AN ARGUS SPECIALIST PUBLICATION

SEPTEMBER 1985 £1.00

COMPUTING Today

COMPUTERS TAKE TO THE TRACK
how they got into Formula racing

PROGRAMMING FOR SPEED
optimising disk based systems

LANGUAGES
QL Pascal from Metacomco

BLOCKCODE
novel structures as food for thought

SIMULATIONS
micros that model the outside world

CODE MACHINE
Picturesque's assembler for the Amstrad

THE CRACKER

The spreadsheet designed for normal people who make mistakes. Instant error detection and easy correction. Yes, this is a special feature. It means that what you do is right, first time, most times. For Businessmen, Engineers, Scientists and most simpletons.

£100+£2pp+VAT, CP/M-Z80, CP/M-86, MP/M-86, CCP/M-86, MSDOS, PCDOS

DISASSEMBLERS, Z80, 8086

Powerful practical file based disassemblers. Produces error mesages, full listings and cross-reference tables. The 16-bit version suitable for whole 8086 family and 8087. This version can handle. .CMD, .COM and .EXE files also ROMS.

£80+VAT, CP/M-Z80, CP/M-86, MP/M-86, CCP/M-86, MSDOS, PCDOS

TRANSLATOR Z80 TO 8086

This is a single pass translator designed to allow you to get your Z80 source code into an 8086 form easily. It has no real size limit and works fast. Data areas handled intelligently. Output for popular assemblers. An easy way to learn 8086 assembly language.

£80+VAT CP/M-Z80, CP/M-86, MP/M-86, CCP/M-86, MSDOS, PCDOS

Software Technology Ltd

PO BOX 724, BIRMINGHAM B15 3HQ
TEL: 021-454 3330
TELEX: 337675 TELPES G

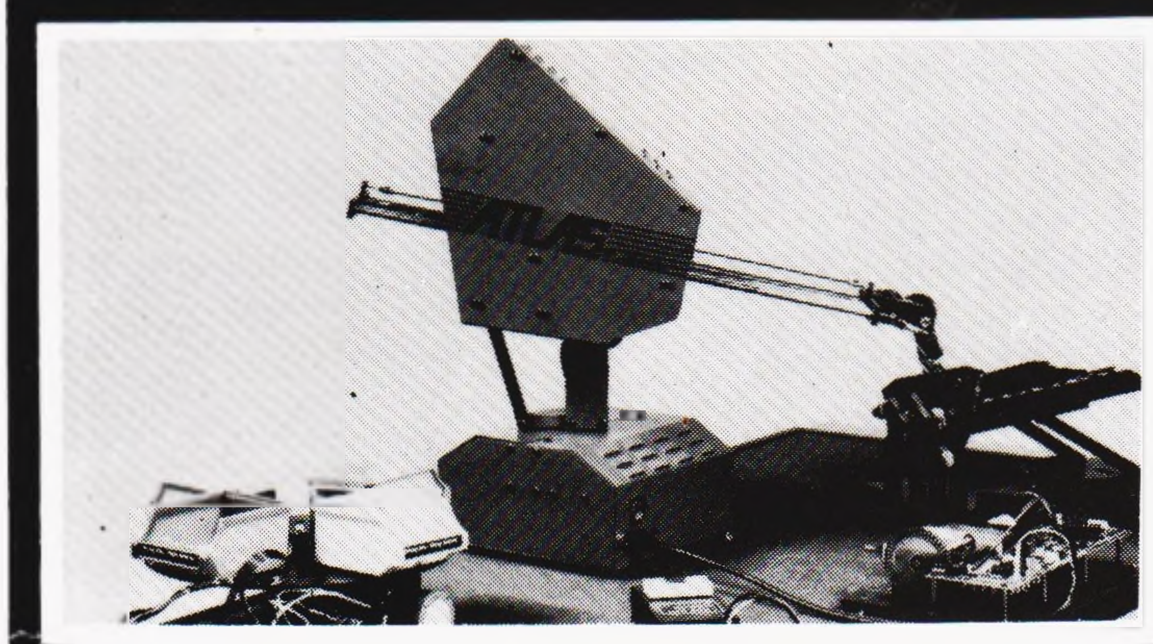
THE MALTRON KEYBOARD

Why Hasn't Anyone Thought Of It Before?

A keyboard to fit hands. The shape tells your fingers where they are and removes wrist angles — the main cause of unpleasant aches and pains in wrists, neck and shoulders. A new letter layout for modern usage based on a computer analysis of the English language. Much easier to learn, far fewer mistakes, thumbs used productively..... **But** at a flick of a switch you can retain the century old QWERTY layout but have all the advantages of a truly ergonomic shape. For greater comfort and efficiency — think MALTRON. Illustration shows the MALTRON for the IBM PC. Keyboards for BBC Micro and others available.

Contact: **P.C.D. MALTRON LTD**
15 Orchard Lane,
East Molesey,
Surrey KT8 0BN
Tel: 01-398-3265

The LJ Robotic Work-cell system



Based around the popular LJ ATLAS Robot, this work-cell provides an automatic parts-selection system running under full microcomputer control.

For full details of this new work-cell and other LJ robotic systems send for our comprehensive data sheets.

L.J. Electronics Ltd.
Francis Way, Bowthorpe Industrial Estate, Norwich NR5 9JA.
Telephone: (0603) 748001 Telex: 975504.

AMSTRAD INTERFACES

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

★★ MODEM ★★
★ £153.00 ★

Incorporating serial and parallel interfaces to allow software control of all functions each feature controlled from basic with the bar commands Call from m/c or on entering bar modem all controls are menu driven for ease of use. bell/ccitt standards 300/300 600 1200 1200/75 75/1200 full and half duplex Auto dial and auto answer contact bulletin boards prestel compatible software buletin on its own sideways Rom Unique panel display it displays what the modem is doing mode of operation and digits when auto dialing. standard B T plug connector *Note this modem is not B T approved*

★★ SIDEWAYS ROM ★★
★ £26.05 ★

The unit holds 4 Roms Each can be 2 4 8 or 16K in size incorporating a device to allow slower Roms to be used less than Amstrad suggested 200. that means cheaper Roms free utility Rom with every unit

RS232

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

£39.96

PARALLEL PORT

Make that Robot move
Control electrical appliance
Twin 8 bit ports
Operates direct from basic
2 × 14 way speedblock connectors

£22.57

8 BIT PRINTER PORT

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

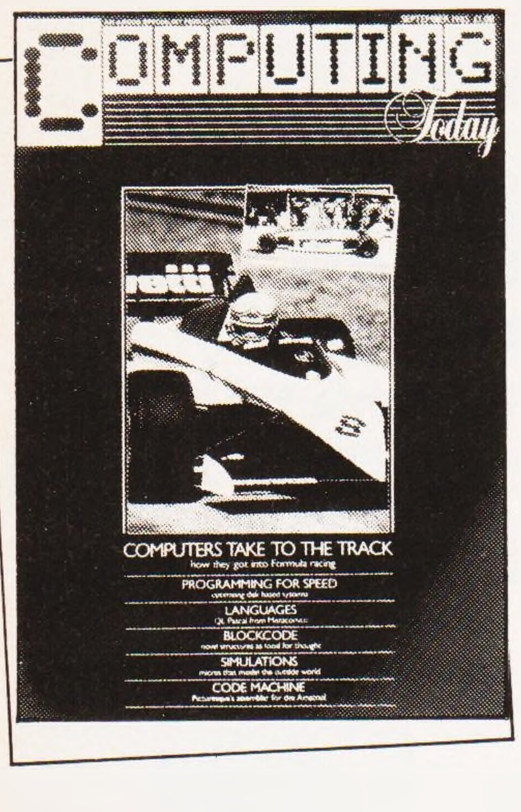
£17.35

All units are cased and have through connectors
★ Please add VAT ★

COMPUTER HARDWARE & SOFTWARE

KDS ELECTRONICS

15 Hill Street, Hunstanton, Norfolk PE36 5BS
Tel: (04853) 2076



CONTENTS

VOL 7 NO 9 SEPTEMBER 1985

REGULARS

SERIES

GENERAL FEATURES

SPECIAL FEATURES

Next Month's Computing Today.....47

COVER

Computers in sporting events are nothing new. Computerised lap timings have been familiar sights on our TV screens for years. In motor racing, computer systems are often used to control and monitor fuel flow. Now, computer aided design techniques are being employed to perfect wishbone suspension systems for Formula One cars. Story: page 20.

Editor: Don Thomasson
Assistant Editor: Jamie Clary
Technical Illustrator: Jerry Fowler
Additional Illustration: Grant Robertson
Advertisement Manager: Anthony Shelton
Classified Sales Executive: Caroline Falkner
Advertisement Copy Control: Sue Couchman, Lynn Collis
Publishing Director: Peter Welham
Origination and Design: Design International
Cover Design: Argus Design

ABC **Member of the Audit Bureau of Circulation**
ISSN 0142-7210

Computing Today is normally published on the second Friday in the month preceeding the cover date. Distributed by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, Lodnon EC2A 4JS. 01-247 8233. Printed by: Alabaster Passmore & Sons Ltd, Maidstone, Kent.

Subscription notes; UK (£16.20) including postage. Airmail and other rates upon application to Computing Today Subscriptions Department, Infonet Ltd., Times House, 179 The Marlowes, Hemel Hempstead, Herts. HP1 1BB England (phone 0442 48432).

EDITORIAL & ADVERTISEMENT OFFICE: 1 Golden Square, London W1R 3AB. Telephone: 01-437 0626. Telex: 8811896.

WDSoftware

For the QL

WD Utilities (3rd ed) **(base £5.50)**

PRINT 60-file DIRectory or view it on one screen, one-key LOAD. COPY or PRINT 60 files with one key (allows for namesakes). Multiple FORMATting to prevent corruption by stretching of tape. TOOLkit to give dated, numbered modules in program development. PRUNE old files to release space (one key DELETEs a file). Full instructions in QUILL file. Use up to 6 EXTRA MICRODRIVES (add on your Spectrum ones)!

WD Utilities for CST Disks **(base £8)**

100-file capacity, for CST/Compumatamate disk system AND up to 4 extra microdrives. User-friendly timesavers.

Ref QL (4th ed) **(base £4)**

700 useful QL references in an ARCHIVE file. Too long to share a cartridge.

For Spectrum/QL/BBC

WD Morse Tutor **(base £4)**

From absolute beginner to beyond RYA and Amateur Radio receiving. Adjust pitch. Set speed to your test level (4-18 wpm). Learn from single characters, via groups with wide spaces to random sentences; decrease spacing to normal. Write down what you hear, then CHECK on Screen or Printer (or speech for Spectrum fitted with Currah Microspeech). Also own message, random figures, letters or mixed.

For Spectrum 48K

Tradewind **(base £4)**

Sailing/trading strategy game with graphic surprises.

Jersey Quest **(base £4)**

Text adventure with Bergerac and the Dragon, (not disk).

Prices: (incl Europe postage, elsewhere add £1). Spectrum/BBC cassettes, base price only. QL or Spectrum Microdrives. £2/cartridge plus base price. 5¼" floppies. £2 plus base Morse for £11.30. 3½" floppies, £4 plus base.

Two or more programs on one medium — pay medium plus base EG. WD Utilities and RefQL for £10.50, but IMPOSSIBLE to mix QL/BBC/Spectrum programs on one medium. Send YOUR cartridge and base price, but FORMAT it FIRST in your DRIVE 1 for compatibility.

WDSoftware, Hilltop, St Mary, Jersey.
Tel: (0534) 81392 Dept CT

SYSTEM SCIENCE

C Compilers

16-Bit		8-Bit	
DeSmet C	£139.00	Aztec C][— Apple][	£175.00
Lattice C	£425.00	Aztec C II/BAS	£165.00
C86 & optimiser	£345.00	Aztec C II/COM	£295.00
Microsoft C ver 3.0	£475.00	C/80 Software Tool.	£50.00
Aztec C86/BAS	£195.00	C/80 Mathpak	£30.00
Aztec C86/COM	£395.00	BDS C	£125.00
C80 requires LINK	£50.00	ECO-C for Z80 Code	£185.00

LISP Interpreters

LISP-80 S.Toolw/ks	£45.00	LISP-80 S. Toolw/ks	£45.00
MuLISP/MuSTAR	£275.00	MuLISP/MuSTAR	£190.00
MuMATH	£215.00	MuMATH	£235.00
IQ LISP	£195.00	micro-PROLOG	£call

FORTH-83, Lab. Microsystems

PC-FORTH	£89.00	Z80-FORTH	£89.00
8086-FORTH	£89.00	Floating point ext.	£89.00
Floating pt-sw/8087	£89.00		

ASSEMBLERS

Microsoft8086	£139.00	**M/soft** MACRO-80	£185.00
2500AD 8086	£89.00	**2500AD** Z80	£89.00
DR Assembler Plus	£185.00	**DR** Assembler Plus	£185.00
Cross-Assemblers	£call	**Cross**-Assemblers	£call

Editors

Final Word-IMB,MS-DOS	£275.00	**Final Word**-CP/M	£275.00
SEE IMB, Apr.	£50.00	**PMATE IMB**, Apr.	£195.00
Vedit Plus-IMB,MS-DOS	£215.00	**EC-editor** IMB	£65.00
FirsTime C (syn.check)	£275.00	**FirsTime Pascal**	£245.00
Tools- many C tools available		Dbase to C conv.	£995.00
Crosstalk comms-IBM,Apr.	£125.00	Uniform- disc conv.	£69.00
Pascal Compilers	£call	Fortran Compilers	£call

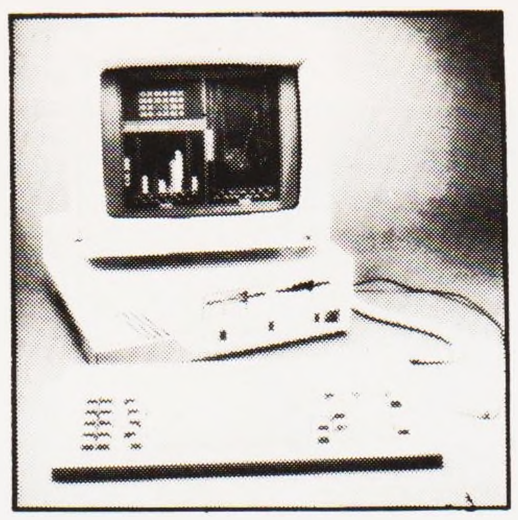
VENIX-86 full Unix for IMB PC-XT and PC-AT from **£850.00**

HSC 16 bit Co-Processors for Z80 CP/M systems
— choice of 8086 or 6800 — 6 MHz. clock
— MS-DOS, CP/M-86 and CP/M-68K — 256Kb to 1.25 Mb memory
— use as RAM DISK under CP/M — fit most Z80 systems
— simple to install, simple to use. Prices from **£625.00**

Prices are exclusive of VAT and postage

6-7 West Smithfield, London EC1A 9JX Tel: 01-248-0962

RM NIMBUS
POWERFUL. FLEXIBLE AFFORDABLE
AVAILABLE



Research machines Nimbus: 16 bit super micro.
You've read the rave reviews — now try a demonstration. The new RM Nimbus can be seen and tested immediately.
Complete with full range of software, including: word processing, spread sheet, database, accounts and graphics.

- The fastest 16 bit business computer
- Built-in colour hi-resolution graphics
- 80186 Main processor running at 8 MHz
- RM graphics processor
- 8051 peripherals processor running at 11 MHz
- 8910 sound processor running at 11 MHz
- MS DOS version 3.05 operating system
- 192 K standard RAM expandable to 1 megabyte
- 2 × 720 K disk drives as standard
- Hard disk option — 10, 20, 40 or 80 megabyte
- Networking up to 64 stations
- Interfacing up to 30 peripherals devices can be attached, (printers, instruments, modem, etc)
- Full range of software now available
- MS WORD, Wordstar and professional word processors
- MULTIPLAN Supercalc spreadsheets
- Superfile, D Base II, Datamaster databases
- PEGASUS, SAGE, MULTIPAC, EASY JUNIOR accounts
- CAD packages, Colour screen dump
- Mouse and joystick operated painting packages
- Powerful RM basic, Logo and Pascal languages
- And much, much more

Telephone straightaway for an instant trial

Regional Systems
2 Greenleaf Road, Walthamstow, London E17 6QQ
Telephone: 01-521 7144

NEWS

VIDEO DIGITISER FOR COMMODORE 64

CRL has now entered the home computer peripheral market with a product that incorporates the latest technology at a price affordable by every computer owner.

The Video Digitiser Module for the Commodore 64 allows Video signals from any source to be displayed on the computer screen, stored to disk, processed and subsequently printed out. similar systems have been available before, but at £149.95 this model is said to bring a sophistication and ease of use previously out of reach of the home comuter owner.

Designed in Austria over the past two years, with a great deal of feedback from potential users, the digitiser is compatible with a normal video camera, video surveillance camera, video recorder, video-out on a TV or the output from a video-compatible weather satellite receiver.

Once a picture is digitised, an image is produced on-screen. Although 256 x 256 resolution is stored in memory, 200 x 160 points are visible, while the remainder of the image is accessible by 'panning' across the screen, under the control of the cursor keys. The image is on screen in four shades of grey which can be altered using the function keys to one of the 16 other colours available. This colour expansion is particularly useful for weather satellite pictures, where specific items need to be highlighted during scientific experiments, or in the preparation of artwork. Another feature, the incorporation of a 'light pen', will allow sections of the image to be isolated, converted to user defined graphics and integrated into the user's own programs.

The digistiser has many uses and is available mail order only, until the *Personal Computer World* Show at Olympia in September.

● For further information contact Time Vernon at CRL, or Crosweller · Publicity on 01-402 9134.

CANON PRINTER FOR MSX

The latest peripheral for Canon's V-20 MSX personal computer is the T-22A serial thermal dot matrix printer.

The T-22A is a quiet, compact, yet high performance printer which can be connected to an MSX machine for data printout. When connected to an MSX machine it performs a wide variety of tasks. Character sizes are standard, enlarged and condensed, with the printer offering 80 standard characters per line at a fast 56 characters per second, while in the condensed character mode it produces 140 characters per line at 62 cps.

A choice of two graphic image modes are available — single and double density — ensuring that high resolution graphics and program lists transfer easily onto paper. The T-22A uses thermal paper and can be roll or sheet fed.

Measuring a compact 312mm x 220mm x 89mm, the T22-A is available for £149 excluding VAT through the Canon dealer network.

● For further information contact Geoff Thorn/Linda Bndewell on (01) 773 3173.

SPECTRUM AND COMMODORE MICRO FACTS

Collins have published two new Gem Micro Facts glossaries presenting Spectrum and Commodore users with comprehensive coverage for all aspects of program design, development and writing.

They bring together all the terms, keywords and commands needed for the two machines, organising them in a straightforward A-Z format for quick and easy reference. Whether the programmer requires a machine code instruction, the precise form of standard BASIC command on the computer, or the way that the screen data is stored in the machine's memory, direct reference to the appropriate entry in the Gem Micro Facts will give the answer.

Topics covered also include error messages and character sets and there are tables of data summarising information. Cross references to related articles appear throughout the text.

Presented in the popular compact and handy Gem format, these two new titles contain the essential facts for using the Sinclair Spectrum and Commodore 64.

● For further information contact Nicole Gautama, (01) 493 7070 Extention 4580.

OFFICIAL GUIDE TO OXFORD PASCAL

Eastbourne-based publishing house, Holt Saunders, has announced the launch of the definitive guide to Oxford Pascal — recommended by Oxford Computer Systems.

Ian Sinclair, who has written over 30 books on computer-related topics was so impressed by Oxford Pascal that he decided to write the guide he felt it deserved.

There will be two versions of the book, which will be on sale in major stores at the end of August: **Oxford Pascal On The BBC Micro** and **Oxford Pascal On the Commodore 64** — both very similar in content and format.

Ian Sinclair says: "Pascal has been a language waiting for a good implementation and I believe that Oxford Pascal is just that implementation for the Commodore 64 and for the BBC micro."

The books have been welcomed by Oxford Computer Systems. Managing Director Alan Wyn Jones and Technical Director Tony Wilkes have given them their seal of approval, and are writing the Forward.

● For further information contact Kate Kentish/Landy Hashimi on (01) 388 9871.

BEEB THREE FROM LOGIC ENGINEERS

Surrey computer consultants, Logik Engineering, have announced the release of three new packages for the BBC micro.

