system works. There are three standard disc formats:

- *System format (CP/M).*
- *IBM format (IBM PC CP/M compatible).*
- *Data format (for future use).*

The formats have two things in common: each has 40 tracks per disc and all of the sectors are 512 bytes in length. The main differences between each format are the number of sectors on a track and the sector numbers.

Here's a list of the sector information for each format:

System format:
9 sectors per track
Sector numbers = 65 to 73.
IBM format:
8 sectors per track
Sector numbers = 1 to 8.
Data format:
9 sectors per track
Sector numbers = 193 to 201.

You don't need to worry about the format of the disc being edited because Dedit will work it out for you. Anyway, let's see how to use it.

Type in Program I and save it. On running the program the Amstrad will change to Mode 2 and draw a large box in the centre of the screen.

You will be asked for the drive number which contains the disc being edited. Your response should be 0 for drive A and 1 for drive B (assuming you have two drives).

The selected disc drive will start up for a moment. This is done to work out the format of the disc, the result of which will be displayed at the top right of the screen.

Next you are asked for the start track for editing, which must be between 0 and 39. As you'd expect you are next asked for the start sector for editing. The sector number range will vary, depending upon the disc's format. Don't worry though – the sector range will be printed with the question.

The selected sector will now be loaded from disc into memory and displayed on the screen in a hex dump format.

At the start of each line is a hexadecimal number indicating the offset of the data in the sector, that is, the number of the sector byte we've got to display from. The rest of the line shows the contents of that data byte, together with the contents of the next 15 bytes and their corresponding Ascii characters.

The screen display is not large enough to display all 512 bytes of the sector. Dedit gets round this by displaying only 256 bytes at a time. To see the second half press the 2 key on the numeric pad (f2 on the CPC 664). Pressing this once more will display the first half again.

The number next to "page" at the top of the screen indicates which half is being displayed – 0 if it's the first 256 bytes of the sector and 1 if it's the last 256 bytes. To the left of the page number is the track and sector number of the sector held in memory.

You now have several options open to you. You can either edit the sector in memory, read another section of the disc or write the sector back to the disc.

Table I lists all of the commands and control keys. There's a very useful

```
New track/sector options:

SHIFT (cursor right) next sector
SHIFT (cursor left) previous sector
SHIFT (cursor up) forward 1 track
SHIFT (cursor down) back 1 track

Editing commands:

(cursor left) right one byte
(cursor right) left one byte
(cursor down) down one line
(cursor up) up one line

Other command keys:
(Numeric pad on the CPC 464)

0 (f0 CPC664) Help page
1 (f1 CPC664) New disc parameters
2 (f2 CPC664) Page toggle
3 (f3 CPC664) Edit mode hex/ascii
4 (f4 CPC664) Write sector to disc
5 (f5 CPC664) Exit program
```

Table I

Help page which lists all of the commands. You can display it by pressing 0 on the numeric pad (f0 on the CPC664).

If at any time a disc error occurs the error message will be printed at the bottom of the screen along with "Retry, Ignore or Cancel".

It's always worth re-trying (R) in case the Amstrad's had a little hiccup. This has occurred several times in the development of Dedit and its cause is still a mystery.

Let's take a typical look at how Dedit should be used.

First of all the disc drive must be selected. In most cases 0 will be entered, indicating that drive A is being used. Next the track and sector numbers are needed. Suppose we enter 2 for the track and 65 for the sector.

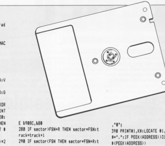
The sector will now be loaded and displayed on the screen.

If, however, the sector you selected isn't the one you want to edit you can move to another section of the disc by pressing the appropriate cursor arrow keys while holding down Shift – see Table I for a command summary.

For example, if we pressed (Shift+cursor right) the next sector would be loaded. In our case this would be sector 66 from track 2.

Once the desired sector has been located we can start editing. There are two editing modes, hex and Ascii. Toggle is selected by pressing 3 on