BCOMP is a powerful and easy-to-use compiler for the BBC micro which converts standard BBC BASIC programs into a special form which is both more compact and faster to execute. The compiled program may be executed immediately or saved to tape or disc for later use .

BSIDE is a disk based program which, when used in conjunction with the BCMP BASIC Compiler, allows BASIC programs to be executed from a sideways ROM.

BASIC programs put into a sideways ROM are actually executed from within the sideways ROM and not downloaded as with the ROM filing system. This means that all of the BBC memory that is normally used for program storage is now available for data and high resolution graphics. This makes BSIDE particularly useful for applications which use large data arrays and/or the higher resolution screen modes.

DASM is a sophisticated 6502 machine code disassembler program for the BBC micro which provides the following features:

- Disassembly of programs in RAM, sideways ROM or disc files.
- Full labelling facility which allows the user to define up to 800 symbols (disk version) or 1500 (ROM version) which will then be used in disassembled listings. MOS entry points are included as pre-declared symbols.
- Automatic symbol table building which generates symbols for the destination of any combination of JMP, JSR and BRANCH instructions.
- Flexible control of output format which allows data tables and text strings to be clearly shown in the disassembled listing. Up to 200 (disk version) or 400 (ROM version) format controls may be defined.
- The output from DASM may be sent to any combination of the screen, printer and a named disk file.
- Disc files may be produced which can be *EXEC'd to produce source programs for re-assembly using the BBC's built in assembler.

Command entry uses the BBC micro's function keys together with menus and prompts. In conjunction with the use of easy to understand error messages this makes all the powerful features of DASM quick to learn and easy to understand.

● For more information contact R F Burns on (07375) 52170.

EPSON PRINTER USER'S HANDBOOK PUBLISHED

For all users of Epson FX, MX and RS series printers there's a new book to help use this equipment successfully: the Epson Printer User's Handbook.

Published by Century Communications, it is designed to take the mystery out of connecting and installing these printers, containing simple, explicit instructions and explanations.

There is also a wealth of advanced information for the more experienced user. From describing how to hook up the printer and set the DIP switch and command sequences that adapt the printer to different kinds of computer, the book goes on to explain the use of Epson's impressive graphics capabilities.

There are instructions for printing italics, boldface and other special characters and type fonts.

Advice follows on using the printers with word-processing, spreadsheet and database packages, including dBase, Lotus 1-2-3, Symphony, and Wordstar.

Tips on keeping the printer running longer, faster and more economically are included, and there is an appendix detailing the new Epson+ models and the new NLQ interface boards.

The book costs £9.95. ISBN 0 7126 0561 4.

● For further information contact Jane Farrow on (01) 902 8892.

ROM BASED SOFTWARE FOR ECHO MUSIC SYSTEM

Software development for the Echo three-octave musical keyboard from LVL is continuing as planned, with the introduction of new ROM-based software.

The Echo Music Keyboard is supplied complete with disk or cassette based software which allows the user to play music, change octave, adjust the tuning and select different musical instrument sounds such as the piano, organ or harmonium etc.

The new ROM has additional features, including recording and playback of music on disk or tape.

The fact that Echo itself is a computer language means that its capabilities are not limited by the BBC's standard sound commands — Echo itself controls the sound generator in the computer.

The recommended retail price of the ROM is £29.95 including VAT, and comes complete with manual.

● For further information contact Wayne Bradley, on (0602) 39400.

REAL-TIME ARACHNID

Paul Fray Ltd has announced an ingenious add-on for the BBC Microcomputer called the Spider.

Physically, a combination of sideways RAM and advanced ROM-based software, the Spider extends the capabilities of the BBC micro into serious control applications — in the home, the laboratory and in industry.

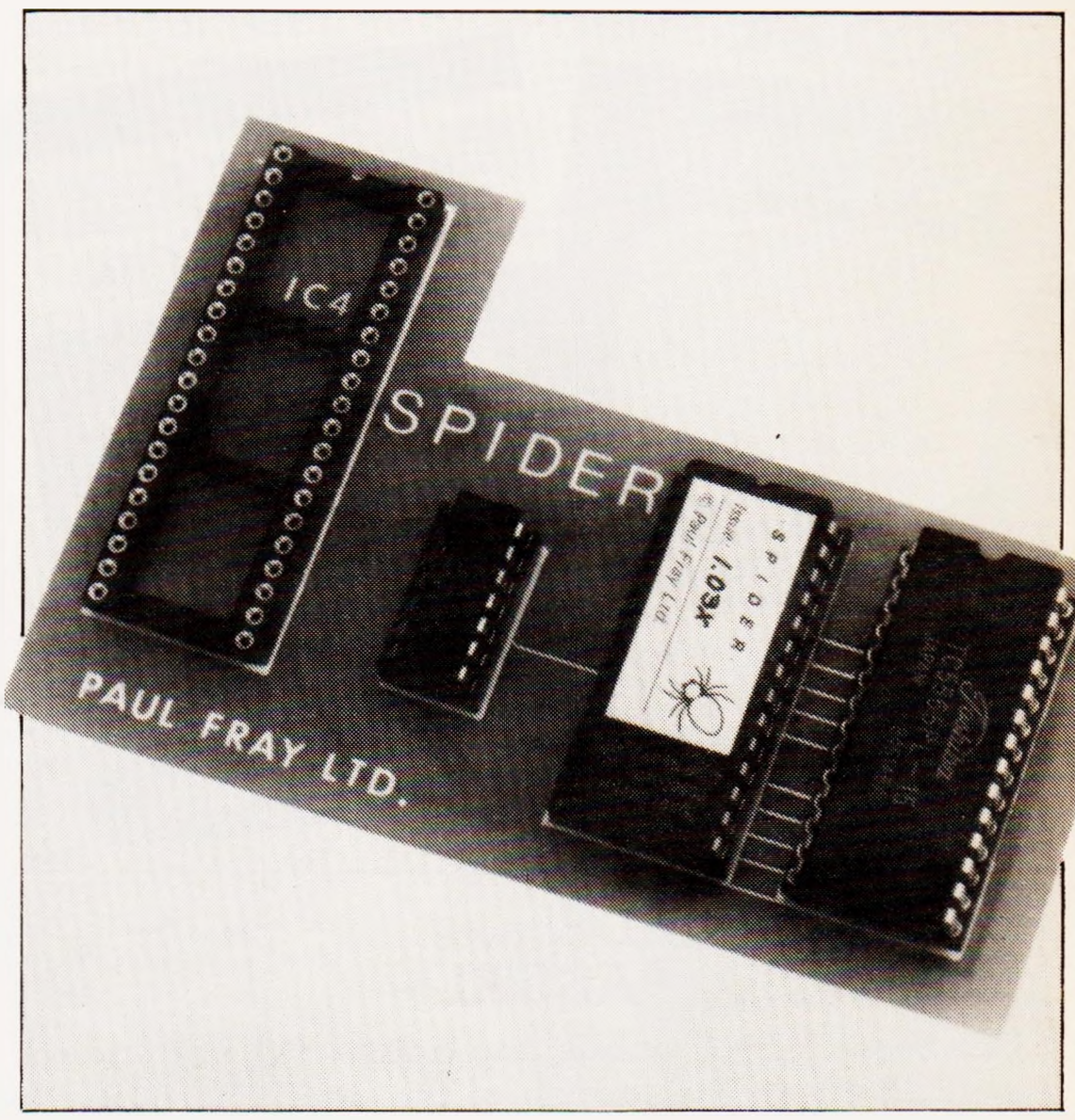
The Spider is quite unique in that it not only adds a set of practical real-time I/O tools to BBC BASIC, but allows the Beeb a degree of parallel processing.

Spider also provides a set of process timers. These can time-out quite independently of the signals from external devices. The computer can respond to any of these events with all the power and subtlety of BASIC procedure calls.

In the same way that a real spider catches flies, the computer can respond quickly and efficiently to an event in its 'web' of sensors. Should another 'fly' trigger the web during the brief period in which it is dealing with the first event, its presence can still be noted. It can then be dealt with as soon as the first has been fully secured.

Paul Fray Ltd is a Cambridge-based consultancy for Industrial and Educational micro users. It provides turnkey solutions which involve interfacing with various sensors, networking, and other real-time control and data capture problems. The spider is the Company's first product to be made available to the general public.

● For further information contact Dr Paul Fray on (0223) 66529.



SERPENT SCARA

Cybernetic Applications have launched a new training robot — the Serpent.

The Serpent is a horizontal movement robot complete with control interfaces for the BBC micro, the Commodore 64, and the Apple IIe.

SCARA robots originated in Japan and having become rapidly established in manufacturing in that country are now spreading throughout Europe and North America. The robot is designed, manufactured and sold from Britain. It is a World leader and Cybernetic Applications expects exports to follow in the wake of the success of the Mentor and Nepture robots abroad.

● For further information contact Mark Rayner on (0703) 37133.

THE CORE STORE

The Core Store is a new company set up to offer a complete service to all C language users.

The Core Store stocks a complete range of C compilers, utilities, books, function libraries and programming aids, all selected and evaluated to give the C programmer the tools he needs at an attractive price.

The company intends to maintain a register of C programmers in order to put people with common interests in contact with one-another via a C users' group. They are also acting as a UK liason to channel comments and suggestions to the ANSI Committee 3XJ10, which is currently drawing up a standard for the C language under the chairmanship of Thomas Plum.

● For further details contact Linda Holmes or Irene Dunn on (0605) 45420.

EPROM PROGRAMMERS FOR HIRE HANDLE LATEST DEVICES

Two new programming products, the P9020 EPROM Programmer and the XP640 Universal Programmer, are capable of handling devices up to 512 kbits, and are designed to simplify and increase throughput of programming.

Now available on hire from Microlease, the P9020 can program up to eight devices simultaneously. Features include an RS232 serial interface for downloading from the development system at rates up to 19.2 kbaud, and internal 64 kbit static RAM buffer expandable to 256 kbits. High-speed programming algorithms which can cut programming time by up to 90% are also offered. Reliability is assured by the comprehensive checking system for both devices and programmer.

Also new to the Microlease hire range, the XP640 provides everything that is needed to handle the complete design and development cycle of EPROMS. It combines an EPROM duplicator with an EPROM emulator module, to provide a comprehensive EPROM workstation in a bench-top package.

For ease of use, the XP640's set-up parameters can be stored for automatic recall on power-up. Interactive menu control and prompts simplify its operation. There is a wide range of PROM and editing functions. Editing may be used with the unit's on-board line display, or via its video output.

The programmer has an RS232C interface for remote control and a Centronics compatible parallel interface for hex dumps. User RAM is 512 kbits. A full range of programming and self-check functions are standard.

● For a copy of the latest Microlease hire catalogue and further information, contact Sandie Petrie, Microlease plc, Forbes House, Whitefriars Estate, Tudor Road, Harrow, Middx HA3 5SS. Telephone: 01-427 8822. Telex: 895 3165.

ELECTRON TRACKBALL

Wigmore House Limited are manufacturing the first trackball for the Acorn Electron.

The Wigmore trackball uses the existing joystick or analogue port on the Electron so that existing joystick software can still be used.

The Wigmore Electron trackball costs £24.90 including VAT. Mousepaint retails at £11.50 including VAT.

● For further information contact Wigmore House on (01) 734 8826 or (01) 734 0173.

FIRST CLASS PERIPHERALS OFFERS 10MB APPLE BACK-UP FOR £899

A 10MB Winchester subsystem, intended specifically for the Apple 2+ and the Apple 2e, is offered by First Class Peripherals Limited for an all-in price of £899.

Known as the Sider, the subsystem is available as a complete mail order package, ready to plug in to your Apple 2+ or 2e. It is being shown in the UK for the first time at the PC User Show, Olympia.

"This really is a plug-and-play, all inclusive price," says FCP's Nicky Perfect. "The price includes cables, host adaptor, installation software, manual, VAT and carriage. Backed by Xebec, this is a piece of absolutely first-class hardware, and we believe the price is unbeatable."

The Sider boots directly to the Apple 2+ or 2e, and a rigorous quality assurance program ensures that the Sider can be plugged in without problems A significant feature is that the disk can be partitioned, allowing the user to allocate space to four operating systems on the same disk. The Sider supports Apple DOS 3.3, PRO DOS, Apple PASCAL and CP/M.

The Sider will be marketed exclusively by First Class Peripherals Limited, which was set up last month to handle mail order sales of "plug-and-play" computer peripherals.

● First Class Peripherals Ltd, 1st Floor, Cockayne House, Crockhamwell Road, Reading RG5 3JH. Tel: (0734) 699663.

NEW MSX AND £50 TRADE-IN

Until 31st August Mitsubishi Electric (UK) Ltd will be offering a £50 trade-in on any computer or video games machine against the sale of a new Mitsubishi MSX home computer.

This offer will run additionally to the free starter software pack — consisting of six top tape games worth over £45 — and a comprehensive 300 page operation instruction manual and basic language handbook .

The Mitsubishi 32k ML-F48 and 64k ML-F80 MSX home computers retail at around £219 and £275 respectively and are available from most Mitsubishi electric dealers.

BRIGHTON TECH BUYS COLNE LATHE

Last month engineering students at Brighton Technical College were able for the first time to use their BBC microcomputer ECONET system in conjunction with a CNC machine tool purpose-designed for education and training.

The Colne 5 CNC training lathe has software offering the facility for toolpath-emulation colour graphics. This allows students from the college's Mechanial and Production Engineering Departments to check a programmed cut before cutting commences. Using the department's ECONET Network system they can use additional software kits to work at separate BBC microcomputers, thereby sharing access to the lathe itself.

Colne Robotics gives high priority to the safety aspect of equipment destined for training environments such as schools and technical colleges. The Colne 5 lathe, for example, incorporates a lockable panic button which disconnects the mains supply and a transparent guard covering all moving parts.

In addition to software on disk and ROM for the BBC Model B, this bench-top lathe can also be supplied with software on cartridge for the Commodore 64 micro. The complete package for the BBC includes lathe hardware, comprehensive manual and software kit, plus a selection of accessories and tools, for £2295.00.

17th century manuscripts into modern notation. Over the course of several years, I developed (working mainly at night) software on a DEC PDP 11 computer for the encoding, editing and printing of this music."

Vendome's main goal was to assemble the music in publishable form and he therefore aimed at a combination of speed of input and quality of output. To date, nearly ten thousand pages of music have been processed, many of which have now been published.

● For more information contact Andrew Potter on (0865) 56767.

ANADEX SMOOTH HAZARDOUS LANDING

In 1984 Southampton University designed a flight simulator which, later the same year, won the Royal Aeronautical Society CAE prize.

The system's potential to simulate hazardous landing conditions has now led to commercial interest from an Australian flying school, and the University's continuing development of high speed computer architecture is also being applied to important navigational, cardiac and VLSI studies.

Based on the formal aerodynamic model of a Cranfield A1 high performance aerobatic aircraft, the flight simulator has the capacity to imitate hazardous flying and landing conditions. The program took nine months to develop using an Anadex DP-9725B colour printer for important and highly accurate screen-dumping of VDU displays in order to fault-find and complete the design and software. The University's departments of Electronics and Aeronautics both developed the system and to date over thirty military and civil pilots have successfully flown the simulator.

However, flight simulation only represents a small part of the advanced system design work being carried out by the University's Department of Electronics. The combined energies of students and lecturers are also devoted to R & D in integrated circuit mask design and IC design systems.

● For further information please contact Tony Grant on (04868) 23882.

THE (8) MINUTE WALTZ

Oxford Music Processors (OMP) is a computer company developed at the University of Oxford, which is likely to revolutionise the way in which music notation is prepared for printing.

Originally devised as a tool for realising extensive academic research it will enable the music publishing industry and musicians of all kinds, whether professional or amateur, to provide themselves easily with music notation to the highest standard.

OMP carries out almost all the tasks which have traditionally required painstaking and slow manual arrangement of music notation on the page. Highly specialised skills have been required to arrange the notation in such a way as to make the music easy to read and interpret. With OMP, the computer takes all the various rules of notation design into consideration and arrives at a finished product within a tenth of the time of traditional alternatives.

Richard Vendome, who is developing OMP at the Faculty of Music at Oxford, says: "My post-graduate thesis presented the enormous task of transcribing 400 keyboard works from

ILEA RECOMMEND RESEARCH MACHINES NIMBUS

A plan to supply a single UK manufactured micro to more than one thousand colleges and schools has been announced by the Inner London Education Authority (ILEA).

ILEA has agreed an open-ended programme with the Oxford-based microcomputer manufacturer, Research Machines Limited to supply London schools and colleges with the 16-bit RM Nimbus micro.

The agreement, which could be worth more than £2m, was won by Research Machines Limited in competition with the UK's largest microcomputer company, ACT.

"This is by far the most substantial recommendation for 16-bit microcomputers ever made by a UK local education authority," said Derek Esterson, ILEA Computer Advisor.

According to Mike Fischer, managing director of Research Machines Limited: "If every school and college within ILEA were to order only two Nimbus' that would amount to orders for Research Machines worth around £2m. We are naturally very pleased to have won ILEA's recommendation."

Said Esterson: "We looked at a number of systems, including micros from IBM, ICL, Comart, Olivetti, and Sinclair, among others, and short listed Research Machines', Nimbus, ACT's Apricot and the LSIA Octopus."

"ILEA conducted extensive benchmark tests on the short listed micros and the performance of the RM Nimbus showed it to be up to three times as fast on some tests as the ACT machine," continued Esterson. "As a result, ILEA has decided to recommend the RM Nimbus as the standard across-the-board 16-bit micro for all the colleges and schools within ILEA's area of responsibility.

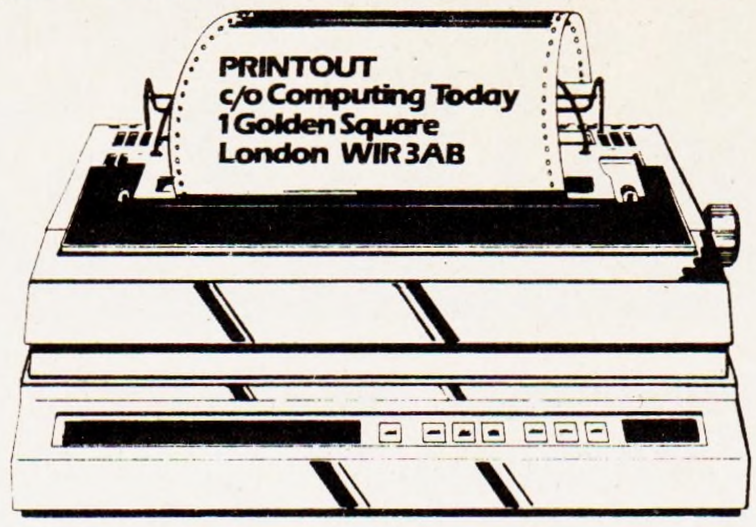
"As well as being exceptionally fast, the Nimbus has excellent graphics capabilities. It runs under the industry-standard MS-DOS operating system, which means there will be no shortage of good software available. It is also very good value on any price/performance scale."

Educational purchasers of the 64-station RM Nimbus Network will automatically get a bundle of professional quality software, worth more than £5,000. Licensed to run on up to 16 stations, the software includes Microsoft's word-processor and Multiplan spreadsheet, the Superfile free form multi-user database from Southdata, RM CAD draughting package, PC Paintbrush, Quest, Touch 'n' Go keyboard tutor, RM Pascal, RM Basic, RM Logo and the program editor Steed.

● For further information contact Christian de Quincey on (01) 235 7040.

PRINTOUT

Your opportunity to ask questions, put us straight, seek advice.



FASTEST UNIX

Dear Sir,

The increasing complexity of programs designed for today's microcomputers has resulted in greater and greater emphasis being placed on the speed at which these machines will run. This is particularly the case with computers designed to run the UNIX operating system, which can be notoriously slow on an underpowered machine.

It was with great interest therefore that we observed the latest publicity campaign of the giant electronics company, Intel, exhibiting at London's Unix Show running at Olympia last week.

Large signs proclaimed, "The fastest Unix on a micro", and this was supposedly backed up by a series of technical tests, laid down by the computer magazine, *Byte,*

On closer examination, however, we were dismayed to learn that the computer in question was actually considerably slower than our own British designed and built system, the IMP Mentor, being shown at the same exhibition. On the most widely used performance test for such machines, the Sieve of Erathosthenes, the Mentor has been found to run at a speed four times that stated by Intel for its own machine.

We would be grateful therefore if the company in question could be warned of the illegality of its advertising campaign as soon as possible.

Yours faithfully,
Mark I'Anson,
Managing director,
Integrated Micro Products Ltd.

STOUT FELLOW

Dear Sir,

With reference to the enquiry from David Bell, on page 14 of your June issue, I may be able to help.

I have a 48K Video-Genie and use the Newdos 80 V.2 operating system. With the aid of 'Superzap' and 'Macromon', I imagine that I could probably relocate the program to disc and make the necessary alterations to the printing routines (memory-mapped instead of ports).

Perhaps you will pass this letter to Mr Bell. I would need a copy of the tape in question and to know what DOS he is using. There would be no question of any charges, whether or not I should be successful.

D. W. Arnold,
20 Hargate Road,
Buxton,
Derbyshire SK17 9BL.

A LIVELY CRITIQUE

Dear Sir,

I don't know whether this is a general view, but to me it appears that there is increasingly less of interest in the computing magazines (yours being an honourable exception). There seems to be three main reasons for this:

1) Some magazines appear to have the policy of stuffing themselves full of advertisements and not worrying too much about the quality of what comes between adverts (You were guilty of this in 1984, I feel.)
2) There is a concentration of very long listings in machine code, which are all but incomprehensible to those who are not fortunate enough to have one of the few favoured machines (mostly the ones reputed to have been designed by Robert Maxwell, I believe.)
3) There is an increasing tendency to write very esoteric articles which only a very small proportion of average readers could make much out of. Over the years, I have (I hope) become more computer literate, but I find the percentage of articles I can fully comprehend remains much the same. I have *not* been at the computer game so long that I cannot remember what it was like to be a beginner.

In your last issue, I particularly liked the leading article answering the question: "What do I do with a computer?" Not as you say, easy to answer verbally, but I would be interested to read how others use their micros. I felt a touch of fellow feeling about word processors if not about motor-racing! I feel that it is very important to remember that although computers are not human their users are. Because of this, often the corresponding pages of a magazine are among the most interesting.

I would not hold Computing Today guilty in general under my third point, but I did feel the news pages at the beginning might well have put a beginner off. You probably need a degree in electronics to understand what the lead story was about, and while I am not suggesting it should have been left out, a gentler start might have been welcome. If, for example, you had started off with the news on the Atari, any reader would at least have understood some of the contents.

These comments are meant to be helpful, not critical. I have enjoyed much of your contents this year.

Yours faithfully,
G. T. Childs.

Your comments have been amongst the most useful we have received, Mr Childs. Your third point is of particular interest, and your remark about the lead story in the July edition is quite valid; the acronym 'KISS' (Keep It Simple Stupid) has been tatooed onto the news editor's forehead.

A CRACK AT CRIBBAGE

Dear Cribbage Consortium,

I am not quite sure why you felt you had to attack "the anti-GOTO school" in your excellent Cribbage Player series. What is important in our school is to have a structure that is easy to read and, more important, mess with at a later date.

I have taken up the challenge although I feel rather like farmer Giles who was suddenly asked the way back by a hopelessly lost tourist, and after trying to explain the intricate route, finally burst out: "If I were you I wouldn't start from here at all!". However here goes:

```
1630FOR I=1 TO 2:FOR J=1
TO 2 etc.
1640–1740 unchanged
1750IF HN(1)=HN(2) THEN
   PRINT TAB(12,5);"We'll cut
   again.": PROCDEL(300):
   PROCCD(18):PROCCD
   (11):I=1:NEXT I
1760–1770 are removed
1780I=2:NEXT I
```

I do not have a BBC so I have not been able to test the above. However, I do intend to convert the program to a DRAGON 64. The game has always fascinated me. In fact I have been using a spare peg from my cribbage board to switch a micro-switch from lower case to "ALL CAP" on the machine on which I am writing this letter.

Yours sincerely,
Robert J. Smith.

We weren't simply kicking the anti-GOTO school, but suggesting that there are times when the use of GOTO is a more simple and direct solution. Our thanks though, Robert, for providing an alternative.

Subscriptions

Personally, we think you'll like our approach to microcomputing. Each month, we invite our readers to join us in an abundance of feature articles, projects, general topics, news and reviews — all to help committed micro users make more of their microcomputers at home or at work.

However, if you've ever missed a copy of Computing Today on the newstands, you'll not need us to tell you how valuable a subscription can be. Subscribe to CT and for a whole year you can sit back, assured that each issue, lovingly wrapped, will find its way through your letter box.

And it's not difficult! All you have to do is fill in the form below, cut it out and send it (or a photocopy) with your cheque or Postal Order (made payable to ASP Ltd) to:

COMPUTING TODAY Subscriptions,

Infonet Ltd,
Times House,
179 The Marlowes,
Hemel Hempstead,
Herts HP1 1BB.

Alternatively, you can pay by Access or Barclaycard in which case, simply fill in your card number, sign the form and send it off. Please don't send in your card.

Looking for a magazine with a professional approach with material written by micro users for micro users? Why not do yourself a favour and make 1985 the year you subscribe to Computing Today and we'll give you a truly personal approach to microcomputing.

SUBSCRIPTION ORDER FORM

Cut out and SEND TO :

COMPUTING TODAY Subscriptions
INFONET LTD,
TIMES HOUSE,
179 THE MARLOWES,
HEMEL HEMPSTEAD,
HERTS HP1 1BB.

Please commence my subscription to Computing Today with the ..... issue.

SUBSCRIPTION RATES
(tick ☐ as appropriate)

£16.20 for 12 issues UK	☐
£18.70 for 12 issues Overseas Surface	☐
£51.20 for 12 issues Overseas Air Mail	☐

I am enclosing my (delete as necessary) cheque/ Postal Order/ International Money Order for £.....
(made payable to ASP Ltd)
or
Debit my Access/ Barclaycard*
(*delete as necessary)

ACCESS
BARCLAYCARD VISA

Please use BLOCK CAPITALS and include postcodes.

NAME (Mr/ Mrs/ Miss) .....
delete accordingly

ADDRESS .....

.....

..... POSTCODE .....

Signature .....

Date .....

TALKING SHOP

Don Thomasson

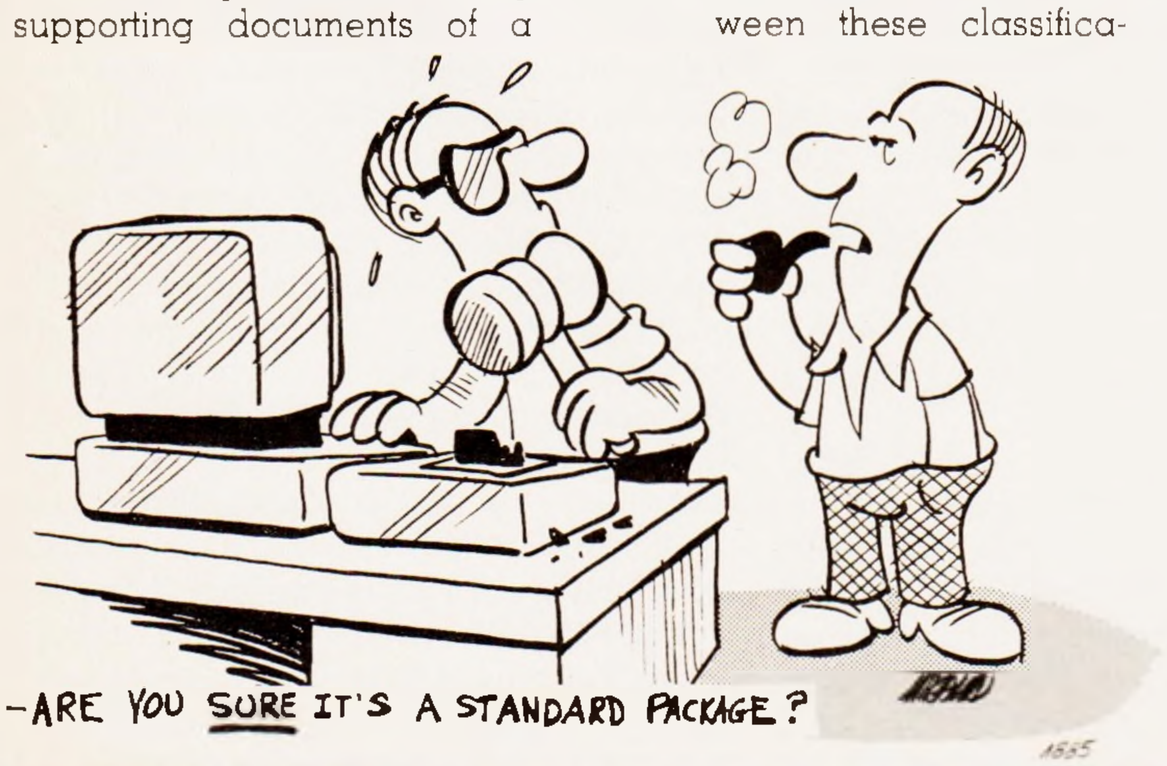
Designers of computing equipment and programs face a perpetual dilemma: If they design for maximum flexibility, their products may be stigmatised as too difficult to use. If they aim for user-friendliness, they may be criticised on the grounds of inadequate facilities. In many cases, the criticisms are justified, but some designs do manage to steer the perilous course that gives good results without demanding too much of the user.

The key to success in this matter is often the documentation provided. The User Manual supplied with the equipment can rarely tell the whole story, and that is, perhaps, just as well, since a thorough-going description of all aspects of the system might only serve to bewilder a first-time user.

What is left out of the User Manual may be covered by supporting documents of a more detailed nature, and by books produced by independent sources. It is a pity that such documentation is sometimes not advertised too widely, while dealers seem unaware of its existence. Independent books may be quite difficult to find among the mass of paperbacks selected by distributors, who seem to prefer trivia to anything really useful.

It has been said that computer users can be grouped in three classes;

- The tyro, who knows nothing beyond what the User Manual tells him, and who might find difficulty in understanding anything more complex.
- The expert, who wants full and detailed information on every aspect of the system.
- Those who fall between these classifications, being in the process of graduating from beginner to near-expert.



Independent books should serve the last of these groups, allowing the second group to have the full advantage of formal documentation.

Our own editonal policy is partially aimed at the third group, and this has had some odd consequences. First, we are receiving a lot of letters and submissions from abroad, and these suggest that there is a dearth of middle-level data in some other countries. Secondly, it is considered that our readers are unlikely to buy much equipment or software. They have presumably bought a computer already, and will want to write their own programs.

This may be superficially true, but needs qualification. AMSTRAD users are likely to be interested in extension devices, like the MAXAM, which we reviewed recently. There will always be a demand for printer paper, tapes, discs and other media, which can be quite difficult to obtain locally in some areas. (Or are available at inflated prices!)

But, as usual, we are straying from the original point, which was the need to compromise between flexibility and user-friendliness. There are many instances which might be used to illustrate this. Sound commands can be very tedious unless you are given useful hints on simplifying their use, and some colour control systems are as bad.

In these respects, the User Manuals tend to be inadequate. A little extended information would make matters much clearer, and though independent books often fill the gap, it should not be necessary for them to do so.

We would be interested to hear what kind of information the middle-level user needs most. We might be able to run a suitable series, or encourage the production of helpful books.

□ □ □

HOWEVER, THERE are indications that at least one manufacturer is beginning to see the need for middle-level information, having noted a preference for his products among those who are climbing towards true expertise. This is a hopeful sign, in more ways than one. First, because purchasers are showing better judgement, and secondly because the manufacturer is responding.

There was a time when experts tended - perhaps unintentionally - to guard their specialised knowledge behind a smokescreen of jargon. The time for that is past. It is in everyone's interest to spread such knowledge more widely. If something can be explained in simple terms, it can scarcely be worth concealing.

A possible exception is knowledge of how to circumvent program protection, which rightly tends to be kept on the secret list, though most systems are breakable with ease, once you have the key information. Somewhere in the system is a bit or variable that needs to be changed. Once this is found, all becomes obvious. (My personal opinion is that there is no such thing as a cast-iron protection system. At best, it can only be a deterrent.)

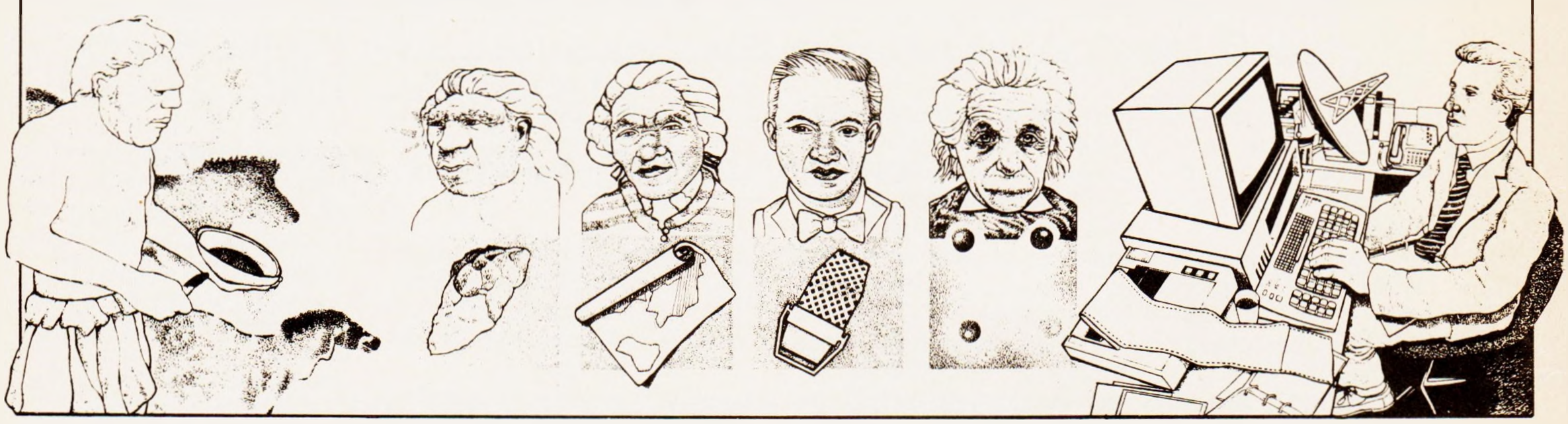
With this exception, it is desirable that everyone who has access to a computer system should know how to use it to best advantage. Some may find the going tough, but this usually means that they need additional data of some sort.

Remember, we want to play our part in this, but we can only do so if you tell us what you need to know.

LESSONS OF HISTORY

Bill Horne

Part six: integration.



During the nineteen-sixties, the technology of integrated circuit manufacture developed steadily, and it became possible to visualise the manufacture of silicon chips with a thousand or more transistors built into the surface. This work ran in parallel with investigations of possible applications for such chips, and while families of relatively simple components were developed there was some difficulty in deciding the right course to take in respect of more complex devices.

The obvious route was towards the creation of logic complexes for use in computers and similar equipment, but this inevitably led to an involvement with system concepts, and especially software concepts. There was a schism between those who accepted this involvement and those who wished to evade it.

The simpler approach was based on making 'bit-slice' components. These were sections of computer logic capable of handling a limited number of data bits. A typical example provides for four data bits, and offers a full range of simple arithmetic and logic functions, plus sixteen storage registers. Four such components would suffice for a 16-bit logic system, eight would handle 32 bit data.

Because it could only perform elementary functions, such a device required an external control system which could call up the sequence of functions needed to execute a given program instruction. This was in line with existing computer practice, which used 'microprograms' to translate operation codes into function sequences. Such a system allowed extreme flexibility, since almost any imaginable set of operation codes could be accommodated by writing a suitable microprogram. This flexibility cut two ways, however. Once a given microprogram had been set up, defining a particular set of operation codes, it was necessary to create a complete range of software tools to suit the system, and that was an expensive process.

One solution which was adopted was to use bit-slice components to 'emulate' existing computers, so that the necessary software tools were already available. While this solved one problem, it created computers which had been designed a decade earlier, and which were no longer fully competitive in terms of performance. As a result, the bit slice approach has tended to become limited to special-purpose systems.

The alternative approach was to incorporate all the necessary elements for the central processor and its microprogram into a single component. This meant using more transistor elements, commonly expressed as going to a higher level of integration. Since each element required the dissipation of a given amount of power, this either meant raising the overall power level or reducing the power dissipated in each element. Operating speed is related to power, so a reduction in power level implied a reduction in speed.

CAPITAL COST

The resulting components were therefore expected to be slower than the bit-slice version, but against that they offered one supreme advantage: since the internal microprogram defined a unique set of operation codes, it was possible to write supporting software tools which would apply wherever the highly-integrated component was used. The cost of the software could therefore be spread over a vastly greater number of hardware units. Where perhaps a thousand bit-slice units might be served by a set of software, a million or more 'microprocessors' might use the same set of tools. Instead of being a major factor, the cost of software shrank almost to insignificance.

It might be suggested that this is an oversimplified view, since it ignores the fact that the creation of software became a capital cost, which had to be met before the product could be marketed. It would be equally true to say that the capital cost of creating a new hardware design was far greater than in

older technologies.

The result was the microprocessors could only be designed in the expectation of very large volumes of sales, and with the support of ample capital. These conditions were best satisfied in America, and though microprocessors have since been developed in Britain, the vast majority of the currently available designs are of American origin.

However, the creation of a computer based on a microprocessor is not just a matter of purchasing the necessary components and working out the interconnections between them. A great deal of consideration must be given to the system configuation, to the way the available store area is divided between fixed 'Read Only' memory (ROM) and alterable 'Random Access Memory' (RAM). The provision for connection of extemal peripheral devices must also be considered.

In this type of work, initial capital requirements were less severe, and a number of British firms began to undertake overall computer design with a measure of success, though the market was initially in the hands of a few small American firms which relied on bulk sales to keep prices down.

The factors controlling the market were numerous. Technical merit was not predominant, since those who could recognise it were few in number. Retail outlets were controlled by people who often relied on rumour as a guide to merit, and fast talkers gained undeserved success. Price was important, and that favoured those with access to the huge American market. Expenditure on sales was often more effective than comparable expenditure on design.

DERISION FROM DINOSAURS

Clearly, the advent of the microprocessor wrought an enormous change in the computer world. The early designs were not able to access large store areas, and this limited interest in the implementation of multiprogramming techniques. The 'mainframe' manufacturers were able to go on their way, deriding the 'microcomputer' as a mere toy by comparison with their enormous machines. In this, they failed to consider the possibility that microcomputers, used in numbers with each machine performing a dedicated task, might offer a more economical solution to some types of computing problems.

This led to a comparison between mainframe manufacturers and the dinosaurs, in that both might well vanish at the peak of their powers, but the true position was more complex.

There are types of computer work which will always need large amounts of storage. Weather forecasting, for example, involves the examination of vast amounts of data to estimate how that data will change with time. The accuracy of the prediction depends on the amount of data available for a given land area, and while a simplified system covering a limited area and a short period of time can be set up with the aid of a microprocessor, the accuracy of the results would be somewhat suspect.

A more general case relates to the maintenance of national data records, such as those relating to car ownership. Here a vast amount of data must be held available, and held in such a way that any single item can be found rapidly, while renewed reminders can be identified as due and prepared automatically.

The size of computers for such applications does not mean that they are especially complex. They are big in storage capacity, and may need to perform certain processes rapidly, but the actual programs may be relatively simple, perhaps simpler than the sort of programs which home-users now employ.

With all such systems, there is likely to be a bottle-neck in the initial entry of data, and in the entry of instructions for data retrieval. A small army of operators may be needed for such work, and they form the Achilles Heel of the whole system. Any error which they make is likely to be attributed to the computer, which is only following the essential rule 'garbage in, garbage out'. This means that if a computer is given false data, it will provide false data in return. It is not a thinking machine, and there are few signs that it ever will be. It is a tool which can perform miraculous things, but it will do so mindlessly. It may apply checks to incoming data, as a means of detecting errors, but will only do that if it is programmed to do so.

Once again, there was a split between two philosophies, on the one hand the vast and expensive mainframe approach, and on the other the more limited by cheaper microcomputer approach. In theory, there was a third approach, taking a middle course, in the 'minicomputer', but it was sometimes difficult to be sure where the boudnaries of that approach lay. Some minicomputers grew until they were almost mainframe systems, others shrank until they could be grouped with the larger microcomputers.