Program I



```
10 REM Dedit
20 REM (C) Computing with the Amstrad
30 REM by Ches Jeske
40 MODE 2:INK 0,1:INK 1,26:PAPER 0
50 MEMORY &7FFF:GOSUB 1270:' POKE MAC
HINE CODE
60 GOSUB 630:' INITIALISE
70 WINDOW SWAP 0,1
80 GOSUB 780:' DRAW SCREEN
90 CLS
100 T=0:CC=0:CR=0:WD=4:WI=24:OFF=64:V
$=D$
110 PRINT CHR$(23);CHR$(1);:TAG #WD:G
OSUB 540:POKE &908C,&80:GOTO 320
120 S$="":F=0:WHILE S$="":C=0:F=F XOR
 1:MOVE OFF+CC*WI+T*8,334-CR*16:PRINT
 #WD,CHR$(143);:WHILE S$="" AND C<20:
C=C+1:GOSUB 750:WEND:WEND:IF F=1 THEN
 MOVE OFF+CC*WI+T*8,334-CR*16:PRINT #
WD,CHR$(143);
130 ADDRESS=PEEK(&908B)+PEEK(&908C)*2
56+CC+CR*16
140 IF ASC(S$)<237 THEN 350
150 IF ASC(S$)<239 THEN GOSUB 1230:IF
 H$="N" THEN 120
160 IF ASC(S$)=237 THEN K$="write":GO
SUB 480:GOTO 120
170 IF ASC(S$)=238 THEN 1250
180 IF ASC(S$)<244 THEN 1140
190 IF ASC(S$)=246 THEN 1200
200 IF ASC(S$)=247 THEN GOSUB 920:GOS
UB 780:GOSUB 820:GOTO 330
210 CC=0:CR=0:T=0:CLS
220 IF ASC(S$)=245 THEN POKE &908C,PE
EK(&908C)XOR 1:GOTO 330
230 IF ASC(S$)=251 THEN sector=sector
+1
240 IF ASC(S$)=250 THEN sector=sector
-1
250 IF ASC(S$)=248 THEN track=track+1
260 IF ASC(S$)=249 THEN track=track-1
270 IF ASC(S$)=244 THEN GOSUB 540:POK
E &908C,&80
280 IF sector>FSN+R THEN sector=FSN:t
rack=track+1
290 IF sector<FSN THEN sector=FSN+R:t
rack=track-1
300 IF track<0 THEN track=39
310 IF track>39 THEN track=0
320 K$="read":GOSUB 480:IF PEEK(&90E3
)<>0 THEN 120
330 LOCATE #1,8,3:PRINT#1,drive:LOCAT
E #1,19,3:PRINT#1,track:LOCATE #1,31,
3:PRINT#1,sector:LOCATE #1,42,3:PRINT
#1,PEEK(&908C)AND &7F:LOCATE #1,58,3:
PRINT#1,A$
340 LOCATE #2,1,1:WINDOW SWAP 0,2:TAG
OFF #WD:CLS#3:CLS#4:CLS#5:CALL &9000:
TAG #WD:WINDOW SWAP 0,2:GOTO 120
350 IF T=1 AND WD=4 THEN POKE ADDRESS
,(PEEK(ADDRESS) AND &F0)+VAL("&"+S$)
360 IF T=0 AND WD=4 THEN POKE ADDRESS
,(PEEK(ADDRESS) AND &F)+VAL("&"+S$)*1
6
370 IF WD=5 THEN POKE ADDRESS,ASC(S$)
380 LOCATE #1,9+CC*3,5+CR:K$=HEX$(PEE
K(ADDRESS)):IF LEN(K$)=1 THEN PRINT#1
,"0";
390 PRINT#1,K$:LOCATE #1,59+CC,5+CR:K
$=".":IF PEEK(ADDRESS)>31 THEN K$=CHR
$(PEEK(ADDRESS))
400 PRINT#1,K$;
410 IF WD=4 THEN T=T XOR 1:IF T=0 THE
N CC=CC+1
420 IF WD=5 THEN CC=CC+1
430 IF CC=16 THEN CC=0:CR=CR+1
440 IF CC=-1 THEN CC=15:CR=CR-1
450 IF CR=16 THEN CR=0
460 IF CR=-1 THEN CR=15
470 GOTO 120
480 J=PEEK(&90E5):U=PEEK(&90E6)
490 POKE &90E5,track:POKE &90E4,drive
:POKE &90E6,sector:POKE &90E3,0
500 IF ASC(K$)=114 THEN CALL &90B1 EL
SE CALL &90C3
510 IF PEEK(&90E3)<>0 THEN CLS:PRINT"
Disk ";K$;" error, Track:";track;"Sec
tor:";sector;CHR$(7):track=J:sector=U
:T$="":WHILE T$="":T$=INKEY$:WEND:CLS
:RETURN
520 IF ASC(K$)=119 THEN PRINT"Sector
```

the numeric pad (f3 on CPC664).

In hex mode the contents of the sector are altered by entering hexadecimal digits (0-9 and A-F). Note that the numeric pad on the CPC464 should not be used to enter the digits 0-9. Instead use the numeric keys above "QWERTY".

The byte being edited can be changed by moving the editing cursor to the desired byte within the sector. This is achieved by pressing the appropriate cursor arrow keys.

The other edit option allows you to enter Ascii characters into the sector. For example, if you typed ABCD the bytes &41,&42,&43 and &44 would be entered starting from the editing cursor's previous position.

You can tell which editing mode you are in by looking at the editing cursor's position. If it is over the hexadecimal bytes you are in the hex mode, otherwise you're in Ascii mode.

Once you are satisfied that the sector has been successfully changed you must write the sector back to the disc. This is done by pressing 4 on the keypad (f4 on the CPC664), after which you will be prompted with "Are you sure Y/N?"

If you want to write the sector back to disc press Y. Pressing N aborts the operation and returns you to the editing mode.