The minicomputers represented an attempt to obtain the best of both worlds, to reduce cost below the mainframe level while offering a performance above that of the microcomputer. This was the clear objective for large parts of the computer industry during the nineteen-seventies, and the routes towards that objective taken by different companies reflected the wide range of views regarding the precise nature of the target.

In itself, the introduction of the microprocessor had solved nothing. Parallel development in storage sytems was necessary, and this applied in particular to long-term bulk storage. The cost of a big magnetic tape storage system was acceptable in the context of a mainframe computer, but could not be considered in relation to a microcomputer. In the early days, punched paper tape was used as a substitute, but it was inconvenient and still a little expensive.

The use of domestic cassette recorders was the breakthrough that resolved this problem, though the failure rate with early systems was far from encouraging. Progressive enhancements have produced higher reliability in some cases, but there are still systems which lack the refinements of the best examples.

ENTER THE VDU

An equally important innovation was the Video Display Unit, which allowed rapid display of data, coupled with convenient input by use of a typewriter-like keyboard. This removed the need for costly printers, which often concealed the last letters printed and had other faults. It also speeded up the execution of simple processes, and allowed the introduction of on-screen data editing, the essential feature of word processing.

That, however, needed a further innovation, the low-cost printer. During the late nineteen-seventies, the cost of printers tumbled, though the new devices were often computers in their own right, with not one central processor but two or more.

Finally, the costly disc recording system of the big computers were paralleled by the 'floppy disc', which cost a good deal more than the cassette recorder, but allowed data to be transferred at much higher speeds.

All these innovations, chronicled here in logical order, rather than in their order of appearance, contributed to the success of the microcomputer. It is salutary to reflect that serious consideration was being given to the use of microprocessor systems no earlier than 1974, and that public consciousness of such systems began to be evident in 1978-9. Much has happened since, so much, indeed, that it is rash to predict what may happen in another five years.

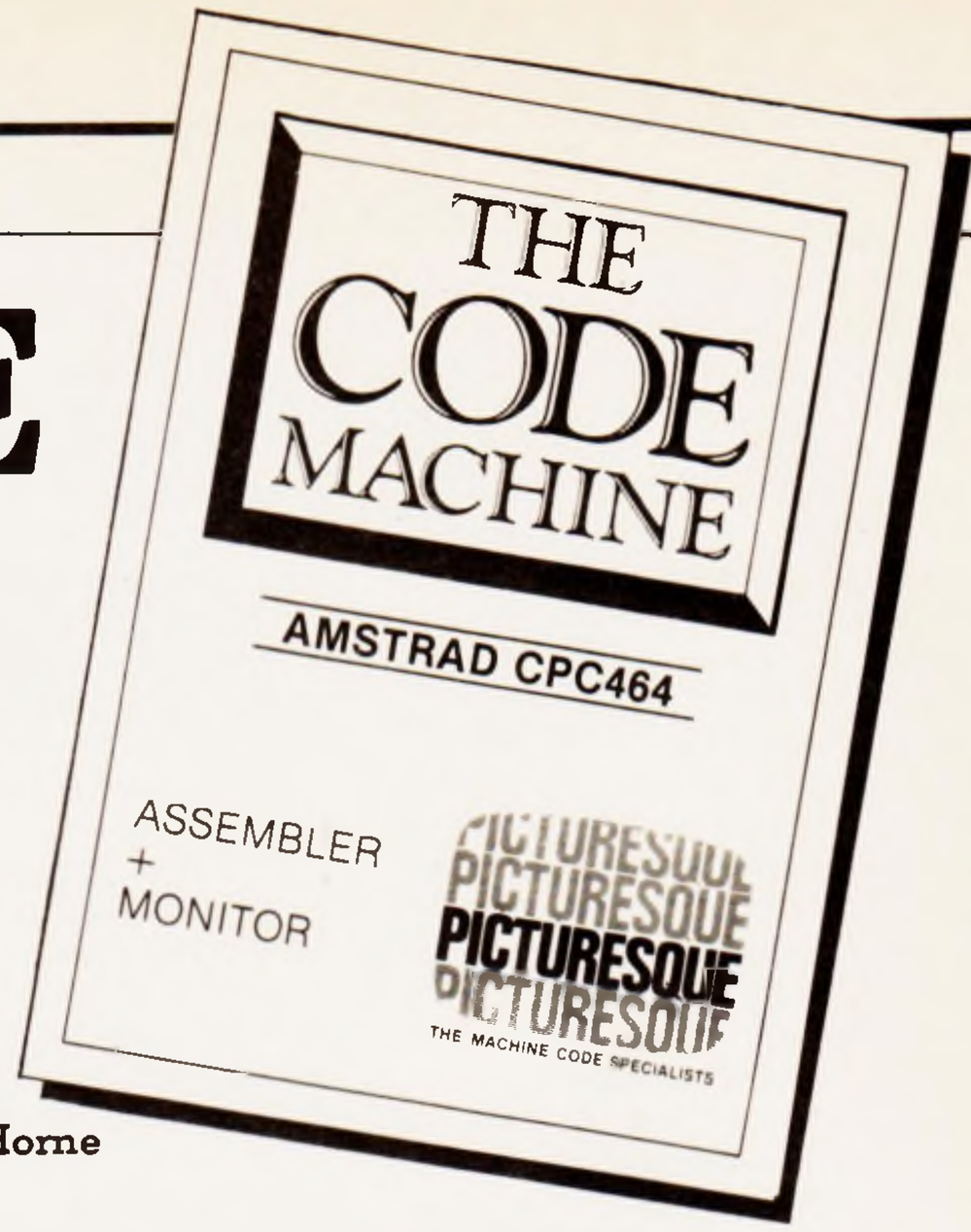
Even when the microprocessor and suitable peripherals for it had appeared, there was no guarantee that effective microcomputers would be produced immediately. Individual manufacturers had differing ideas regarding the facilities which would be needed. Some considered that lower case letters were unimportant, which ruled out effective word processing work. Others used complicated systems of symbols to indicate the position of letters displayed on the screen. Some systems which solved all the problems turned out to be too costly.

Above all, some early systems were far from reliable. Either they had been designed in too much of a hurry, or had been conceived without sufficient thought.

Yet there is no real technical reason why a microcomputer should not be markedly more reliable than a mainframe device. The number of connections involved is far smaller, and that should reduce failures, providing the same standard of design is used. It is only where the search for economy has led to corners being cut that reliability has been suspect.

The true objective of microcomputer design may thus be stated as the achievement of adequate performance at a reasonable cost without reduction in the standards of reliability or operational use. That definition begs a number of questions, but remains valid in general terms. It is clear that some manufacturers have seen rather different priorities, to judge by their products.

THE CODE MACHINE



Bill Horne

'Unlock the power of your CPC464' (it says here). Picturesque's assembler/monitor gets used and abused — but comes out on top.

Some time ago, I reviewed the Picturesque Assembler/Monitor for the Spectrum, reporting that it was much to my taste — so much so that I adopted it as a personal standard. I was therefore delighted to receive a review copy of "The Code Machine", which is the corresponding set of programs for the AMSTRAD CPC464/664, and I have not been disappointed by what I found.

Picturesque are not a large organisation, but their programs have a certain completeness that is sometimes lacking elsewhere. I met the team once, and found them both charming and dedicated, seeking perfection rather than quick profit. Martin Ridout provides the technical content, while Mrs Ridout does the rest, the combination providing thoroughly professional results.

But I must justify this praise by getting down to brass tacks.

THE PROGRAMS

The programs are supplied on tape, one copy of each. They are not protected, in the usual sense, and you are advised to make back-up copies. The cassette case even provides space for the copy! However, it is a matter of extreme simplicity to put the copy on disc.

The Assembler/Editor and the Monitor can be co-resident (sorry — can be in RAM together. Jargon gets to be a habit...) and intelligible memory maps are supplied to show where they are loaded.

This, perhaps, is just as well. For example, the Monitor is loaded as about 8K of code, but some 1.4K of this is used for relocation purposes, and when it has done its job the area which it occupes is overwritten by zeroes. That is useful information.

The instructions for loading and making copies are more than adequate, the processes being eminently simple. When loading the Monitor, you are invited to specify its position in store, but if you have no ideas on the subject a default location is used, putting the program as high as possible in RAM. When the program is loaded, it asks whether you want to make a copy. Copying to disk is then entirely automatic, while copying to tape involves only the usual prompts and key depressions.

Loading the Assembler is much the same, but the optimum location high in store is selected automatically.

The copies differ from the original in that they do not invite the creation of futher copies, which is reasonable. Nor are the copies relocatable, but you can always go back to the original if you want to put the routines in different positions.

THE MONITOR

The Monitor offers an imposing range of options, as shown in Table 1. Entry to a given option is obtained by pressing a single key though in some cases qualifying data has to be added.

A facility for displaying and modifying RAM contents is common enough, but is rarely offered with variants for hexadecimal or ASCII display and input. Block insertion and deletion are less common, but can be useful if you like patching programs. Memory dump and disassembly are more or less mandatory, but most CPC464 versions will not look at ROM. This version does. Conversion in either direction between decimal and hexadecimal is provided.

The breakpoint system, allowing you to run a specified part of a program, and then stop, is backed by a comprehensive register display, which shows the contents of all 13 registers, including the alternatives, with the flag states in F and F' displayed in binary for clarity. If you want more detail, you can run in Trace Mode, which single-steps through a

Table 1: Monitor Commands

A	Move memory contents
B	Set a Breakpoint
C	Continue after a Breakpoint
D	Delete a block from memory
E	Goto Editor/Assembler
F	Fill an area of memory
I	Insert a block of memory
J	Execute code
K	Cancel a Breakpoint
L	Enable/Disable Lower ROM
M	Display/Modify Memory
N	Hex/Decimal Conversion
P	Dump to screen or printer
R	Register Display
S	Search for match
T	Trace Mose (Single Shot)
U	Enable/Disable Upper ROM
Y	To BASIC
Z	Disassemble
£	Select Stream
$	As M, but in ASCII
ESC	Escape to Command Mode

program, providing an extended register display after each instruction is executed. This includes the top five words on the stack, the locations to which BC, DE and HL point and the four following locations, ROM state, interrupt state, and a sequence of nine specified memory location contents. You can really see what is going on.

No less valuable is the precise tabulation of the ways in which the Monitor program modifies the normal operating system action. This is evidence of careful thought planning.

Any moans? Well, I would always like to see a disassembler display a blank line after an absolute jump or return, to mark the end of a code block. This makes the listing so much easier to follow. A personal fad, perhaps, but one that has persisted. Otherwise, no complaints. I may not have tested all the possible combinations of commands and data, but nothing has emerged from the woodwork so far, and I would be surprised by any serious bugs.

Incidentally, the monitor refuses to disassemble itself, which is understandable.

Factsheet: The Code Machine

Price: £19.95 incl. VAT and P&P

Available from: Picturesque
6 Corkscrew Hill
West Wickham
Kent BR4 9BB

THE EDITOR

The Assembler and Editor are combined in AMMAS.

I must admit that the Editor took a little time to become friendly, because it has so many options (Table 2). Each line must be numbered, but there are AUTO and RENUMBER facilities for that, and the numbering has advantages. Each command requires the use of the CTRL key with a character key, though qualifiers may need only one key.

There is automatic tabulation from one format field to the next, which debars use of the Editor for normal text, but makes the entry of source code very simple, This leads to a slight snag, in that use of the 'copy cursor' facility does not always act as expected, the copy having only a single space where the original has several, the extra spaces having been produced by automatic tabulation. However, there is a separate copy routine available which does not suffer this problem.

The format, by the way, allows for four-figure line numbers (up to 9999), six-character labels, four characters for the main instruction mnemonic, and at least 26 characters for the rest. You can work in Mode 1 or Mode 2, and the mode can be changed in midstream if you wish, the instruction also allowing screen colours to be changed.

NEW clears out the source code, but partial clearance, retaining labels, is also possible.

SAVE AND LOAD

You can save Source Code, object code in binary, or — with discs only — object code as a CP/M 'COM' file. Another disc feature comes into play if you save a file with a name that matches that of a file already on the disc, You are given the option to overwrite the earlier file, turn it into a back-up file, or abort the operation. This can be very useful, especially if disc space is running short.

If you are using a particular file name a lot, you can define it as a keystring, so that it is always available at the touch of a key.

Loading is automatic, in that the file type determines the way in which the data is used. You can load into a cleared source file, with labels preserved if you wish, or append further text to existing source code.

Verify, missing in BASIC, is implemented here.

Finally, external commands are accessible. The commands ERA REN and DIR are usable in simpler way than AMSDOS requires, and there are some extra external commands, such as MON to enter the Monitor, and so on.

THE ASSEMBLER

As was said not long ago regarding MAXAM, an assembler is an assembler is an assembler . . . However, there are possible minor variations.

One difference in the present case is that the 'unofficial' Z80 instructions are covered, all 98 of them. These are hidden instructions which were revealed by D. S. Peckett back in February 1981 (in CT, of course!) but never formally acknowledged by Zilog. The Assembler handles these instructions, saving the need to enter them as explicit bytes, but the disassembler gives the HL forms, rather than the IX, IY forms.

The usual directives are implemented, but in a less familiar class is DEFL, which can be used to redefine a label, so that it can have different values in different parts of the program.

PRNT allows control of output to screen or printer, allowing only part of the assembler output to be displayed or printed.

Nothing has yet been said about possible program size. The source code is stored at the top of free space, running downwards, and the object file grows upwards from the bottom of free space. It is therefore difficult to be precise, since the ratio of one to the other is not fixed. The Code Machine manual suggests that 6000 lines of source code might produce 10K of object code, but that would probably be a lot too much for the available space.

No matter, the system will set up the source code as a nuber of sections, and these can be read in and assembled in turn, minimising the total space required. Up to 26 sections can be used, each with 750-1000 lines of source code, so the overall file could run to 20,000 - 26,000 lines, enough to generate some 32K of object code. That should be enough for most purposes.

It is also possible to save the source code as it is produced, in 2K blocks — and you could save on tape with the source code coming from disc, or vice versa.

COMMENT

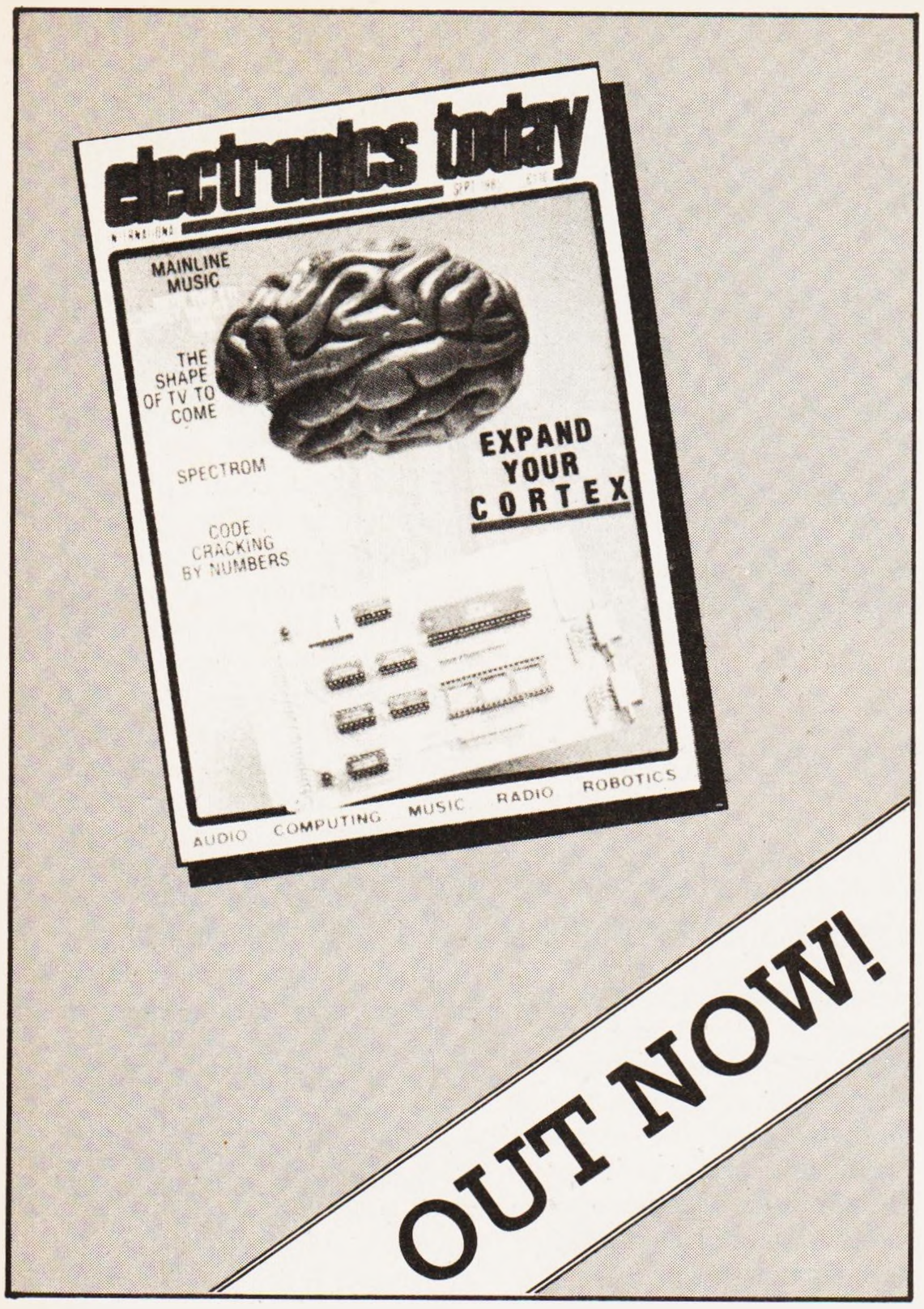
These programs conjure up a vision of someone sitting at the keyboard for hours on end, finding our what further facilities might be needed, and putting in each facility in turn. If the result is in any way less than satisfactory, the very range of facilities ccan be bewildering at first, depite the clearly-written manual, which runs to nearly 70 A5 size pages. When the controls become more familiar, this problem tends to fade away.

When so many half-baked programs are floating around, it is a special pleasure to find offerings like this, hand-crafted and polished until they shine.

Table 2: Editor Commands

A	Assemble	N	New
B	Go to Basic	R	Renumber
C	Copy	S	Save
D	Delete	V	Verify
E	Edit	X	Clear
F	Filename	1	Mode 1
K	Label	2	Mode 2
L	Load	/	List
M	Go to Monitor	+	Auto

The above are used with CTRL pressed



READERS

HELP US
TO HELP YOU:

MENTION

COMPUTING TODAY

WHEN RESPONDING
TO ADVERTISEMENTS

NEW LOW PRICES

WHY PAY MORE

NEW LOW PRICES

QUALITY SS/DD AND DS/QD DISCS IN THEIR OWN LIBRARY BOX

CERTIFIED 100% ERROR FREE AND FULLY GUARANTEED

We are able to offer these prices due to our policy of bulk buying from a major manufacturer and feel that we can offer you the customer a better deal. Why not take advantage of these low prices.

48 HOUR DELIVERY SERVICE
24 Hr. ANSWERING SERVICE

PRICES INCLUDE DELIVERY PER BOX OF 10

PRICES EXCL. VAT	1—4	5—9	10+
SS/DD 48 TPI	12.50	12.00	11.50
DS/QD 96 TPI	17.50	17.00	16.50

SPECIAL OFFER BULK PACKED IN 25's WITH ENVELOPES — PRICES INCLUDE DELIVERY

PRICES EXCL. VAT	25	50	100
SS/DD 48 TPI	27.50	52.50	100.00
DS/QD 96 TPI	40.00	77.50	150.00

VISA

PLEASE SEND PAYMENT OR PHONE TO USE CREDIT CARD
LARGE QUANTITY AND EDUCATION DISCOUNTS GIVEN

Access

Micro Resources Limited

Southfield House, 11 Liverpool Gardens, Worthing, Sussex BN11 1RY
Telephone: Worthing (0903) 213174

DISC DATA

Don Thomasson

Useful but flawed is the conclusion on Don's latest bed-time reading.

Books about popular computers have to tread a precarious path between mere repetition of the data given in the official manual and too enthusiastic an exploration of additional data (I usually err on the latter side!). It is true that a summary of the manual could be useful to someone who is considering purchase of the relevant equipment, but it is usually possible to borrow a manual for that purpose.

In "The Amstrad CPC464 Disc System", Ian Sinclair has allowed an appreciable overlap with information given in the manual, but he has also reached out into less well-known areas, particularly by providing routines which do useful things and spark off useful ideas.

The book begins with a 12-page chapter on discs in general: what they are; why they are better than tape; how the tracks and sectors are arranged; the meaning of formatting and the details of data capacity. Anyone who is reasonably clued-up on the subject will probably skip this section, though it contains a few little nuggets buried in the routine material, and these are worth picking up.

Writing about Amstrad discs is made more difficult by the fact that there are two operating systems, CP/M ann AMSDOS. In the effect, AMSDOS makes use of CP/M, but modifies the command structure and specialises the system to BASIC programs. In some ways the result is convenient, but in other ways it can be frustrating. However, since AMSDOS uses CP/M, rather than vice versa, it might have been helpful to describe CP/M first, then show how AMSDOS uses it.

Instead, the book begins with an emphasis on AMSDOS, making occasional references to CP/M later. This means that the relationship between, say, ERA in CP/M and

ERA in AMSDOS does not emerge as clearly as it might.

It is in this part of the book that the first of the programs appears, a BASIC/machine-code hybrid that displays tape headers. That may seem out of place in a book on disc systems, but it is intended to help in copying tape to disc. Like the other programs in the book, it is backed by detailed explanations, and the underlying ideas are perhaps more valuable than the programs themselves.

In connection with this routine, there is the first hint of something that recurs in the later chapter: a diatribe against the tape protection system. This culminates in a program to copy protected tapes, something which I feel may be unfortunate. Yes, this so-called 'protection' is frustrating. Yes, it can be broken by setting up a mere fifteen bytes of machine code (my method, not the one in the book!). Yes, the Amstrad house journal carries advertisements for programs that will copy protected tapes. Granted all that, it can be risky to 'blow the gaff' in public. However, that is not my worry. . .

Chapter four is devoted to CP/M, covering in a fairly terse way some of the utilities which are only mentioned in the manual. It is true that the value of some of these routines is limited by the fact that they are written for the 8080, not the Z80 (though we have had Z80 versions on our Sorcerer for five years now). Nevertheless, it is frustrating to know they are there but be unable to use them.

Next, there are two chapters on database systems, including a BASIC program of some 143 lines called "Filing Cabinet". I suspect that there is an error in line 100 of this program, a suggestion that lines 60-80 may be used to hold printer setting-up statements. It rather looks as if lines 100-120 should have been quoted. Perhaps there was a renumber. . .

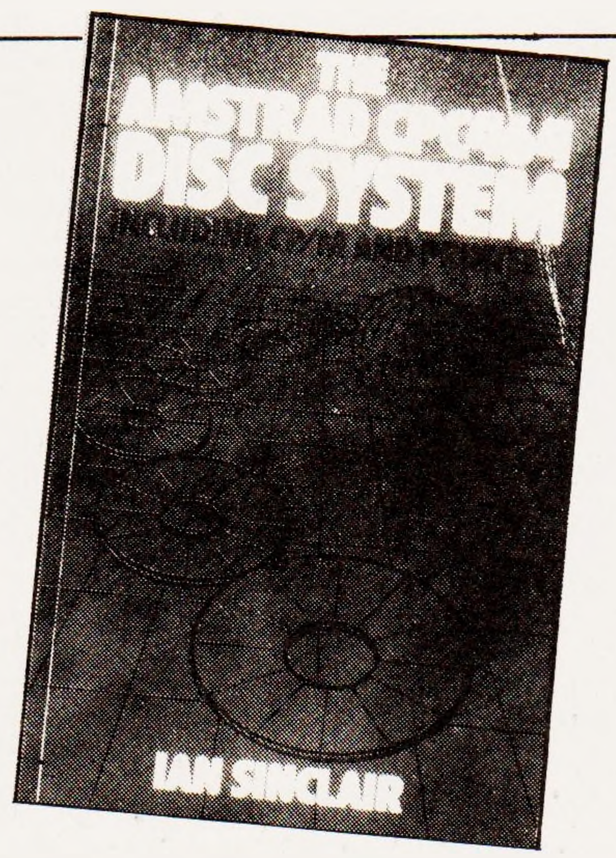
The book then goes off at a tangent, devoting a chapter to printers, On the whole, this is less successful than the rest. It may be of use to someone making a choice between three popular makes, but the information offered is not complete enough to be of real value.

The last chapter offers a number of disc utility programs, and these are very useful indeed, if only for the ideas they inspire. One will display the data in any given disc sector, the next will copy tapes — protected or not — there is a screen-to-disc facility, a memory dump and a disc editor. A veritable playground for the experimenter.

Finally, it must be said that the book is not without its bugs. In explaining why the disc drive must be switched on first, the key point is missed completely: if the drive system is not powered during initial reset, it cannot set up its command words, claim RAM, and otherwise prepare for action. If the main unit has been switched on first, the error can be cancelled out by repeating reset, pressing ESC with Shift and Control depressed. There is no need to switch off and start again.

Again, it is suggested that double line throw with Epson printers can be cured by halving the distance of the throw. That could have bad side effects. The problem arises because the CPC464 earths line 14 of the printer cable. Put a slip of drafting tape over the relevant contact, and the earth is removed. (And we hear of people 'hacking their printer cable about' to achieve the same result.)

The last bug noted is curious. In the disc copy program, machine code is loaded from 0145 to 016D, and the user is then instructed to call NEW. Goodbye machine code, since NEW erases RAM from 0040 to HIMEM, unless (AE7F/80) has been changed. It certainly does on my machine. . .



The Amstrad CPC464 Disc System, by Ian Sinclair, 00p/£7.95 paperback (William Collins & Co Ltd, 8 Grafton Street, London W1X 3LA).

VERDICT

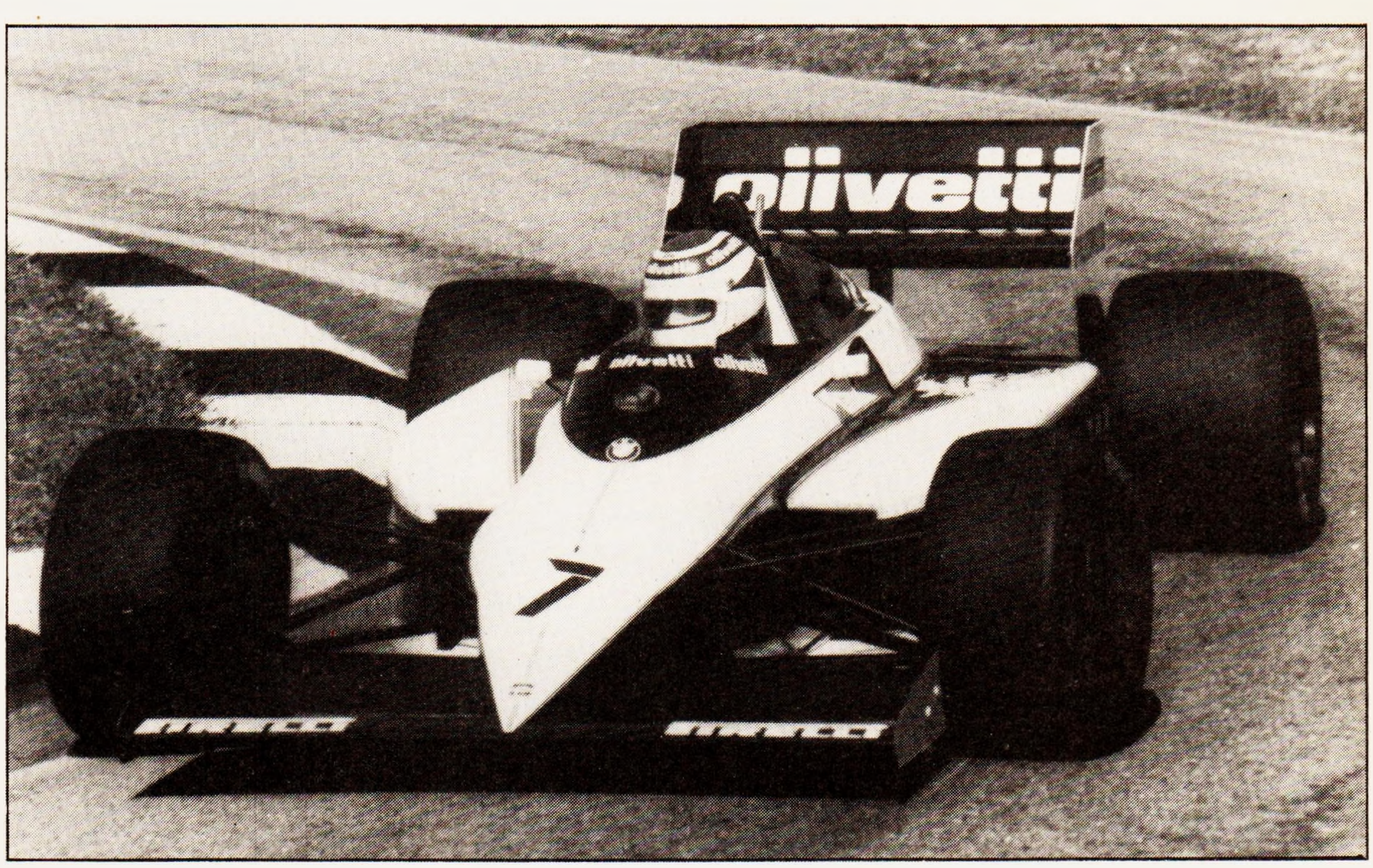
But bugs are bugs, and none of us can avoid them completely. On the whole, the book is useful. It has given me some ideas which look worth developing, and that is always welcome.

To end on a personal note, long-term readers may wonder what has happened to our beloved Sorcerers. Well, mine has left home with my younger son, who will no doubt continue to put it to good use — if his brand-new wife permits! The other Sorcerer continues to serve my elder son, but now that I have re-equipped myself with an Amstrad it may not be long before the younger generation follows suit.

COMPUTERS TAKE TO THE TRACK

Don Thomasson

Computers are becoming an increasingly important part of the motor racing scene, and we examine some of the reasons for this.



Motor racing conjures up different images in the minds of different people. Many see it as essentially a matter of blood and guts, but a closer look reveals that the prime essential is precision, coupled with high standards of engineering.

the keyboard as he sought vainly for strategy that would help his cars to win. Even computers cannot perform miracles.

On the other hand, they are not always used to best advantage. Those massive tabulations of lap times, for example, are informative to the practised eye, but the information can be made a lot clearer by plotting it in graph form. Suppose the first and second cars are coming nearer together: is the second car going faster or the leader slowing down? A graph will show the answer at a glance, whereas the tabulated figures would have to be examined in detail to see what was happening. A graph display might be useful to commentators, especially those who always seem to guess wrong. . .

CIRCUIT DESIGN

We have all heard of computers

being used to design electrical circuits, but the idea of using them to design racing circuits is less familiar. In 1971 the owners of the Watkins Glen circuit in America decided to extend it, and they asked the Aeronautical Laboratory of nearby Cornell University to design a layout that would increase the lap time to 100 seconds. Pole position for the first race on the new track was gained by a time of 102.642 seconds. Not bad at all.

Some of the factors used were eye-openers. Lateral force in flat corners was limited to 1.4g, increasing to 1.74g for banked corners. Braking force was set at 1.1g at 50mph, with a steady increase to 1.5g at 185mph to take aerodynamic downforce into account. The figures for forward acceleration were 0.9g at 50mph, 0.5g at 100mph, 0.2 at 150mph, and zero at the terminal velocity of 185 mph.

More recent figures of this kind are hard to come by, but even those quoted are impressive enough. Note that the acceleration at low speeds implies a forward impulsion almost equal to the weight of the car, and no more than 60% of that weight is normally loaded on to the rear driving wheels. However, acceleration produces a rearward bias of the weight distribution, as demonstrated by the 'wheelies' performed by motor-cyclists. This increases the downforce on the rear wheels, but makes the front wheels 'go light', which may explain why early experiments with four-wheel drive were not too successful. The application of power reduced steering forces, the front of the car sliding outwards in a classic example of 'understeer'.

It seems that four-wheel drive has now been made to work in rally cars, and it may be no coincidence that the use of computers in car design has increased enormously since

The need for precision arises in a number of ways. The drivers must place their cars on the track with accuracy, or they will lose speed. Reflected in lap times, this can be important in two ways: during practice, lap times determine the allocation of places on the starting grid, and the fastest time is rewarded by possession of 'pole position', the favoured side of the front row. During a race there are fastest and record laps to be obtained. They offer no concrete advantage under modern rules, but are much prized by those who are credited with them.

LAP TIMING

It was different back in 1954, when one point counting towards the championship was awarded for the fasted lap. Unfortunately, the British Grand Prix at Silverstone was only timed to the nearest whole second, and the point had to be shared between seven drivers. It was clear that improved methods were needed.

By 1971, timing had become so accurate that pole position was determined by a margin of no more than one millisecond! Since even a grand prix car would move forward no more than 2-3 inches during that time, it might be felt that precision had been overdone a little, but the figures looked impressive on the results sheet.

Recording lap times presents many problems. Taking the time of the starting signal as a base, the moment when the car crosses the timing line must be noted. That may set up a thousand or more entries, which strongly suggest the need for computer assistance, if the necessary data can be generated in suitable form.

A photoelectric sensor and light beam may spring to mind, but that can be defeated if two cars cross the timing line side by side. Nor can such a system identify the cars. The only infallible method requires that the cars should transmit identifying signals, which can be picked up by a buried ground loop.

Such an arrangement may serve for the central timing computer, but timing systems run by individual teams are likely to fall back on cruder methods. However, they only need to keep track of their own cars and those of their immediate rivals.

There was an interesting scene in a television film of the recent Le Mans 24-hour race. In the small hours of the morning, a worried-looking team manager sat staring at the screen of his computer, fingers poised over those earlier experiments, transforming what was once a 'black art' into something rather more efficient and effective.

ENGINE CONTROL

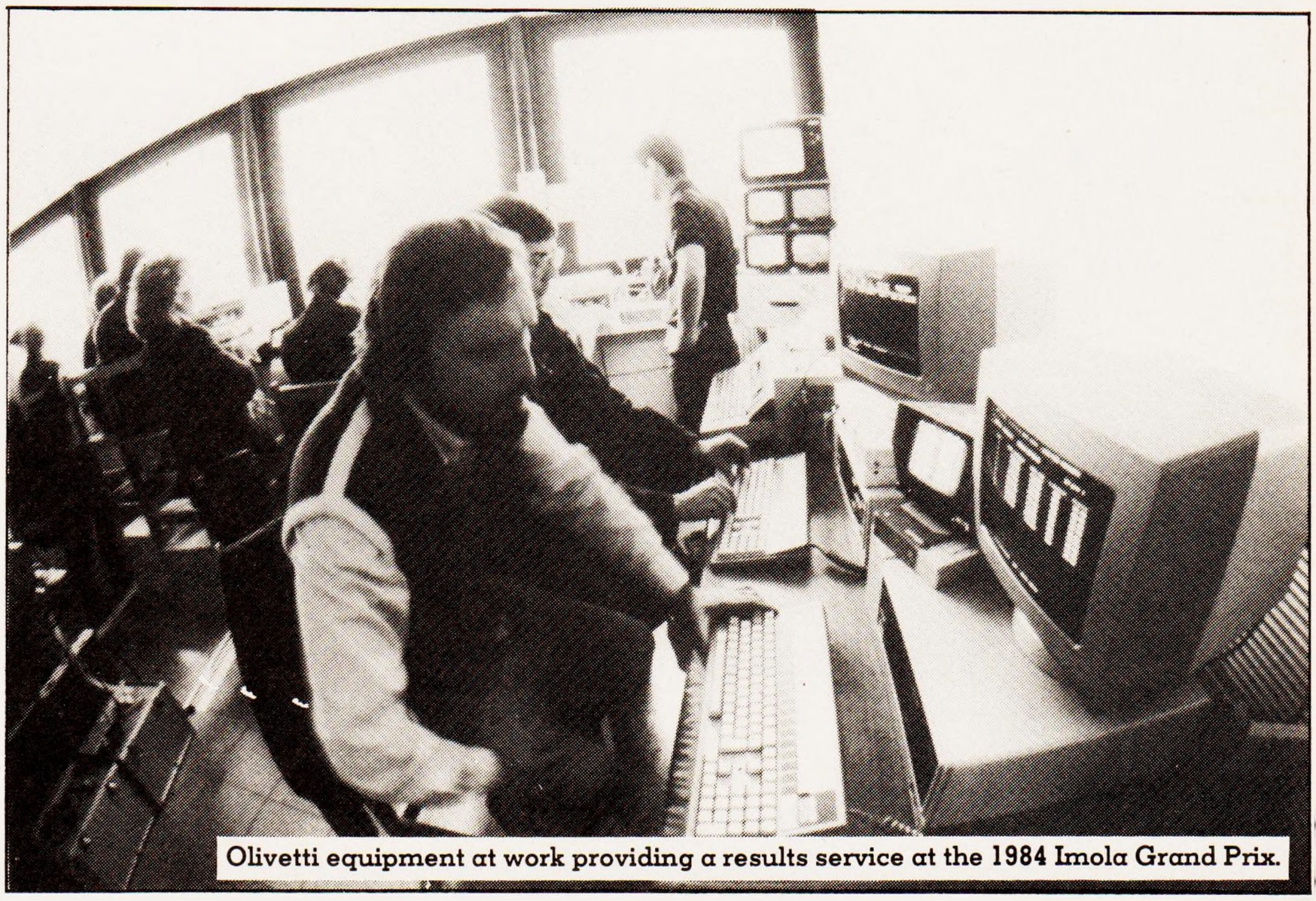
The computers in the pits at Le Mans played their part, but computers in the cars had a more direct effect on the results. Some seven hours after the start, the car in second place developed a misfire. The Motronics computer used to control the engine was rapidly replaced by a spare, the spare being carried in the car in case there was a failure away from the pits. Before long, the car was back, the misfire persisting, and a light on the instrument panel glowing to show that the computer was unhappy.

Hurried diagnostic work traced the cause to a water temperature sensor that was giving high readings, making the computer increase fuel flow in an attempt to cool things down. The solution was simple: Disconnect sensor and cross fingers.

Imagine having to diagnose a fault like that in the vicinity of a hot engine at ten o'clock at night in a dimly lit pit with precious minutes ticking away, and you might wonder if this is an instance of technology being carried too far. Not so. The engine control computer is an essential these days.

It is widely known that the basic petrol engine is far from efficient. The efficiency can be increased by simple mechanical devices such as automatic advance and retard and precise mixture control, but a point is reached where such devices are no longer good enough. There are too many relevant factors to be taken into account, especially where turbocharging is involved.

The need for efficiency, as well as pure power, has been stressed by the introduction of rules limiting the amount of fuel that can be used during a race. Some cars carry displays which tell the driver how much fuel he has left, but that has not prevented a number of cases where a car in contention has run dry in the last few laps. The display may have been at fault, but it is more likely that it was ignored. Having driven brilliantly to get into the lead, few drivers would be prepared to ease off and conserve fuel if that meant fall-



Olivetti equipment at work providing a results service at the 1984 Imola Grand Prix.

ing back down the field. The best answer is to ensure that the fuel is used economically enough to last out.

Since those who report on motor racing are not usually computer experts, information on engine control systems is sparse, but a quote from Ford executive Walter Hayes is revealing:

> "What we are aiming to do is to put the thinking in the engine, where it belongs, and free the driver so he can concentrate in a more single-minded fashion on his driving."

Fine. But what does the driver say if the 'thinking engine' gets too independent, and ignores th instructions given to it? Entering a bend, the driver calls for more power to swing the tail out, but the engine decides that too much fuel would be used. Crunch.

As with all devices intended to simplify human tasks, the engine control computer can create new problems, partly by reducing the degree of control available to the driver.

DATA COLLECTION

The way was paved for the introduction of engine control computers by the earlier use of data collection systems carried in the car and linked by radio to recording and display gear in the pits. The inside of a racing car could scarcely be described as a benign environment, and the lessons learned from data collection installations was useful in deciding how to make computers stand the racket.