Now it's time for you to discover how your Amstrad stores information on the disc. Happy hunting!

```
written successfully";CHR$(7)
530 RETURN
540 GOSUB 820
550 CLS:LOCATE 1,1:PRINT"Input Drive:
 ";:nc=1:mi=0:ma=1:GOSUB 840:drive=VA
L(w$):LOCATE #1,8,3:PRINT#1,drive
560 IF drive=0 THEN :A ELSE :B
570 FSN=PEEK(PEEK(DPH+16*DRIVE)+PEEK(
DPH+1+16*DRIVE)*256+15)
580 R=8:IF FSN=&41 THEN a$="SYSTEM" E
LSE IF FSN=&C1 THEN a$="DATA  " ELSE
a$="IBM   ":R=7
590 LOCATE #1,58,3:PRINT#1,a$
600 CLS:LOCATE 1,1:PRINT"Input Track:
 ";:nc=2:mi=0:ma=39:GOSUB 840:track=V
AL(w$):LOCATE #1,19,3:PRINT#1,track
610 CLS:LOCATE 1,1:PRINT CHR$(22);CHR
$(1);"Input Sector (";CHR$(8);FSN;CHR
$(8);"-";CHR$(8);FSN+R;CHR$(8);"): ";
CHR$(22);CHR$(0);:nc=3:mi=FSN:ma=FSN+
R:GOSUB 840:sector=VAL(w$):LOCATE #1,
31,3:PRINT#1,sector;" "
620 CLS:RETURN
630 CALL &908E:'init machine code
640 DPH=PEEK(&BE40)+PEEK(&BE41)*256+1
0:D$="0123456789ABCDEF"
650 KEY DEF 14,0,245:KEY DEF 5,0,246
660 KEY DEF 15,0,247:KEY DEF 20,0,237
670 KEY DEF 12,0,238:KEY DEF 13,0,244
680 KEY DEF 2,1,241,249,249
690 KEY DEF 0,1,240,248,248
700 KEY DEF 8,1,242,250,250
710 KEY DEF 1,1,243,251,251
720 POKE &908B,0:POKE &908C,&80
730 WINDOW #1,1,80,22,25:WINDOW #2,2,
80,5,25:WINDOW #3,2,6,5,20:WINDOW #4,
9,56,5,20:WINDOW #5,59,75,5,20
740 RETURN
750 S$=INKEY$:IF (S$<CHR$(237) OR S$>
CHR$(252)) AND INSTR(""+V$,UPPER$(S$)
)<2 AND LEN(V$)>0 THEN S$=""
760 IF V$<>"" THEN S$=UPPER$(S$)
770 RETURN
780 CLS #1:PEN #1,1:LOCATE #1,1,1:PRI
NT#1,"Amstrad Disc Sector Editor."
790 MOVE 0,380:DRAWR 216,0
800 MOVE 0,72:DRAWR 0,272:DRAWR 600,0
:DRAWR 0,-272:DRAWR -600,0:MOVER 48,0
:DRAWR 0,272:MOVER 405,0:DRAWR 0,-272
810 RETURN
820 LOCATE #1,1,3:PRINT#1,"Drive:
 Track:     Sector:        page:
  Format:        ":RETURN
830 nc=2:mi=2:ma=10
840 w$=""
850 s$="":WHILE s$="" OR LEN(w$)=nc A
ND s$<>CHR$(127) AND s$<>CHR$(13) OR
s$=CHR$(127) AND w$="":s$=INKEY$:WEND
860 IF s$=CHR$(13) THEN 900
870 IF s$=CHR$(127) THEN w$=LEFT$(w$,
LEN(w$)-1):PRINT CHR$(8);CHR$(16);
880 IF s$<>CHR$(127) THEN PRINT s$;:w
$=w$+s$
890 GOTO 850
900 IF VAL(w$)<mi OR VAL(w$)>ma THEN
FOR a=1 TO LEN(w$):PRINT CHR$(8);CHR$
(16);:NEXT:PRINT CHR$(7);:GOTO 840
910 RETURN
920 WINDOW SWAP 0,1:CLS
930 LOCATE 31,1:PRINT CHR$(24);" *COM
MAND MENU* ";CHR$(24)
940 LOCATE 1,3:PRINT CHR$(24);" Movin
g to new track or sector:";CHR$(24)
950 PRINT"  <SHIFT> "+CHR$(243)+" ...
 move forward one sector."
960 PRINT"  <SHIFT> "+CHR$(242)+" ...
 move back one sector."
970 PRINT"  <SHIFT> "+CHR$(240)+" ...
 move forward one track."
980 PRINT"  <SHIFT> "+CHR$(241)+" ...
 move back one track."
990 LOCATE 1,9:PRINT CHR$(24);" Editi
ng commands:";CHR$(24)
1000 PRINT"  "+CHR$(243)+" ... move r
ight one byte."
1010 PRINT"  "+CHR$(242)+" ... move l
eft one byte."
1020 PRINT"  "+CHR$(241)+" ... move d
own one line."
1030 PRINT"  "+CHR$(240)+" ... move u
p one line."
1040 LOCATE 1,15:PRINT CHR$(24);" Oth
er command keys (numeric KEY pad):";C
HR$(24)
1050 PRINT"  f0 ... Help."
1060 PRINT"  f1 ... Select new disk p
arameters."
1070 PRINT"  f2 ... Page toggle, (tog
gles display between bytes"
1080 PRINT"                        0-2
55 & 256-511 of a sector)."
1090 PRINT"  f3 ... Toggles between H
EX & ASCII for editing."
1100 PRINT"  f4 ... Writes sector to
disk."
1110 PRINT"  f5 ... Exit program."
1120 LOCATE 30,24:PRINT CHR$(24);" PR
ESS SPACE TO EXIT ";CHR$(24):s$="":WH
ILE s$<>" ":s$=INKEY$:WEND
1130 CLS:WINDOW SWAP 0,1:RETURN
1140 IF ASC(S$)=243 THEN 410
1150 IF ASC(S$)=242 AND WD=4 THEN T=T
 XOR 1:IF T=1 THEN CC=CC-1
1160 IF ASC(S$)=242 AND WD=4 THEN 430
1170 IF ASC(S$)=242 AND WD=5 THEN CC=
CC-1:GOTO 430
1180 IF ASC(S$)=240 THEN CR=CR-1:GOTO
 450
1190 CR=CR+1:GOTO 450
1200 TAGOFF #WD:WD=WD XOR 1:TAG #WD:T
=0
1210 IF WD=4 THEN WI=24:OFF=64:V$=D$
ELSE WI=8:OFF=464:V$=""
1220 GOTO 120
1230 IF ASC(S$)=237 THEN K$="Save sec
tor" ELSE K$="Quit program"
```

```
1240 LOCATE 1,1:PRINT K$;" ARE YOU SU
RE (Y/N)?";CHR$(7);:H$="":WHILE INSTR
("#YN",H$)<2:H$=UPPER$(INKEY$):WEND:C
LS:RETURN
1250 TAGOFF #WD:PRINT CHR$(23);CHR$(0
):MODE 1:CALL &BB00
1260 END
1270 IF PEEK(&9000)=&DD THEN RETURN
1280 RESTORE 1360
1290 CHECKSUM=0
1300 FOR G=&9000 TO &90DC
1310 READ B$:POKE G,VAL("&"+B$)
1320 CHECKSUM=CHECKSUM+VAL("&"+B$)
1330 NEXT
1340 IF CHECKSUM=30232 THEN RETURN
1350 PRINT"M/C DATA ERROR";CHR$(7)
1360 DATA DD,2A,8B,90,DD,E5,C1,78,E6,
7F
1370 DATA 32,8D,90,CD,62,90,79,32,8D,
90
1380 DATA CD,62,90,DD,E5,06,03,CD,81,
90
1390 DATA 06,10,DD,7E,00,32,8D,90,CD,
62
1400 DATA 90,3E,20,CD,5A,BB,DD,23,10,
EE
1410 DATA 06,02,CD,81,90,DD,E1,06,10,
DD
1420 DATA 7E,00,FE,20,D2,45,90,3E,2E,
C5
1430 DATA CD,5D,BB,C1,DD,23,10,ED,3E,
0D
1440 DATA CD,5A,BB,3E,0A,CD,5A,BB,DD,
E5
1450 DATA C1,3E,00,B9,C2,04,90,C9,21,
8D
1460 DATA 90,ED,6F,CD,70,90,ED,6F,CD,
70
1470 DATA 90,C9,E6,0F,C6,30,FE,3A,FA,
7B
1480 DATA 90,C6,07,C5,CD,5A,BB,C1,C9,
C5
1490 DATA 3E,09,CD,5A,BB,C1,10,F7,C9,
00
1500 DATA 00,00,DD,21,DD,90,21,DB,90,
CD
1510 DATA D4,BC,DD,75,00,DD,74,01,DD,
71
1520 DATA 02,21,DC,90,CD,D4,BC,DD,75,
03
1530 DATA DD,74,04,DD,71,05,C9,21,00,
80
1540 DATA ED,5B,E4,90,3A,E6,90,4F,DF,
DD
1550 DATA 90,D2,D5,90,C9,21,00,80,ED,
5B
1560 DATA E4,90,3A,E6,90,4F,DF,E0,90,
D2
1570 DATA D5,90,C9,3E,FF,32,E3,90,C9,
84
1580 DATA 85
```

Give your fingers a rest . . .