Of necessity, the systems were to some extent tailor-made to suit a particular car, but extensive use was made of standardised modules. This is a comparatively little-known branch of computer electronics, but one that has a wide and varied potential market not yet fully exploited. Work in this field requires much the same kind of mental approach as that involved in amateur electronics, coupled with a sound knowledge of engineering in general. Users like motor racing teams can afford to pour a great deal of money into such work, and there is plenty of scope for useful development.

In particular, the use of computers in data analysis has not been fully explored. Manual analysis is tedious and error-prone, where a computer can be patient and accurate. Before this can be put to best use, however, there must be at least one person involved with the necessary knowledge and experience of all the disciplines concerned. Such people are not easy to find.

SUSPENSION DESIGN

It is fitting to end by looking at a system which exemplifies the possibilities mentioned above.

The suspension system of racing cars commonly use a pair of triangular 'wishbones' pivoted to the main body and linked at their outer ends by an 'upright' which supports the axle and wheel. The upright can rotate about its long axis, and this rotation is controlled either by a steering arm or a fixed link. The arrangement offers great flexibility, and slight changes in the dimensions of the various components can transform the handling of a car.

Working out the effect of such changes manually can entail the creation of a whole series of drawings showing the movement of the suspension elements in three dimensions. The new system does away with all this drudgery. Supplied with the relevant dimensions, it will work out the consequent changes in camber, steer, roll centre position and so on, displaying the results on a VDU screen. There are some limitations, such as an inability to take rubber-bushed joints into account, but they are of limited importance.

The system works on a 'Turkey' basis, with the program in ROM so that it is available at switch-on. Only the combined computer and keyboard unit is supplied. That works with a black and white television set for the display, and a printer may be added for hard copy records.

This interesting device, which among other things has to solve the difficult problem of working out the possible configurations of a four-link structure, costs £620. That seems incredibly cheap, until you notice that the display looks remarkably like that of a ZX81!

For those who wish to discover more, the system is supplied by Andreason Racing, The Moorside, Winnal Trading Estate Winchester. Tel: (0962) 60755.

Nelson Piquet's Olivetti Brabham-BMW.

BUSINESS COMPUTERS
Apricot F1E £637 (£614) £658. Apricot F1 £894 (£870) £933. Epson PX8 £900 (£872) £892. Commodore PC10 £1595 (£1564) £1664. Commodore PC20 £2573 (£2485) £2685. Sanyo MBC 775 £1920 (£1899) £1999. Cannon A200C £1609 (£1586) £1686. Sanyo MBC550 £723 (£699) £799. Sanyo MBC550-2 £975 (£939) £1039. Sanyo MBC555-2 £1343 (£1322) £1422.

ORIC AND SINCLAIR COMPUTERS
MCP40 Oric printer/plotter £109 (£110) £122. Sinclair pocket TV £97 (£95) £101. Sinclair QL Computer £374 (£365) £386. QL Floppy disc interface £107 (£103) £109. 3.5" disc drive to suit this interface £177 (£176) £196. Sinclair Spectrum Plus Computer 48k £123 (£127) £147. Original 48k Sinclair Spectrum Computer £89 (£95) £116. Kit to upgrade the spectrum to Spectrum Plus £30 (£30) £40. Microdrive £49 (£50) £60. RS232 interface 1 £49 (£50) £60. Special offer:- Microdrive + Interface 1 + 4 cartridges £97 (£99) £107. Blank microdrive cartridges £2.50 (£3) £4. Spectrum floppy disc interface (See Cumana disc section for suitable disc drives) £97 (£89) £99. Interface 2 £20-£40 (£20) £24. 32k memory upgrade kit for 16k Spectrum (issue 2 and 3 only) £31 (£28) £30. Spectrum Centronics printer interface £46 (£42) £47. ZX printer has been replaced by the Alphacom 32 £61 (£59) £72. ZX81 computer £29 (£29) £39.

COMMODORE COMPUTERS
Commodre 64 £161 (£159) £189. Convertor to allow most ordinary mono cassette recorders to be used with the Vic 20 and the Commodore 64 £9.78 (£9) £11. Commodre cassette recorder £43 (£44) £50. Centronics printer interface for Vic 20 and the Commodore 64 £45 (£41) £46. Disc drive £191 (£186) £217.

AMSTRAD, ATARI ENTERPRISE AND MSX COMPUTERS
Amstrad 464 Colour £342 (£348) £388. Amstrad 464 Green £232 (£247) £287. Amstrad 664 Colour £439 (£431) £481. Amstrad 664 Green £331 (£332) £382. Atari 130XE computer £158 (£163) £183. Atari 520ST computer with 3.5" disc drive, mouse, monitor and software £675 (£670) £730. Atari 800XL computer + recorder £120 (£123) £143. Atari 800XL computer + disc drive £229 (£230) £260. Atari data recorder £34 (£37) £47. Atari disc drive £172 (£171) £191. Atari 1020 printer £93 (£99) £115. Enterprise 64 computer £172 (£170) £190. Enterprise 128 £223 (£229) £249. Goldstar MSX £138 (£138) £158.

ACORN COMPUTERS
Acorn Electron £119 (£119) £139. BBC Model B with free speech i/f £345 (£333) £373. New 64k BBC Model B Plus with doub density disc interface £497 (£484) £514. Acorn disc i/f + DNFS £97 (£95) £100. See below for suitable disc drives. Colour monitor £188 (£228) £268.

CUMANA DISC DRIVES
To suit disc interfaces of Sinclair Spectrum and BBC B. Single:- 40 track single sided £117 (£120) £150, 40 tr double sided £149 (£149) £179, 80trs ds £166 (£166) £196. Dual:- 40tr ss £209 (£211) £251, 40 tr ds £285 (£283) £323, 80tr ds £307 (£304) £344.

PRINTERS
New Epson LX80 £249 (£249) £282. Tractor for LX80 £25 (£33) £53. MCP40 4 Colour printer/plotter £109 (£110) £122. Brother HR5 £148 (£152) £184. Brother M1009 £201 (£203) £234. Shinwa CTI CPA80 £218 (£222) £258, Cannon PW1080A £309 (£306) £356. Brother EP22 £135 (£124) £144. Brother EP44 £212 (£208) £228.

SWANLEY ELECTRONICS
Dept CT, 32 Goldsel Road, Swanley, Kent BR8 8EZ, England.
TEL: Swanley (0322) 64851

Official orders welcome. All prices are inclusive. UK prices are shown first and include post and VAT. The second price in brackets is for export customers in Europe and includes insured air mail postage. The third price is for export customers outside Europe (include Australia etc) and includes insured airmail postage.

BLANK DISKS

5¼in blank discs (40 track)
Prices per box of ten

	single-sided/ double-density	double-sided/ double-density
Scotch 3M	£14.95	£19.95
Dysan	£16.95	£24.95
Memorex	£12.95	£16.95

(unlabelled with free case)

S-J-B SUPER SAVERS!!!

50 Memorex SS/DD disks supplied
in a perspex storage box.
only £59.95

50 Memorex DS/DD disks supplied
in a perspex storage box
only £74.95

We also supply 3" and 3½" disks

FURTHER DISCOUNTS ON BULK ORDERS

CREDIT ACCOUNTS AVAILABLE

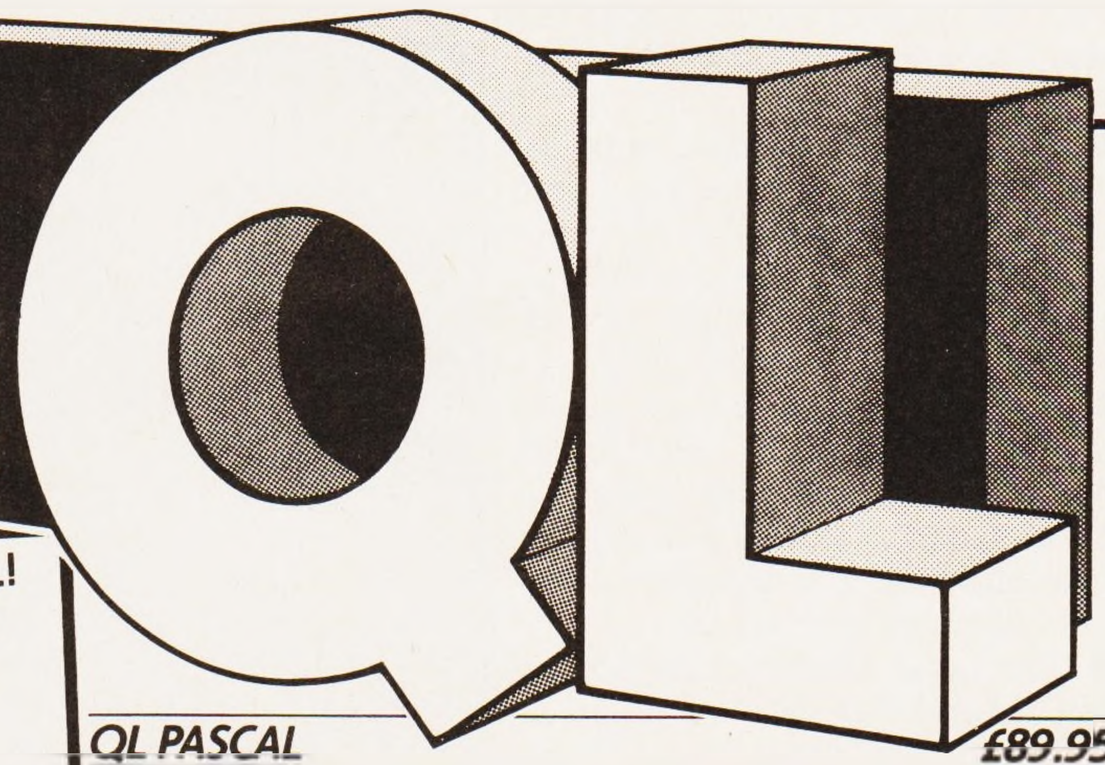
FREE FAST DELIVERY ON ALL ORDERS

ALL PRICES INCLUDE VAT

Please send cheques/PO's to:

S-J-B DISK SUPPLIES
DEPT CT 11 OUNDLE DRIVE,
WOLLATON PARK, NOTTINGHAM NG8 1BN
TELEPHONE: 0602 782310

RELEASE THE POWER OF THE QL



QL C £99.95 INC. VAT

New - a professional specification C compiler for the QL! Designed by LATTICE,® QL C is the most powerful and complete C compiler available for the QL.

> Complete Kernighan and Ritchie implementation.
> Full floating point arithmetic.
> Comprehensive library of UNIX and QDOS functions.
> Separate compilation. Linker.
> Powerful data types (inc. structures and arrays).
> Macros and conditional compilation.
> Over 90 detailed error messages.
> Compatible with LATTICE® compiler on IBM PC etc.

QL PASCAL — £89.95 INC. VAT
Fast, single pass Pascal compiler generating native 68000 code. Complete implementation of the ISO 7185 Pascal Standard. The Pascal recommended by Sinclair Research for use with the QL.

ASSEMBLER — £39.95 INC. VAT
High specification macro assembler supporting the full Motorola instruction set. Complete with linker. Extensive range of features. Thousands of copies now in use worldwide.

BCPL — £59.95 INC. VAT
A true compiler. Ideal for systems programming - utilities, games and applications.

LISP — £59.95 INC. VAT
LISP interpreter for exploring "The language of artificial intelligence", with graphics.

Available from W.H. Smith, John Lewis, HMV, Menzies, Boots and other leading retailers or direct from Metacomco.

Every DEVELOPMENT KIT includes Metacomco's popular screen editor and a detailed manual.
All KITs will operate either on a standard QL or on QL's with floppy disks or memory expansion.

METACOMCO
26 Portland Square, Bristol BS2 8RZ.
Tel: Bristol (0272) 428781

QL and QDOS are trade names of Sinclair Research Ltd.

Price includes VAT, postage and packing UK mainland only. Delivery, allow up to 28 days.

Phone today or post this coupon to: Metacomco, 26 Portland Square, Bristol BS2 8RZ.
Please send me: Assembler £39.95 ☐ LISP £59.95 ☐ BCPL £59.95 ☐
QL PASCAL £89.95 ☐ QL C £99.95 ☐ I enclose a cheque for £_____ or debit my
ACCESS/VISA No: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ CARD EXPIRY DATE ____
NAME _____
ADDRESS _____
POSTCODE _____ TEL. NO: _____ SIGNATURE _____

QL PASCAL

Metacomco's Pascal for the Sinclair QL. Reviewed by Garry Marshall.

Pascal is *the* language for systematic program development. It is taught on many computing courses and is widely used for software development. Metacomco have now produced a Pascal development kit for the QL. This extends the range of languages provided by them for the QL by adding Pascal to their Lisp, BCPL and Assembler, which were reviewed in June's *Computing Today*.

The history and merits of Pascal are well known, but it will do no harm to rehearse them briefly again. It was developed in the late 1960s by Niklaus Wirth, who aimed to create a language suitable for both teaching and applying the concepts of programming systematically and with clarity. He also intended that the language should be available and usable on computers of all kinds. That these aims have been met is self-evident from Pascal's widespread use and availability.

The popularity, and the merits, of Pascal stem from the fact that it makes the writing, reading and modifying of programs easier than do the vast majority of programming languages. The ways in which it does this include the following:

- Structured control statements, for decision-making, selection and repetition, are provided so that clear and concise programs can be written with the flow of control proceeding from the top to the bottom.
- Programs can be broken up into smaller independent procedures and functions each with their own private variables, and well-defined input and output. These can be written and tested individually.
- Data types and structures can be defined and then used as required.
- Variables, procedures and functions can be given long names which can be chosen to suggest their purposes, and so make programs easier to understand and to write.
- Recursion, which can provide the most natural and direct way to program an application, is supported.

The widespread acceptance of Pascal has had a number of consequences. One is that there is an international standard for the language which has been promulgated by the International Standards Organisation (ISO). This means that all programs written using a version of Pascal that conforms to the standard will be portable in that they will run on any other such version. Metacomco's QL Pascal conforms to the ISO standard.

Another consequence of Pascal's success is that many of its features have been borrowed for incorporation by other languages. This is particularly true of the dialect of BASIC supported by the QL, known as SuperBASIC. The statements that it possesses include IF . . . THEN . . . ELSE and, for bracketing a sequence of instructions to be executed repeatedly, REPEAT and END REPEAT. It also possesses an EXIT command for exiting from a repeated sequence: placing this immediately after REPEAT creates a the equivalent of a WHILE loop, and placing it immediately before an END REPEAT creates a REPEAT . . . UNTIL loop. That superBASIC possesses these statements makes it more difficult than otherwise for Pascal to show to advantage with the QWL. As a result, this review will concentrate on the areas of Pascal that do not duplicate those of SuperBASIC. By doing this, it is possible to show how Pascal can be used for applications that would be more difficult to implement when using superBASIC and, at the same time, to see if this version of Pascal operates properly and does conform to the international standard.

FACTSHEET: QL Pascal

Price:	£89.95
Available from:	Metacomco 26 Portland Square Bristol BS2 8RZ

BIBLIOFILE: Programming in Pascal

Author:	Grogono
Publisher:	Addison-Wesley
Pages:	420
Published:	1984
Price:	£8.95
Illustrated:	Yes
Hardcover:	No
ISBN:	0-201-12070-4

GETTING STARTED

The development kit consists of an EPROM, two Microdrive cartridges and a manual. The EPROM contains part of the Pascal compiler, and must be inserted in the ROM socket at the back of the QL after (as the manual is at pains to stress) unplugging the QL. With the EPROM in place, powering up gives the usual initial screen but headed by 'QL Pascal' with a version number. After selecting either the monitor or television mode of display, typing 'ROM' causes the EPROM to check itself. It does this, producing the report 'GOOD ROM' or, if you are unlucky, 'BAD ROM', instantaneously. One of the Microdrive cartridges contains the remainder of the Pascal compiler and also contains a code to match that in the EPROM, so providing an effective anti-pirating system that still allows you to make your own back-up copies. The other cartridge contains an editor and the Pascal run-time system. The editor is the same as that supplied with Metacomco's other language development kits. The run-time system contains a linker to link a compiled Pascal program with the Pascal run-time library to give a stand-alone program that can be run independently. It also contains an installation program with which the kit's default mode of operation can be changed, and a file containing extensions to ISO Pascal, including graphics routines, which can be called in when required. The manual will be dealt with later.

The first step in running a Pascal program is to create the program with the editor and to store it in a file on cartridge. This requires the running of the editor which, after placing the cartridge containing it in the left-hand Microdrive, is done by typing:

```
EXEC_W mdv1_ed
```

The editor allows direct entry of text, supports a wide range of screen editing facilities and possesses a range of commands, including one for saving the current file. The manual provides some short programs which can be used while mastering the system.

Having created and saved a file contianing a program, the next step is to leave the editor, with its 'Quit' command, and to run the Pascal compiler, after inserting the cartridge contining it, with:

```
EXEC_W mdv1_pascal
```

Once invoked, the compiler asks a number of questions, including whether you want to create a file containing any compile-time errors and whether you want to make use of the additions to standard Pascal. To begin with, at least, all the questions except the second can be answered by pressing ENTER to give the default response, because the default mode corresponds to the simplest way of using the system. The response to the second query requires a name for the file to

take the compiled program, and this name should begin with the name of the Microdrive on which the file is to be stored and end with the extension REL, so that it might be, for example, mdv2_prog_REL. This is a typically clumsy file name, and stems from the requirements of the QL. The REL extension is standard for files containing reloca table binary code, being used by, among others, CP/M.

After this, the linker can be run from its cartridge by:

EXEC_W mdv1_paslink

The first thing that it does is to ask for the name of the file containing the compiled program, and it must now be typed. Then it will ask for the names of any further binary input files. For a simple program there will be none, and it is sufficient to press ENTER. But when developing a large program, its segments can be compiled independently and then linked together at this stage. It is also possible to link in binary files created with any of the other Metacomco language systems, that is, with their Lisp, BCPL or Assembler. In this way programs can be developed with a mixture of languages using the one that is most suitable for each segment. At all times the linker will continue to ask for further binary files until ENTER is pressed by itself on a request.

Following this, the linker asks for the name of an output file in which to place the linked program. There are two basic ways to respond at this point. Just pressing ENTER will cause the program to be run and to display any output on the screen. Giving a file name causes the linker to place the linked Pascal program complete with the Pascal run-time support library in the file from where it can be run at any time by typing 'EXEC_W' followed by the file name, always remembering that the file name must include the Microdrive identification.

To summarise this, the procedure for running a program is:

1. To create a file containing the program with the editor.
2. To run the compiler to create a second file containing the compiled program.
3. To run the linker to link the translated program, first, with any additional segments and, second, to the support that is needed to make it an independent program.
4. To run the program.

This description is geared to the assumption that we are dealing with a correct program. Even so, it shows that there are several different ways in which to proceed. When developing a program, there are even more options, which will provide help in the various stages of development, including debugging and testing, that are necessary to reach a correct program. The compiler can produce a file containing descriptions of any errors that may occur during compilation, indicating them with error numbers the meanings of which are listed in the manual. The ability to link various segments together allows the program to be developed and tested in a systematic way. Each segment can be tested individually, and then in its communications with other already-developed segments, before it is placed,

```
PROGRAM double(input, output);
CONST space=' ';
VAR oldchar, newchar: CHAR;
    count: INTEGER;
BEGIN
  oldchar:=space;
  count:=0;
  WHILE oldchar<>'ø' DO
  BEGIN
    READ(newchar);
    IF (newchar<>space) AND (newchar=oldchar) THEN
    BEGIN
      WRITELN(newchar, oldchar);
      count:=count+1
    END;
    oldchar:=newchar
  END;
  WRITELN('Number of pairs is ', count)
END.
```

Fig. 1 Program to accept text from the keyboard and look for repeated letters.

```
PROGRAM reverselist
(input, output);
TYPE link=^object;
     object=RECORD
              next:link;
              data: CHAR
            END;
VAR top, p: link;
    c: INTEGER;
BEGIN
  top:=NIL;
  FOR c:=65 TO 70 DO
  BEGIN
    new(p);
    p^.data:=CHR(c);
    p^.next:=top;
    top:=p;
  END;
  p:=top;
  WHILE p<>NIL DO
  BEGIN
    WRITE (p^.data);
    p:=p^.next
  END
END.
PROGRAM double(input, output);
```

Fig. 2 A program to create a linked list.