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

See our special offer on Page 77.

Extensions: Here's more commands for you

AFTER reading the excellent article in your May edition about RSX extensions, I have written three very useful commands.

The new commands are: ¦TURBO, ¦COPY and ¦SHOW.

The first command allows the user to save programs at nearly four times the usual speed just by typing ¦TURBO and then saving in the normal way.

The last two commands are connected together and allow the user to store a copy of the screen in memory so that it can be used later.

Potential? Storing a graphics picture in a drawing program.

They are ¦COPY (copies the screen) and ¦SHOW – show the copied screen.

These additional commands should prove very useful to other Amstrad owners. – **Duncan Miles, Rochester, Kent.**

■ For some peculiar reason the program generates an error report when run. However, the new commands still work.

```
5 REM RSX Commands
10 REM By D.E.Miles 1985
20 REM
30 MEMORY &3999
40 FOR f=40000 TO 40071:REA
D a$:POKE f,VAL("&"+a$):NEX
T
50 CALL 40000
60 DATA 01,4A,9C,21,84,9C,C
D,D1,BC,C9,55,9C,C3,63,9C,C
3,6C,9C
70 DATA C3,78,9C,54,55,52,4
2,CF,43,4F,50,D9,53,48,4F,D
7,00,21
80 DATA 5F,00,3E,05,CD,68,B
C,C9,21,00,C0,11,00,40,01,0
0,40,ED
90 DATA B0,C9,21,00,40,11,0
0,C0,01,00,40,ED,B0,C9,00,0
0,00,00
100 REM
110 REM The new commands ar
e :-
120 REM  ¦TURBO  fast savin
g
130 REM  ¦COPY  copy screen
140 REM  ¦SHOW  show copied
 screen
```

You can't dodge it

THE problem I have experienced seems to have arisen from a slipping tape drive as there was a squealing noise from the tape deck while saving a program which has resulted in the message "Read error b" when trying to run the program.

Can you tell me if there is any way of retrieving the readable part of the data? I fear your answer will be no, but I must ask as I do not relish retyping what is a rather lengthy program. – **P.A. Sotheron, West Drayton, Middlesex.**

■ Unfortunately we can't see any way of doing this. The problem might be a fault with the deck (it might just need cleaning) or even the tape itself. Either way, you're going to have to type the program in again.

Tip for Roland . . .

I'D love to see your reviews have screens from the games and a scoring system.

And I'm sure that others would like tips on how to do games and maps of games, such as Jet Set Willy and Final Frontier.

Otherwise your magazine is really wonderful, as our maths master says.

One tip for Roland in the Caves – Press the Down keys as soon as you appear on the screen, and the computer thinks you've got out and adjusts your score. – **H.W.F. Breach, Henbury, Bristol.**

■ As you can see in this issue we've included screen shots of some of our program listings running, but not reviewed software. The problem is freezing the game long enough to take a picture.

The trouble with points systems is that they look quite "scientific" but are really subjective and prone to error. As for tips, isn't that cheating?

Print-out problem

I HAVE been the proud owner of a CPC464 since November. Since then I have tried my hardest to make some sense of the programming language Basic.

Before I purchased my Amstrad I had never used a computer, and, as you can imagine, I was more than a little confused by all the jargon and backtalk which seems to be associated with them.

At the beginning of February I bought an Epson RX-80 F/T+ printer for my Amstrad, as I use my computer as a type of electronic names-and-addresses pad and I thought it would be easier if I could get the occasional printout.

My problem is that I cannot get my computer to print out information on the screen as well as on the printer.

I noticed that a command for the computer to print out a hard copy is different from the command to simply print up on the monitor.

At the moment, if I want to look through the information contained in the program then I have to either LIST the program OR alter all the print #8, commands for print commands. HELP!

Why does my computer do this? Is it just my inexperience with these contraptions that makes this error occur, or is there something wrong with my computer?

Also, could you please tell me what the % (per cent) sign stands for in the character generator listing in your issue dated May, 1985? – **Richard F. Jones, Macclesfield, Cheshire.**

■ All print commands send the information to a particular stream, the default being 0, the screen. All you need to do is set up a variable to hold the stream number. To print on the screen set it to 0, and to print it on the printer set it to 8.

The % after a variable forces it to be an integer – that is, a whole number.

Double trouble

COULD you please tell me two things? The first is – How do I achieve more than two colours in one text square in Basic?

The second is, say you were moving a character across the screen using TAG, how would you tell the Amstrad that it has made contact with, say, your "man"?

All I have achieved so far is when the two top-left pixels meet, then the computer says ▶

you have hit something.

"PS: I have found 103 screens out of the 130 in Jet Set Willy. Could any readers tell me how to get past Decapitation? – **Graham Stride (aged 14), Ottershaw, Surrey.**

■ To print a multicoloured character you need to define a character for each colour and print them all at the graphics cursor using TAG, at the same coordinates. You'll also need to set the XOR or OR graphics option.

The easiest method of collision detection is using TEST to see what is at the next pixel. As for Jet Set Willy, the only advice we can give is to keep your head! Can our readers help?

No LOCK switch

PLEASE could you tell me of a way of turning Caps Lock on or off from within a Basic program? This would be a great help as it would stop me keep having to use UPPER$ and LOWER$.

PS: Will you be doing a beginner's course on using the cassette for filing information? – **Neil Carter (aged 15), Poplar, London.**

■ There isn't a Basic command to turn Caps Lock on and off, unfortunately. We haven't planned a cassette filing course – however if there's enough demand for it, it'll be done. As it is this month's Analysis should help.

A nice bonus . . .

HAVING read the letters in your magazine over the last few issues, I would like to add a few comments of my own.

Firstly, on the methods of setting the graphics pen and paper from Basic. The most direct method is to POKE the correct RAM addresses – these are &B339 for the paper and &B338 for the pen.

However, it is not simply a question of POKEing with the required ink, as the ink should be encoded before being placed in RAM.

The correct number depends on the Mode, and can be determined by drawing a line in the required colour and PEEKing &B338 at the programming stage.

If other numbers are POKEd into these locations, dotted lines and striped backgrounds can be obtained with simple CLG and DRAW commands. This is a nice bonus, and would require a whole article to explore fully!

Secondly, the comment that programs on disc cannot be MERGEd. In fact MERGEing can be performed normally provided the program on disc has been SAVEd in Ascii format – that is, using SAVE "Progname",A.

If it has been SAVEd in normal tokenised form, then the method to use would be to save the current program in Ascii format, LOAD the second program and then MERGE the original program.

The other comment in the same letter concerned random access files on the disc. This would not need CP/M, but a relatively simple machine code program. For this, however, the Disc Drive Interface Manual would be essential. – **Dr S. Peppiatt, Chelmsford, Essex.**

■ Thanks for the programming tips.

We ring Da Bells!

HAVING just typed in Da Bells, from the June issue, I had to write to congratulate you on the best game yet.

As a novice of only two months, I've typed in seven programs now from your magazine and found them all good, including Smiley, Star Fleet, Mad Adder, Egg Blitz, Kingdom of Craal and Tron Cycles. And I don't make as many mistakes now. – **L. James, Lymm, Cheshire.**

■ We're glad to hear that you like our games, and that your typing has improved.

A call for calls

I HAVE owned my CPC464 since November, 1984. I have gained a very good understanding of Amstrad Basic and have recently gone on to learning machine code.

I found the article on RSXs excellent and am now writing my own.

To help me in machine code I bought a book which cost £7.95. The only useful information I got out of it was the list of mnemonics and opcodes which helped me write an assembler-disassembler program!

I can now write my own machine code programs on my own assembler, which is written in Basic and does exactly as I want it to. My only limitation is the number of firmware calls I know.

I only know about 30 and

I'm afraid of crashing!

I AM nearly sixty years of age and severely disabled.

I purchased my Amstrad at the end of last year, and must admit I was absolutely lost until the appearance on the scene of your excellent magazine.

I have typed in most of your games, and I am delighted to report that everything works, much to my astonishment. I make this remark because I also took another Amstrad magazine, which shall be nameless, and have yet to get any of its listings to run.

One of my main queries is this: You stated in one of your earlier issues that no matter what you type into your computer you cannot damage it. Yet I am constantly reading "You may crash the system".

The word "crash" frightens me, because I would hate to damage my computer memory.

I have enclosed a cheque for this month's tape. I usually like typing in my own, but this month's includes a program called Monitor.

Before I use this program, which is extremely well explained by Kevin Edwards, he says in the bottom line, second column: "The machine could crash". Also, in the penultimate column: "Be very careful with the editing command as it is a very powerful and dangerous feature if abused". Two statements which frighten the life out of me.

Is it true when you disconnect the computer it automatically resets to its original condition when you switch it on again? By answering this I think you will answer my queries and solve my fears.

Thank you for the many hours of enjoyment you, and my computer, have given me. – **R. McClenaghan, Kirkby, Liverpool.**

■ Be reassured, CRASHing the system does it no harm. All it means is that the micro becomes morbidly introspective and refuses to accept your commands or run the program. This can happen for many reasons (some people seem to be able to crash a system just by looking at it).

When you start playing around with machine code, crashes become more likely hence the warning on our Kev's monitor. However, while it's annoying and inconvenient, its not harmful. Just switch the Amstrad off and on again, or, less violently, press Ctrl, Shift and Escape to get back to the original conditions.

some of them are useful and others almost useless.

It would be a good idea to list five or six extra calls each month in a chart which gives easy reference.

Now for my question: Is there a call or specific buffer in memory which controls the save speed? If so, can this be changed without delving too deep into machine code?

Can you please give me some information as it would be very useful to use this in RSX commands? – **Colin Newcombe (14), Sudbury, Suffolk.**

■ Everything you need to know about the operating system when writing in machine code is contained in The Complete CPC464 Operating System Firmware Specification from Amsoft.

The routine at &BC68 sets the cassette write speed.

Package may help

I WORK in a small office which owns four bed-and-breakfast hotels. Having a CPC464, we have decided to put it all on computer.

Could you please tell me where I will be able to get a program for a hotel register? – **M.J. Reid, London.**

■ No one knows of a program specifically written to deal with hotel registration, though you could probably use a database package such as Mini Office to good effect.

A side effect

YOUR solution to the problem of delays in allocating a tape buffer (June issue) – that is, open a dummy file early in the program, then reserve the buffer space using MEMORY – has the unfortunate side effect of causing an "Improper argument" crash whenever SYMBOL AFTER is used subsequently.

The only way to resolve this is to reset the machine, since, as you point out, the buffer is permanent! Any clues?

Meanwhile, a tip on tracking down all lines in a Basic program which GOTO or GOSUB a particular line.

First, RENUM the program. Then edit the line (say 310), so that it is now line 311, and delete line 310. Then RENUM again.

Basic warns you of all the lines referring to non-existent line 310, and line 311 is magically changed back to 310 again. – **Jim Treadwell, Cheadle Hulme, Cheshire.**

■ Sadly, we're clueless concerning your first point. It's just a feature of the Basic. Thanks for the tip which could help sort out some tangled programs.

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

Listing wanted

CAN you tell me whether anyone has produced backgammon as software for the CPC464?

If not, would it be possible for Computing with the Amstrad *to publish a listing?* – **R.A.R. Giddis, Epson, Surrey.**

■ Deep Thought software has produced a backgammon for the CPC464. As for a listing, when someone sends one in and it's good enough, we'll publish it!

Make a note

IT may interest your readers to know that the CPC464 executes all sounds one octave lower than the note corresponding to the tone period setting given in Appendix VII of the User Manual.

International A (440Hz) is in fact generated at a tone period of 142 and not 284 and all tone periods require corresponding correction (dividing by 2).

The correct relationship between frequency and tone period should be:

Frequency (in Hz) = 62500/tone period and not as given on page 2 of chapter 6 of the manual and repeated on page 3 of Appendix VII.

Incidentally, the frequency formula on this page also requires correction – the term (N-10) should be substituted for (10-N).

The foregoing observations have been pursued with Amsoft, who advise that future copiues of the manual will be corrected. – **A.P. Pearce, Seaford, Sussex.**

■ Thanks for the information. The frequency formula in the original manual is wrong. However the correct formula with the expression (N-10) is given in the firmware guide and the guide to the CPC664.