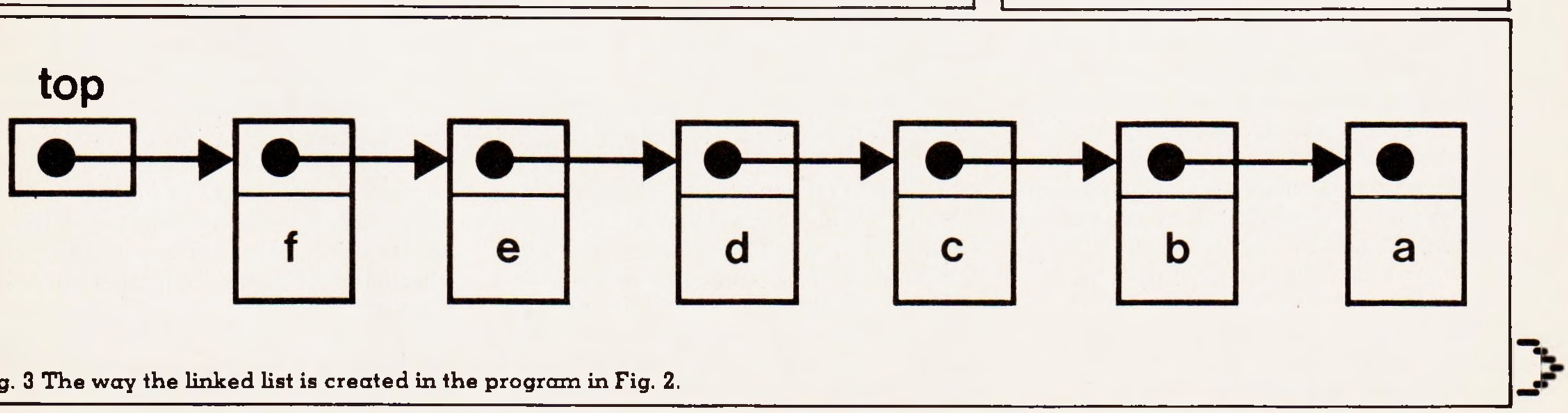


Fig. 3 The way the linked list is created in the program in Fig. 2.

fully developed, in its own file. The development of the next segment can then commence.

Metacomco's QL Pascal development system provides a powerful and truly flexible system for the development of Pascal programs that is wholly within the spirit intended by the language's originator and, further, supports mixed language programming by allowing a program to be built up by mixing binary files created with any of Metacomco's language development systems.

SOME PASCAL PROGRAMS

To demonstrate and to explore the capabilities of Pascal, we will present a few relatively short programs with which the system was tested, first, to see if it coped and, second, to see if it handled ISO standard Pascal. The programs concentrate on the area in which Pascal differs most from BASIC, that is in the ways that it allows its programmers to structure data.

Whereas BASIC supports only integers, reals and character strings, Pascal allows its users to create data structures of almost limitless complexity to meet their needs. It supports the 'RECORD', so allowing any number of items of any type to be grouped together and then to be treated as a single entity. The RECORD is entirely analagous to the record used by a database program, consisting of fields, each of which can be of a standard type or of a type declared by the programmer or, indeed, previously declared RECORDs. It also supports 'POINTER' variables, which can be used to point to other variables. They can be used to create linked lists so that Pascal can be used for list processing. They can also be used to create, for example, a relational database, in which items are associated directly with others. By including a 'POINTER' of the required type in the definition of a 'RECORD', it can be used to point directly to the item associated with its RECORD.

Our programs will involve RECORDs and POINTERs, and will indicate how they can be used with files. Pascal's file handling is not one of its strongest points, but it does provide us with a unifying theme for creating our simple programs that involve data structures by showing that they can be used to implement non-trivial tasks.

The first program (Figure 1) accepts text from the Keyboard and scans it character by character for any occurrences of double letters. It specifically does not report two spaces in succession. When the text has been terminated by pressing '@' the program prints out all the double letters and gives a count of how many there were in the text.

When given the input: 'Pascal is better than many programming languages for writing good programs.'@' it produces the output:

```
tt
mm
oo
Number of pairs is      3
```

The next program (Figure 2) uses pointer variables to create a linked list and then to print it out. The elements of the list are printed in the reverse of the order in which they were handed to the list when it was created. The list created by the program is shown in Figure 3, and may be of help in understanding how the program works. Its basic element is an object consisting of a character and a pointer to the next object in the list. The type of a pointer to an object is denoted by '↑object'. Variables of this type can be declared, while new instances can be generated as required so that lists of any length can be created. The program builds up its list starting with the character having ASCII code 65, then places the character with code 66 in front of it and pointing to it, and so on. When the list has been created, printing it gives 'fedcba'.

The program gives an example of a RECORD for structuring data, but the following is probably a more representative example:

```
RECORD
    title,
    author,
    publisher: array[..12] OF CHAR;
    pages,
    date: INTEGER;
    price: REAL;
    illustrated,
    hardcover: BOOLEAN;
    ISBN: array[1..10] OF CHAR
END
```

This RECORD definition clearly corresponds to the filing card design shown in the second illustrations which could, in turn, be used as the basis for designing a record for entry in a database. By including this definition in a program, all the details of a book can be grouped together and treated as a single entity. Files of such RECORDS can then be created and manipulated.

The two programs given here and others involving files of RECORDS were tried on the system, and they all worked exactly as they should have done. No problems with compiling Pascal programs were encountered and no deviations from ISO Pascal were detected.

THE MANUAL

The manual is, by its own admission, a work of reference rather than a tutorial treatment of QL Pascal. As a source of reference it is entitely adequate although its coverage is brief in places, even to the point of terseness. The page breaks could be positio. ed more helpfully: on a couple of occasions I eventually found vital information which was in the final few files of a section and placed at the top of a new left-hand page, although it appeared that the section had come to a natural end at the bottom of the previous page. Although the manual will meet all the needs of the experienced software developer, especially after a little more attention to its presentation, it is not really suitable for the less experienced. They will, of course, need it for the operating instructions that are specific to the development kit, but I think that they will need to supplement it with a good text. For this purpose, I would recommend 'Programming in Pascal' (2nd edition) by Peter Grogono, published by Addison-Wesley.

CONCLUSIONS

The development kit is intended for software developers and I would say that for them, and for any experienced Pascal programmer, it is a powerful and flexible tool for use with the QL. It provides the full range of facilities, and combinations of facilities, required during program development. A beta-test copy was used for this review, so that there were a few rough edges, but they should have been removed by the time that the final product becomes available in June and at a cost of £89.95. I did manage inadvertently to crash the system on a few occasions, but this was when I was learning to use it, and had done something really unlikely and stupid.

The system is, as far as I can tell, a full implementation of ISO standard Pascal. It also provides additions to the standard, including graphics routines, although these were unfortunately not implemented on the beta-test copy.

The use of the same editor as for Metacomco's other languages is only sensible, unifying the way in which their different languages are used. It also contributes to the ease with which mixed language programming can be carried out which is a definite bonus for all the development kits. Mixed language programming allows critical sections of a program to be written in the most suitable fashion, whether for speed of execution or anything else. Also, modules can be transferred from one project to another with ease regardless of the source language.

My only complaint of any consequence concerns the debugging facilities. The compiler is quite helpful in this respect and can display the successive lines in which an error exists during compilation. It also gives an error number which is intended to be of assistance in correcting the error, but the interpretations of these numbers supplied in the manual are often unhelpful, not to say misleading. For compile-time errors, at least with the program line containing the error located, this is not an insuperable problem, but at run-time with no other help it becomes more serious.

The programming enthusiast looking for a version of Pascal for the QL will also find that this system can meet every need. The full range of its facilities will not be needed, at least initially, and a good tutorial text will be essential. But it provides a full implementation of Pascal, with added graphics capabilities, for learning, exploring and applying Pascal.

MAKING MODELS

D E Prior

A powerful technique for modelling the real world with a microcomputer.

To the railway modeller much of the joy in the hobby stems from the painstaking construction of a scene from the realms of the world of the real live railway. Once the lay-out is done, little time at all is usually spent running it. Within hours of having declared the system complete, the enthusiast is almost certain to be discovered up to the armpits in a toy hill-side, once more doing what gives most joy: building — not running — his railway system.

In much the same way the microcomputer enthusiast derives most of the pleasure and satisfaction of the hobby from the challenge of the programming exercise itself. Once the program is running and producing "the goods" and a demonstration made to the uninitiated of this most recent child of the man/machine union, the true computing addict is soon before the screen once more seeking ways to make the latest brainchild quicker, brigher, cleverer.

A great deal of fun, then, must lie in a hobby which combines with the pleasure of a job well done, not the anticlimax of success, but the spur of a milestone passed. Such is microcomputer based modelling and simulation.

Computer program designing and writing for a simulation embodies all the time engrossing characteristics of conventional program production, but with a difference. The computer modeller, as with the programmer, still can't wait to run the program. The reason is not, however, primarily to see if it works; it is because until it does work, the modeller cannot go on to simulate.

Computer based simulation is being used more and more in both commercial and scientific environments to assist in the understanding of problems. One of the reasons for this is the gradual shift in the problem solving philosophy from reductionism to holism.

REDUCTIONISM AND HOLISM

Reductionism is the process of seeking understanding of a problem by breaking it down (reducing it) into a number of smaller, more manageable, individual problems and seeking to understand each one of these separately. This was the basic approach of the nineteenth and early twentieth century scientists. A crude example of a reductionist approach is trying to understand 'life' by chopping up a human being to look for it. The problem is not solvable this way. 'Life' is a property which 'emerges' from the whole organism. The whole human 'alive' is more than the sum of the parts and to seek to understand better the nature of life the organism and the environment must be studied as a whole, complete situation, not in bits and pieces. The realisation of this principle, that problems are more meaningfully comprehended by standing back from them and viewing them in their contexts rather than isolating them from their surroundings and further splitting them up, gave rise to the 'holistic' philosophy or the doctrine of 'wholes'.

The legacy of the reductionists for the computer user is that many of the techniques available for problem solving on a computer are those which give precise answers to very precisely set up, small separate (and hence 'unreal', in terms of the real world) problems. But real world problems are complex and getting more so. The holistic approach is rapidly assuming a position of importance in that it affords a way of at least appreciating something of the mechanisms behind the workings of large, complex systems.

Complexity and a computer are brought together under the guiding hand of a simulation analyst. The reductionist trap is avoided by setting up the problem in the computer in such a way that instead of giving it an algorithm for *calculating* the solution, it is fed a procedure for *seeking* one. No matter how many variables there are, the computer will juggle with them *all* until a point of balance is obtained on either side of the assignement symbol.

THE ARTICLES

This article lays the foundations of microcomputer based modelling and simulation for the lay micro-computer enthusiast. The goal is to present to the competent home micro owner a methodology of modelling and simulation. The objective is to convey the knowledge, skills and techniques needed to build working, testable simulations of situations of one's own choosing.

Now before anyone reaches for their keyboard it must be evident from the foregoing that the art of modelling and simulation requires a willingness to concede certain philosophical issues. The issues are not difficult to grasp, but they do require that the would-be modeller adopts an open minded, unprejudiced, attitude in the thinking and approaches to problems.

A favourite story of mine, orginally recounted by Stafford Beer, concerns the man who was grovelling on his hands and knees one night beneath a lighted street lamp.

A passing policeman approached him and asked: "Good evening, Sir. Might I ask what the trouble is?".

"Certainly, officer", answered the man, "I am looking for my cuff-link".

"Where did you lose it, exactly, sir?", clucked the constable.

"Over there, in those bushes", said the man, pointing some distance away into

the darkness.

The policeman frowned. "Then why sir," he asked, "are you hunting for it here?".

"Because," replied the man, "this is where the light is."

Many modellers are far too ready to seek the model which answers their purpose in the comforting light of established techniques. Too easily are they persuaded to remove this factor or that from their consideration because not to do so would toss the model into the dark, thorny bushes of intractability.

WHAT IS A MODEL?

A model is a representation of an experience capable of being validated and used for further study of the experience represented.

The problem here is that experiences do not run about the world with tags on them saying "look here, I am the experience called a 'clock'", or "I am the experience called an 'inventory system'", or "I am the experience called an 'ecological system'".

Clocks, inventory systems and ecological systems mean different things to different people. The way in which people might represent them is a mixture of both how they see them and for what reason they want to see them.

This is not so much a statement of the obvious as it may seem. Many people who choose to discuss this or that often do so having assumed without question that the thing under consideration has led, and leads, a life independent of mankind. they believe that there is no need to consider talks about 'talking about the experience', because the experience is absolute and has the same meaning to everyone.

That there is a need for these sorts of considerations is firmly underlined in the everyday world by the requirement for negotiations in, say, industrial disputes to open by talking about 'talking about the problem' before they begin to discuss ways of actually tackling the problem.

The first rule, then, in modelling is to ensure as far as possible through discussion that both the modeller and the user (the person for whom the model is being built) are on common ground. However, when the modeller and user are one and the same then great care is needed to ensure that as little subjectivity as possible creeps in to cloud the issues.

The above definition states that a model must permit validation and must open the door for further study of the system represented by the model.

Consider the following:-

A FIRE

A pile of dry sticks is placed upon dry, crumpled paper and the paper ignited. When substantial flames are observed coal is heaped liberally over the pile of wood and the whole is left to burn.

This is a representation, using words, of a familiar experience. Could it be called a model? Does it permit the truth of the statement to be established or admit further study of a fire? Well, the validation step could proceed along traditional lines — repeat the experiment and see what happens — build 'a fire'. The validity then of the statement would be brought into question had, say, the word 'wet' been used in place of 'dry'.

The illustration itself is trivial for the characteristics of wet and dry wood or paper are well known, but the principle is non-trivial. It is usually not easy to demonstrate the truth of the model proposed to study a situation and, more often than not, mere acceptance by all involved is all that is offered by the modeller.

So far as furthering the understanding of the system 'a fire' through the auspices of the statement of the model, the above offers few, if any, possibilities. There are no clearly assigned relationships between clearly discernible elements of the system 'a fire' that might be used in any way to add to the grand sum of knowledge about 'a fire'.

If the above statement is deemed to be a model then it would appear to be a rather poor one.

What, then, about this:-

A FIRE

A rate of burning which is sufficient to meet the current level of combustible air.

It is possible, in the light of the above statement, to make hypotheses about the nature of a fire. For instance, the words convey the impression that a fire is a negative feedback system. If the level of combustible air increases, then the fire's rate of burning increases and in so doing decreases the level of burnable air. As the level of burnable air decreases, then the rate of burning decreases and in so doing effectively increases the level of burnable air since the fire will burn for longer and longer if it continues to burn more and more slowly.

The hypothesis may be tested by carrying out suitable experiments upon a 'real' burning surrounded by real 'combustible air'. Notice the subtle shift in validation emphasis from the model itself to propositions about the model's behaviour.

DIFFERENT MODEL TYPES

Enough has been said to show that successful model construction requires thought as well as practice. Too much philosophy, however, runs the risk of the simulator not seeing any of the wood for the trees.

That one ought to be able to do something with a model is justifiable in the light of common experience; mechanical models can be driven, and in driving them things can be discovered about the 'real thing' they represent. Word pictures generally cannot be driven to find out things about what they describe. However, the word picture is still the first and foremost representation of an experience, and as such must play an important role in the modelling method.

Another kind of model is the 'pictorial' or, to give it its more up-market name, the 'schematic' model. Into this category may be placed diagrams, sketches, maps, etc.

Yet another sort is the 'iconic' model — a faithful replica of the system under investigation, fashioned out of any material which lends itself to the occasion: wood, paper, plastic, and Plasticene are common. Miniature icons and aeroplanes, cars, cooling-towers and pylons are used in wind-tunnel experiments to determine their performance in air streams and hurricanes. At the other end of the scale, icons of molecules are built to aid thought about the spatial and relative disposition of constituent atoms and particles.

MATHEMATICAL MODELS

The level of abstraction in mathematical models is highly sophisticated and the experiences represented are distilled until only their essence remains. For example:

$$Y = mx + c$$

could be a model of the sum of money in Joe's money-box after x months assuming Joe dropped m pounds in every month and that uncle William dropped c pounds in the box to start him off. On the other hand it might be a model linking the rate of errosion of a river bed to the volume of water flowing.

The power with which the mathematician can represent many apparently different problems with the same set of symbols is one of the reasons why a mathematical modeller may make an unimaginative simulator: knowing a set of solution techniques can lead, albeit subconciously perhaps, to the problem being built up in such a way that the method of solution aligns with one of the known techniques. The cuff-link is sought only where the light is.

WHAT IS A SIMULATOR

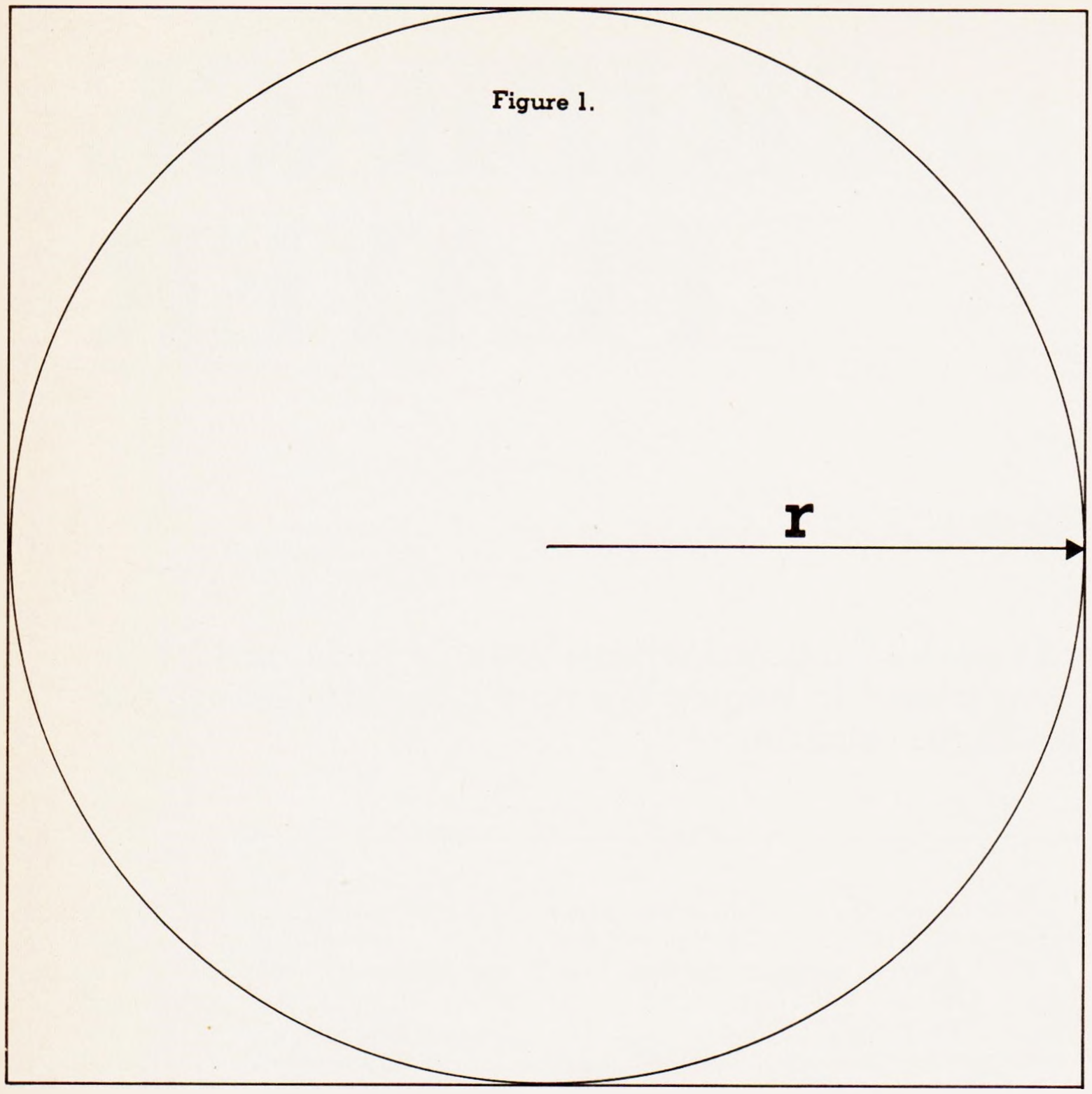
For the purpose of the discussions here —

A simulation is the process of driving a model through time without the intervention of the modeller.