Sign of the power

IN your May issue of Computing with the Amstrad the program called "Character Generator" has something in the listing I have not seen before:

```
250 IF by%(t%)>=2^g% THEN
260
```

Would you please explain the ^ between the 2 and g%? – **A.P. Eaton, Lichfield, Staffs.**

■ The ^ symbol means "to the power of". However, on the Amstrad this is given by the ↑ symbol which shares a key with the £ sign. Most printers list ↑ as ^, however, so where you see ^, use ↑ and you'll have no problems.

Fishing for boots

WITH reference to Mr Smith's letter, in your June issue of Computing with the Amstrad, requesting hints for Mission One – Project Volcano, may I suggest fishing at the shaft before reaching the grill?

He will catch a pair of rubber boots which he must put on. – **Mrs D. Primrose, Newcastle upon Tyne.**

Over the chasm

AFTER reading your review of Forest at World's End, I ordered a copy of it which was sent to me by a relative living in the UK.

Unfortunately my knowledge of the adventure jargon is scarce and I can't get through the great chasm and the command "Help" does not give any ideas.

It is very frustrating not to

▶

get further on after a month.

Can you help me?

Congratulations on the magazine. – **Gustavo A. Cimorra, Zaragoza, Spain.**

□ It's a while since we played it, but the idea is that you drop a log across the chasm.

Good luck!

Silly sums

IS it me or is it the CPC464 that can't count? Look at these calculations:

```
10 j = 6.285
20 k = 5.468
30 PRINT "j-k = "; j-k
40 IF j-k = 0.817 THEN GOTO
 60 ELSE 50
50 PRINT "Incorrect": GOTO
70
60 PRINT "Correct"
70 x = 62.85
80 y = 54.68
90 PRINT "x-y = ";x-y
100 IF x-y = 8.17 THEN GOTO
 120 ELSE 110
110 PRINT "Incorrect": GOTO
 130
120 PRINT "Correct"
130 a = 6285
140 b = 5468
150 PRINT "a-b = "; a-b
160 IF a-b = 817 THEN GOTO
180 ELSE 170
170 PRINT "Incorrect"
180 PRINT "Correct"
```

Running the program will give you:

```
    j-k =  0.817
    Incorrect
    x-y =  8.16999999
    Incorrect
    a-b =  817
    Correct
```

Does it only calculate correctly when using integers? Or perhaps it is necessary to always use the ROUND function. – **John Todd, Bangor, N. Ireland.**

■ When dealing with decimals computers are forced to work within certain bounds of accuracy.

The way the Amstrad stores these floating point numbers, as they're known, inevitably introduces slight errors which can cause silly happenings in sums.

Once you're aware of this and not expecting absolute precision, it isn't as much of a problem as it first might seem.

Not too basic, please

I FULLY understand that you must cater for the first time user of the CPC464 but please not do so at the expense of experienced computer users.

As Amstrad has found that the majority of users have owned at least one other computer before upgrading to the CPC464, I feel that the majority of CPC464 users are being let down a little by the content being nearly 100 per cent for the absolute beginner.

Please note below a program routine to draw circles on the CPC464, Pretty fast isn't it?

Another point which may be of interest to CPC464 users is that due to the relative newness of the machine there aren't a lot of routines published in the press as yet specifically for the CPC but I have found that routines for the BBC Micro, especially regarding drawing and related graphic routines, do transfer quite well.

Programs written in a standard Microsoft Basic for other machines will also transfer easily, and in most cases the results, when tailored to the Amstrad, are better than the original.

I have a few comments to various points raised in previous Postbag letters.

As a radio ham might I suggest that Mr O'Neil (March 1985) removes the transformer in the GT64 and separately encases it then the hum bars will disappear.

And as regards the RFI from the CPC464 there's not a lot he can do except use a commercial screening paint and coat the plastic cases with it (heated rear screen repair paint will suffice).

To convert easily from decimal to binary why not use the BIN$ function like this:

```
10 d=(decimal number)
20 b$=BIN$(d,8)
30 PRINT b$
```

The same format works for HEX$. Note the 8 in the BIN$ function is the amount of digits to return the result in. – **Steve Potter, Brinsley, Notts.**

■ Thanks for the contributions which no doubt will satisfy the more 'experienced computer users'.

```
10 RAD:REM set radians mode
20 radius=100:REM give vari
able radius a value 100
30 x%=320:y%=200:REM set x+
y to centre of screen
40 cosine=COS(2*PI/32):sine
=SIN(2*PI/32)
50 xy=0:ORIGIN x%,y%
60 MOVER radius,xy
70 t%=32:WHILE t%>-1:t%=t%-
1
80 xdummy=radius*cosine-xy*
sine
90 ydummy=xy*cosine+radius*
sine
100 radius=xdummy:xy=ydummy
110 DRAW radius,xy,1
120 WEND
130 STOP:REM done it app. .
7 seconds
```

Hobbit

I WONDER if you can help me regarding a game called "Hobbit" based on the book by Tolkien. Can you tell me if it has been released for the Amstrad? – **Miss S.M. Hughes, Plaistow, London.**

■ Melbourne House has done an excellent conversion of The Hobbit for the Amstrad. Mike Bibby is battling his way through it at the moment.

Problem with bells

WHILE looking through my program Da Bells (June issue) I found something which appears to be a bit of a problem.

As you are probably aware, I have used a string to hold the graphic positions of Mr Humpy.

Every time their positions are changed I changed that string to the graphic character required.

The computer however makes a copy into memory every time the string is changed and therefore after long periods – or even short as is the case of Da Bells – the memory fills up.

This effect can be avoided by executing a garbage collection after the computer has returned from the movement routine. This takes the form:

```
x=FRE("")
```

I don't know if you already know this, but I just want to make sure that the main culprit for the memory loss is found.

Not having the Amstrad very long, can you please tell me if there is a better way of avoiding this problem, as it is getting to be a bit frustrating? – **Aramello Chapman, Boston, Lincs.**

■ A better method is to define a string up to the maximum length, at the start of the program as follows:

```
a$=STRING$(255,"*")
```

There is no longer any need to make copies of the string as it cannot get any longer.

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

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

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

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

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

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

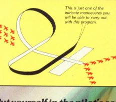
The program comes with a detailed flight handbook that will soon give you the confidence to take YOUR place alongside the ace pilots of the Red Arrows, even if you've never flown before!

Be a VIP visitor with the Red Arrows!

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

Now on sale at:

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



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

ORDER FORM

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

I wish to pay by:

□ Access/Mastercard/Eurocard No.

□ Barclaycard/Visa No.

□ Cheque/PO made payable to Database Publications Ltd.

Name ______________________

Address ______________________

____

Signed ______________________

Send to: Database Software, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

(No stamp needed if posted in UK) *Please allow 28 days for delivery*

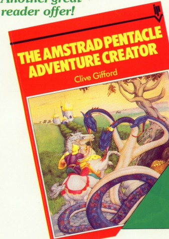
YOU CAN ALSO ORDER BY PHONE: **061-480 0173** *(24 hours)* *Don't forget to quote your credit card number and full address.*

CA8

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

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

Another great reader offer!



This is the book that shows you how easy it is to create adventures on your Amstrad! Just type in the program, then follow the simple instructions to create as many exciting, hard-to-solve adventures as you like.

All you have to do is enter the data of your choice and the Pentacle program does ALL the hard work, building up an adventure you can save – and even sell!

As a bonus there's also a full set of data showing Pentacle in use with an adventure set in bygone days – Castle of Doom. Then a change of pace is provided with a full graphical adventure, The Search for Speck.

Now adventuring will never be the same again!

FREE!

The book that will really set you adventuring with your Amstrad will be sent to every subscription ordered on the form on the right.

Worth £2.99!

GIVE YOUR FINGERS A REST!

JANUARY: ***Smiley:*** *Guide Smiley round the labyrinth avoiding Grumpies.* **Code:** *Be a mastermind.* **Binary:** *Baffled by binary bits? Let our utility help you out.* **Dancer:** *Simple but fun. This is a lovely mover.* **Trapper:** *Pen the monsters of the maze.* **Scroller:** *A slick sideways scrolling routine.* **Letter Litter:** *Learn the keyboard.*

FEBRUARY: ***Digger:*** *Search for crystals and avoid the aliens.* **Simon:** *Our version of this classic game.* **Kingdom of Craal:** *A devious fantasy.* **Text Editor:** *A powerful word processor.* **Reaction Timer:** *Test out your reactions on this simple little program.* **Origins:** *Changing the graphics origin produces dramatic effects.*

MARCH: ***Starfleet:*** *Can you qualify as a starfleet pilot?* **Swamp:** *Join the battle against giant frogs.* **Parachute:** *Parachute to earth in this educational program.* **Dump:** *Screen dump utility for Modes 0 and 1.* **Hexer:** *A hexadecimal loader.* **PLUS . . . 3D Four in a Row.**

APRIL: ***Mad Adder:*** *Appease our aggravated adder.* **Egg Blitz:** *Eggs, boy scouts and a stunt pilot make up the fun.* **Simple Sprites:** *Create sprite-like effects.* **Caesar Cipher:** *Crack the code.* **Pilot:** *An easy to learn CAL language.* **Spelling:** *Learn to spell with this educational game.* **Aleatoire:** *The odd ways mariners make decisions.* **Palindrome Tester:** *Slick string handling techniques.*

MAY: ***Tronn Cycles:*** *Enter the Maze and pit your wits in this two player arcade classic.* **Castle of Fear:** *Fantasy is the order of the day in this baffling concoction.* **Mouse:** *Can you trap the furry fellow in this simple but addictive game?* **Cedric:** *Fun with shape recognition in this novel interpretation of pelmanism.* **Character Maker:** *Define your own characters with our superb*

Computing with the AMSTRAD

MAIL ORDER OFFERS ORDER FORM

All prices include postage, packing and VAT and are valid to August 31

Please enter your requirements by ticking boxes £ p

Annual subscription

UK £12	6001	□	
Eire £13 (IR £16)	6002	□	
Overseas (Surface) £20	6003	□	
Overseas (Airmail) £40	6004	□	
Please send me my free Amstrad Adventure Book	6052	□	

Commence with ________ issue TOTAL ____

Tapes

£3.75 (UK & Overseas)

January (Smiley)	6020	□	
February (Digger)	6021	□	
March (Starfleet)	6022	□	
April (Mad Adder)	6023	□	
May (Tronn Cycles)	6024	□	
June (Da Bells)	6025	□	
July (Missile Command)	6026	□	
August (Ah Bod!)	6027	□	

TOTAL ____

3" Discs

£6.75 (UK & Overseas)

January (Smiley)	6060	□	
February (Digger)	6061	□	
March (Starfleet)	6062	□	
April (Mad Adder)	6063	□	
May (Tronn Cycles)	6064	□	
June (Da Bells)	6065	□	
July (Missile Command)	6066	□	
August (Ah Bod!)	6067	□	

TOTAL ____

Back issues

£1.25 UK
£1.50 Overseas (Surface)

January	6006	□
February	6007	□
March	6008	□
April	6009	□
May	6010	□
June	6011	□
July	6012	□

TOTAL ____

Mini Office

Tape	£5.95	6035	□
3" Disc	£9.95	6036	□

TOTAL ____

dk'tronics speech synthesiser

£34.95 (UK) 6040 □

TOTAL ____

Cassette tape and disc annual subscription

Tape £40 (UK & Overseas) 6005 □
3" Disc £70 (UK & Overseas) 6059 □
Commence with ________ (state month)

TOTAL ____

>>>

TOTAL ____

How to keep your collection complete

Bound in rich burgandy pvc and bearing the Computing with the Amstrad logo, this handsome binder will hold a year's supply of the magazines firmly secured in place with metal rods. **£3.95**

Dust Cover Only £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.

NEW!

dk'tronics Light Pen

Full details on Page 33

*All the program listings from each month's issue are available for only **£3.75** on cassette and **£6.75** on disc.*

*utility. **RSX:** New commands for your micro. **Cards:** Baffle your friends with an apparently simple computer card trick. **Trees:** A clever routine to create 'trees'.*

***JUNE: Easydraw:** Hours of fun and entertainment. This versatile graphics utility allows you to create and manipulate your own pictures, saving them to tape or disc. **Da Bells:** Hilarious arcade action for the whole family with the obligatory Hunchback on the ramparts. Can you rescue Esmerelda? **AMmon:** A superb machine code monitor to complement our Z80 series. **Menu:** Create colourful menus with this intelligent disc utility. **Nim:** Baffle your friends with Alleatoire's simple routine to play the famous mathematical game. **Quads:** A clever routine to display expanding quadrilaterals.*

***JULY: Missile Command:** Take command of your laser base and save earth from the invading hordes. **Jumbo's Wordgame:** Have hours of family fun with our Amstrad version of Odd One Out. **Biorhythms:** Find out if you're going to have "one of those days" with this superb utility. **Rase:** Add power to your programming with this superb machine code assembler. **Compacter:** Reduce the memory used while saving a screen with this handy utility. Load it back in half the time.*

***AUGUST: Ah Bod:** Guide Bod to collect the eggs and avoid the evil Stompers. **Dedit:** A superb disc utility for sector editing and recovering lost programs. **Polygons:** Create and display superb shapes rotating about their horizontal and vertical axes. **Events:** Demonstrations of interrupt routines. **Password:** A simple routine to assist you to generate your own password. **Graphics:** A superb but easy to use character generator and demonstration programs. **M/code graphics:** Routines to examine the screen and pixel patterns.*

Binder

£3.95 UK
£5.00 Overseas
6038 □

TOTAL ____

Dust Cover

£3.95 (UK) 6037 □

TOTAL ____

dk'tronics Light Pen

£24.95 6057 □

TOTAL ____

Amstrad Pentacle Adventure Creator

£2.99 6052 □

TOTAL ____

Red Arrows

Tape	£8.95	6050	□
Disc	£12.95	6051	□

TOTAL ____

Classic Games on the Amstrad Vol. 1

Tape	£5.95	6048	□
Disc	£9.95	6049	□

TOTAL ____

Overseas postage: add 50p per item unless otherwise indicated. ____

Payment: please indicate method (✓) **TOTAL** ____

□ Access/Mastercharge/Eurocard
Card No. ____ ____ ____ ____

□ Barclaycard/Visa
Card No. ____ ____ ____ ____

□ Cheque/PO made payable to Database Publications Ltd.

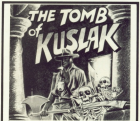
Name ____________ Signed ____________
Address ____________
____

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

YOU CAN ALSO ORDER BY PHONE: 061-480 0171

Don't forget to quote your credit card number and full address CA 8

ADVERTISERS INDEX



Amstrad CPC464

Speech Synthesizer

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

Stereo Output

To utilise the Amstrad stereo output on the back of the computer, the interface has a built in stereo amplifier, this gives all sound output a totally new dimension and greatly improves the sound quality and volume over the computer's internal speaker. Any sound that previously came out of the mono speaker will now be sent out via the interface in stereo. All programs that use the sound in anyway (i.e. commercial software) will now output through the interface, which is fitted with volume and balance controls .

Speech Synthesis

The Amstrad speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an almost unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chip's internal rom.

Text to Speech

Although there are only 26 letters in the alphabet, letters have a totally different sound when used in different words. For example, The "a" in 'Hay' is much longer and softer than in 'Hat'. When you speak you automatically make adjustments because you know just how a word should sound Not quite so easy with a computer.

The machine code software is mainly developed to this mode of operation. 3.5K is used for tables which contain the rules & exceptions to the rules of the English Language.

e.g. I before E except after C) This therefore allows the user to enter words to be spoken in normal English.

Speakers

Supplied with the Speech Synthesizer are two high quality 4" speakers these have been designed to compliment theAmstrad Computer. They are fitted with 1 metre of cable and can be positioned for the best stereo effect. The synthesizer interface fits neatly on to the rear of the computer. It has a through connector to enable other interfaces (e.g. Disc Drive) to connect to the rear of the synthesizer for ease of expansion Please send S.A.E. for a copy of the instruction manual which will give full and comprehensive details.

New Basic Commands

There are 8 new Basic Commands which control all the functions of the interface. Making the Synthesizer very easy to use. You can even control the speed at which it will talk to you. Or use the synthesizer to create sound effects like a fourth sound channel.

```
10 PRINT " 'AMSTRAD' "
```

The above is an example of the Syntax for entering speech into the computer and shows how simple it is to use.

The instruction book gives comprehensive details and examples of how to use the interface both from machine code and basic.

How to Order

The Amstrad Speech Synthesizer costs only £39.95. You can obtain your synthesizer through any good computer store or by completing the order form and returning it to:
dk'tronics Limited, Shire Hill, Saffron Walden, Essex. OR
by telephone quoting your barclaycard or access number. Orders normally despatched within 24 hours.

dk'tronics

Saffron Walden, Essex CB11 3AQ
Tel: (0799) 26350 10 lines

Please rush me

.....[QTY] Amstrad Speech Synthesizer at £39.95 +£1.25 p&p
I enclose cheque/PO/Cash for Total £.........................
or debit my Access/Barclaycard No.

□□□□□□□□□□□□□□□□

Signature...
Name...
Address...

the only choice

Kuma
AMSTRAD CPC464
software

No.1 BEST SELLER

North Sea Bullion

Galaxia

Fruity Frank

Star Avenger

Shadow of the Bear

Home Budget

Database

Zen Assembler

Artwork

Bridge

BOOKS

No.1 BEST SELLER

The Amstrad CPC 464 Explored
by John Braga

This superb book is designed to let every CPC 464 user, at whatever level, get the most from his computer. After an introductory section on the special Basic features, the book looks in depth at the excellent sound and graphic facilities.

NEW

ZEN and the Amstrad CPC 464
by Ian R. Sinclair

This book covers the CPC 464 itself, a detailed guide to Zen, an introduction to the Z-80, ROM, and RAM, subroutines, screen and keyboard, cassette I/O and how to plan a program. The text is littered with illustrations, diagrams and helpful program examples.

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC 464 Micro-computer.

Now available from selected branches of Co-op, Granada, LASKYS and John Menzies

Kuma Computers Ltd., Unit 12, Horseshoe Park,
Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name ....
Address ....
....
.... Phone......................

I own an Amstrad CPC 464 computer □

CWA8

Trade Enquiries Phone 07357-4335

Visitors wishing to call at our Pangbourne Manufacturing and Distribution Centre are advised to phone 07357-4335 first for an early appointment.