The word 'intervention' is carefully chosen. The modeller may be involved passively in the simulation to the extent of managing the advancement of time from one activity or instant of time to the next, but the modeller may not interfere with the logical order of events as the simulation proceeds.

Notice that the definition forces a distinction between simulations in which the time order of events is of crucial importance and those in which time plays no major role. The former type of simulation is the main concern here for the moment, which is why the definition is couched the way it is. Such simulations as the latter type are characterised, for example, by

Figure 1.



the old school chestnut of a model for pi, now growing very popular with the surge in school computing. The problem is to determine a value for pi starting with the diagram of Fig. 1.

Originally a large version of Fig. 1 would be pinned to a cork board and a blindfolded pupil stood facing it. Armed with a dart, and suitably attended for the sake of safety and the Council's woodwork, the pupil would be allowed to pepper the diagram with holes with 50, 100, 200 or, depending upon the fervour of scientific curiosty, 500 throws. Upon completion of the experiment, the ratio of the counts of the number of holes inside the circle only to the number inside the whole square yields one quarter of the value of pi. The greater the number of holes, the closer the outcome to the accepted value for pi.

$$\frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi r^2}{4r^2}$$

$$= \frac{\pi}{4}$$

$$= \frac{\text{no. holes in circle}}{\text{no. holes in square}}$$

$$\therefore \pi = \frac{4 \times \text{no. holes in circle}}{\text{no. holes in square}}$$

The basis of the experiment is the assumption that the locations of holes are independent of the locations of any previously made holes and are randomly distributed over the face of the square. (So this experiment would probably produce too high a result. Right? — Ed.)

The simulation described above is representative of a static simulation since it is effectively stationary with respect to time. Its attraction lies in its usefulness in the 'new' maths tradition of heuristic learning and problem solving and is, when all practical devices for random event generation have been demonstrated first, an interesting programming exercise and a nice illustration of the use of the random number function available on most micros.

WHY SIMULATE?

Most situations of any real interest are complex, too complex for full representation by a set of easily solvable equations. Mathematical analysis does not always preserve the involved systemic relationships which prompted interest in the system to start with. Simulation, however, can preserve the complexity which the niceities of mathematics seeks to dispense with for the sake of tractability. It all depends upon whether or not the modeller finds the complexity too great to reveal understandable cause and effect relationships or finds the data upon which the simulation is to be based unobtainable.

Simulations are performed to assist answering 'what if . . .?' type questions about a situation. They are carried out when seeking answers from the real world situation would put human life at risk, would be prohibitively expensive, or would destroy the real situation, necessitating expensive and time consuming reconstruction of the system after each experiment.

The main reason why simulation is only slowly being absorbed into the analytic armoury of organisations and corporations is because model construction is such a difficult and skillful task. The construction of a useful model requires far more investment in the valuable resources of time and manpower than most managers believe necessary, or are prepared to concede.

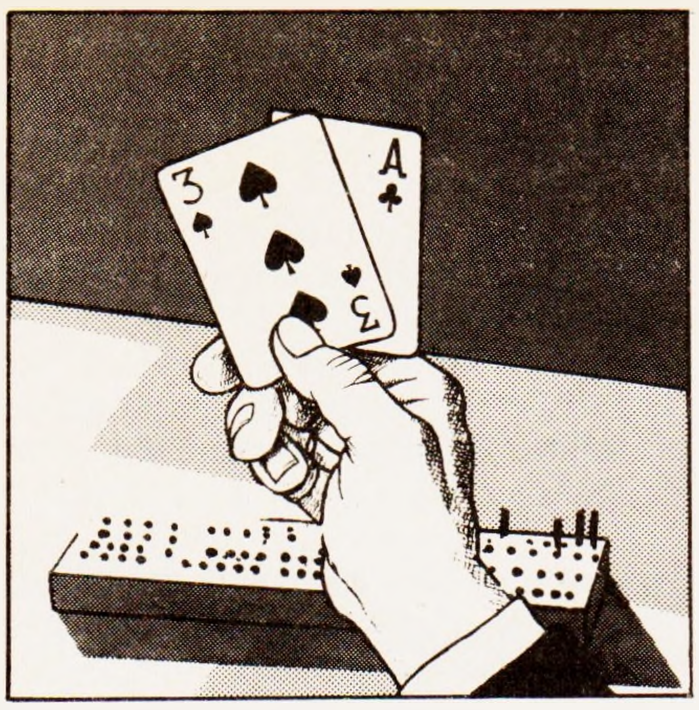
Difficult though the construction of the model may be, it must be emphasised that the problems do not lie in lack of knowledge of a mathematical technique. Computer based simulation is not mathematical modelling, though the two streams of endeavour share a common source in the conceptual modelling phase. No, the problem lies in the superficial grasp humans have of most problems, when the "chips are down" so to speak, and statements of 'fact', upon which lies the understanding of the problem (and hence successful model construction), require justification.

Where does all this leave the lay microsimulator whose eager hand earlier was stayed as it stretched out for the keyboard? With which — words, diagrams, solids or symbols — is one to juggle before arriving at a position whereupon the model of one's choice may be committed to a sliver of silicon? The answer is that one builds upon one's appreciation of the situation using nothing but one's understanding of it together with a very easily learnt, but extremely powerful, diagramatic representation of the problem; in short, one works with the intersection of the first two, words and diagrams.

Once the model has been constructed in this way, then, and only then, does the competent simulator attempt to express the model in a form suitable for processing on a computer. To do this requires a knowledge of the principles behind both continuous and discrete event-oriented simulations, how to sample using random numbers from data drawn from the real world systems, how to draw off from the running simulation information for use in discovering more about the system represented, and how to portray output from the simulation for maximum benefit.

In addition, discussions of how random numbers are generated and how lists of the events which push a simulation on through time may be stored and processed efficiently in the computer often allow more to be discovered about how the computer works than is revealed through the average programming exercise — another good reason for the computer buff to roll up the sleeves and simulate.

The first task, then, is to establish a modelling technique which is capable of being understood by everyone and which, though simple to implement, puts no restrictions on either the imagination or ingenuity in pinning down the problem. Such a technique is called "Casual Loop Diagramming", and this is the subject of a future article.



CRIBBAGE PLAYER

Bill Horne

At the end of part 2, we had reached a point where the discards had been turned up, and table play was about to begin. We now begin to link up the various procedures with the main routine.

Table play is handled by the main routine, assisted by some procedures, because it needs to be able to branch freely to the 'show phase' and win routines. We may as well begin to look at the main routine as a whole.

The routine divides into sections, delineated by loop points. It is shown in Listings 12A and 12B.

PROCB and PROCC are only called at initial start. For a new game, the program loops to line 120, where the variables concerned with scoring are zeroed. PROCD puts up the display frame, and the WIN flag is zeroed.

For each new deal, the program loops to line 150. DL is changed from 1 to 2 or vice versa, being the dealer identity. Note that this happens after the cut for deal, which therefore has to set up the 'wrong' value. PL, identifying the player not dealing, is set to 2 if DL is 1 and to 1 if DL is 2.

The number of cards held by the player (C1) is set ot 6, the number held by the computer (C2) is set to 4, anticipating the discard, and the number of table cards (C8), the GO flag and the table points total (PT) are zeroed.

It may be as well to pause a moment here to talk about the GO flag. At the start of a deal, or when the table cards are cleared, GF=0. In this condition the laying of cards alternates between the players. If the computer 'calls GO', being unable to lay a valid card, GF=1, while if the player calls 'GO' GF=2. PL=GF then calls on the player who has not called GO to take the next turn.

If both players have called GO, or are out of cards, GF=1 becomes GF=3, and GF=2 becomes GF=4. PL=GF−2 then signals which player is entitled to a score for 'a GO', being the second to call GO, or — if all cards have been laid — can claim the score for 'LAST CARD'. The use of GF is ingenious and worth detailed study.

Returning to the main routine, the dealer identity is displayed on the top line, and PROCF is called to perform the deal. Then PROCG is called with SK=1, J5=6, to sort hand 1 (the player's) holding six cards. The message area is cleared, then PROCI displays the cards, PROCJ handles the computer discard, PROCK handles the player discard, and PROCQ turns up the 'starter'.

We have now reached the next loop point, entered when table play has reached a point where neither player can continue, though there are still cards to lay. PROCS registers table total, and then PROCT is called if it is the player's turn, or PROCU if it is the computer's turn. These will be examined later. When they return, the tricky business of scoring is begun. It is tricky because laying a card can score points for several reasons, which are expressed in array ZH$:

1	'15'
2	'31'
3	'A PAIR'
4	'A RUN'
5	'3 OF A KIND
6	'A GO'
7	'LAST CARD'
8	'4 OF A KIND'

ZH is the pointer to these strings, and T is the score associated with them. PROCT and PROCU set up scores for runs (ZR) and pairs (ZP), and HT holds the total of such scores. If it is zero, the routine jumps past the relevant action to line 360.

Otherwise, if ZR is non-zero ZH-4, pointing to 'A RUN', and HT=ZR. PROCX is called to register the score, and the pairs situation is then examined. If there are no pair scores, the routine jumps on. Otherwise, ZH points to an appropriate message, and PROCX is caleld with HT=ZP.

The routine form line 360 on is complex enough to justify a flowchart, given in Fig 1. The influence of GF should be noted. There are three possible exits from this maze: back to line 270 for a further card to be laid, to the 'show phase' routine, or to the win routine. The exits to the win routine had to be placed very carefully, since it may be required whenever the score changes.

COMPUTER TABLE PLAY

When the computer has to pick a card to put on the table, it takes a rather simplistic view or something that is really quite complex.

For example, if there is a five on the table and you add a six, you are asking for trouble, because your opponent can add a four — if he has one — and score 2 for fifteen and 3 for a run. You can risk that, however, if he has previously shown an inability to put down a card of value less than five . . .

PROCU, in listing 13A and 13B, deals with the problem as best it can.

If GF=1 then GF=3, and the routine returns. It has already called GO.

If C2=0, the computer has no cards left, and if GF=0 it reports the fact, sets GF=1, and returns.

If C2=1, there is only one card left, so selection would be pointless. Providing it would not take the table total past 31, it is played. If it is too large, and GF=2, GF=4 and the routine drops out, but if the player is still in action the report 'I can't play — Your go" is displayed. If GF=1 then GF=3, otherwise GF=1.

Next, a check is made to discover whether this is the first card to be laid, in which case the best chance is something of value less than 5, so that the second card cannot reach a total of fifteen. A loop looks for a 4, 3, 2 ace in that order, and sets K to point to the card if a suitable one is found. The two nested loops are terminated. (The CPC464 would allow a

jump out of a FOR loop, but we recognise that this might cause trouble on other machines. It does on the BBC computer.)

Failing a card below 5, preference is given to values 10, 6, 7, 8, 9, in that order. If nothing turns up, the cards must all be fives. One of these is played — subject to the 31 limit.

For cards other than the first to be laid, the decision method changes. First, a card giving a total of 15 or 31 is sought, then a card which is a pair with its predecessor, always subject to the limit of 31.

Next, the possibility of creating or extending a run is examined. The cards in the hand are added, one at a time, to those on the table, and PROCWA is called to score the result. If one added card produces a run, it is played.

Finally, the largest valid card available is picked, though a total of 21 is avoided, since it converts too easily to 31. (There are more cards of value ten than any other value, remember.)

When the card to be played has been chosen, it is announced as a message and PROCZ is then called to rearrange the remaining cards in the hand, PROCV displays the card on the table, entering its value in CP(C8), PROCS updates the table total, and PROCW scores the result.

Listing 12a: main routine, BBC.

```
100   PROCB
110   PROCC
120   SC(1)=0:SC(2)=0:B1=0:B2=0:F1=0:F2=0
130   PROCD
140   WIN=0
150   DL=2+(DL=2):PL=2+(DL=2)
160   C1=6:C2=4:C8=0:GF=0:PT=0
170   IF DL=2 THEN D$=" I " ELSE D$="YOU"
180   PRINT TAB(30,1);D$
190   PROCF
200   SK=1:J5=6
210   PROCG
220   PRINT TAB(0,26);SPC(80)
230   PROCI
240   PROCJ
250   PROCK
260   PROCQ
270   PROCS
280   IF PL=1 THEN PROCT ELSE PROCU
290   ZH=0: IF HT=0 THEN 360
300   IF ZR<>0 THEN ZH=4:HT=ZR:PROCX
310   IF ZP=0 THEN 360
320   IF ZP=2 THEN ZH=3
330   IF ZP=6 THEN ZH=5
340   IF ZP=12 THEN ZH=8
350   HT=ZP:PROCX
360   HT=0:ZP=0:ZR=0
370   IF PT=31 THEN HT=2:ZH=2:PROCX:PROCY:GOTO 400
380   IF PT=15 AND ZH<>1 THEN HT=2:ZH=1:PROCX
390   IF GF<>0 THEN 430
400   IF C1+C2<>0 THEN PL=2+(PL=2):GOTO 460
410   IF C8>0 THEN HT=1:ZH=7:PROCX:PROCY
420   IF WIN=0 THEN 500 ELSE 900
430   IF GF>2 THEN 470
440   PL=GF
450   IF (PL=2 AND C2=0) OR (PL=1 AND C1=0) THEN 480
460   IF WIN=0 THEN 270 ELSE 900
470   PL=GF-2
480   IF C1+C2=0 THEN 410
490   HT=1:ZH=6:PROCX:PROCY:GOTO 400

Add to PROCA

1240  DIM ZH$(8)

1470  FOR J=1 TO 8:READ ZH$(J):NEXT
1480  DATA 15,31,A PAIR,A RUN,3 OF A KIND
1490  DATA A GO,LAST CARD,4 OF A KIND
```

Listing 12b: main routine, Amstrad.

```
100 GOSUB 1000
110 GOSUB 1400
120 SC(1)=0:SC(2)=0:B1=0:B2=0:F1=0:F2=0
130 GOSUB 2300
140 WIN=0
150 DL=2+(DL=2):PL=2+(DL=2)
160 C1=6:C2=4:C8=0:GF=0:PT=0
170 IF DL=2 THEN D$=" I " ELSE D$="YOU"
180 LOCATE 30,1:PRINT D$
190 GOSUB 2500

200 SK=1:J5=6
210 GOSUB 2600
220 CLS #1
230 GOSUB 2700
240 GOSUB 2800
250 GOSUB 3500
260 GOSUB 4100
270 GOSUB 5000
280 IF PL=1 THEN GOSUB 5500 ELSE GOSUB 4900
290 ZH=0: IF HT=0 THEN 360
300 IF ZR<>0 THEN ZH=4:HT=ZR:GOSUB 6200
310 IF ZP=0 THEN 360
320 IF ZP=2 THEN ZH=3
330 IF ZP=6 THEN ZH=5
340 IF ZP=12 THEN ZH=8
350 HT=ZP:GOSUB 6200

360 HT=0:ZP=0:ZR=0
370 IF PT=31 THEN HT=2:ZH=2:GOSUB 6200:GOSUB 6300
GOTO 400
380 IF PT=15 AND ZH<>1 THEN HT=2:ZH=1:GOSUB 6200
390 IF GF<>0 THEN 430
400 IF C1+C2<>0 THEN PL=2+(PL=2):GOTO 460
410 IF C8>0 THEN HT=1:ZH=7:GOSUB 6200:GOSUB 6300
420 IF WIN=0 THEN 500 ELSE 900
430 IF GF>2 THEN 470
440 PL=GF
450 IF (PL=2 AND C2=0)OR(PL=1 AND C1=0) THEN 480
460 IF WIN=0 THEN 270 ELSE 900
470 PL=GF-2
480 IF C1+C2=0 THEN 410
490 HT=1:ZH=6:GOSUB 6200:GOSUB 6300:GOTO 400
```

PLAYER'S TABLE PLAY

PROCT, for the player's table play, is relatively simple. It is shown in Listings 14A, 14B. It bears a strong resemblance to the player discard routine, but is extended to recognise an input of 'GO', and to check for a failure to play a valid card.

When a valid input has been made, PROCP rearranges the player's hand, PROCV displays the card on the table, PROCS updates the table total and PROCW scores the result.

MISCELLANEOUS PROCEDURES

Some procedures have been mentioned in passing, but not listed. We will gather these up before going on to the 'show phase' routine, which is entered directly from the main routine.

First, there is **PROCLL,** a small extension of PROCL which sets ZJ=1 if the Jack of the turn up suit is found in a five card hand.

PROCS displays the table total, and needs no comment.

PROCV puts a card on the table. It must be called after PROCZ, which sets up AA and BB to identify the card, also rearranging the computer's hand.

PROCW checks for pairs in table play, and **PROCWA** checks for runs. The routine drops out with HT, ZP and ZR zeroed if C8=1, which means there is only one card on the table, so there can be neither pairs nor runs.

Otherwise, a FOR loop scans the cards on the table from last towards first, but drops out if the card is not a pair with the next. In successive iterations J7=2, 4, 6, and while the loop continues J7 is added to ZP, giving 2 for a pair, 6 for three of a kind, and 12 for four of a kind.

If this leaves ZP non-zero, or if C8 is less than 3 (three cards being the minimum for a run) the routine drops out with HP=ZP. Otherwise, the table cards are copied into array ST and sorted into order. A check for a run is made, first with all the cards, then less the first card, then less the second, and so on. ZR is set to the number of cards in the run. The routine drops out when only the last three cards have been checked.

PROCX reports the score and the reason for it, calling PROCR to update the score-board.

PROCY clears the table cards from the display, zeroes C8, PT, and GF, then calls PROCS to erase the table total.

PROCZ picks out card K from the computer's hand, storing its identity in AA and BB for PROCV to use. Note that the card remains in the hand, but is not available for table play.

All these are given in Listings 15A, 15B.

Listing 13a: computer table play, BBC.

```
4900  DEFPROCU
4910  IF GF=1 THEN GF=3:ENDPROC
4920  IF C2>0 THEN 4940
4930  IF GF=0 PRINT TAB(2,26);"I'm out of cards.":PROCDEL(250):
      TAB(0,26);SPC(80):GOTO 5170
4940  IF C2>1 THEN 4970
4950  IF PT+VN(HH(2,1,1)  32 THEN K=1:GOTO 5080
4960  GOTO 5130
4970  IF C8>0 THEN 5190
4980 K=0:FOR J2=4 TO 1 STEP -1:FOR J1=1 TO C2
4990  IF VN(HH(2,J1,1))=J2 THEN K=J1:J1=C2:J2=1
5000  NEXT:NEXT:IF K<>0 THEN 5080
5010  FOR J=1 TO C2
5020  IF  HH(2,J1,1))=10 THEN K=J1:J1=C2
5030  NEXT:IF K<>0 THEN 5080
5040 FOR J2=6 TO 9:FOR J1=1 TO C2
5050  IF VN(HH(2,J1,1)=J2 THEN K=J1:J1=C2:J2=9
5060  NEXT:NEXT:IF K<>0 THEN 5080
5070  K=C2
5080  PRINT TAB(2,26);"I play ";S$(HH(2,K,1));S$(HH(2,K,2))
5090  PT=PT+VN(HH(2,K,1))
5100  PROCZ:PROCV:PROCS:PROCW
5110  PROCDEL(250):PRINT TAB(0,26);SPC(80)
5120  ENDPROC
5130  IF GF=2 THEN GF=4: ENDPROC
5140  IF C1=0 THEN 5170
5150  PRINT TAB(2,26);"I can't play - Your go."
5160  PROCDEL(250):PRINT TAB(0,26);SPC(80)
5170  IF GF=1 THEN GF=3: ENDPROC
5180  GF=1:ENDPROC
5190  K=0:FOR J1=1 TO C2
5200  J2=PT+VN(HH,J1,1)):IF J2=15 OR J2=31 THEN K=J1:J1=C2
5210  NEXT:IF K<>0 THEN 5080
5220  FOR J1=1 TO C2
5230  IF HH(2,J1,1)<>CT(C8) THEN 5250
5240  J2=PT+VN(HH(2,J1,1))
5250  IF J2<32 and 31-J2<>VN(CT(C8)) THEN K=J1:J1=C2
5260  NEXT:IF K<>0 THEN 5080
5270  IF C8<2 THEN 5320
5280  C8=C8+1:K=0:FOR J1=1 TO C2:ZR=0:CT(C8)=HH(2,J1,1)
5290  PROCA:IF ZR  0 THEN K=J1
5300  HT=0:IF PT+VN(HH(2,K,1))> 31 THEN K=0
5310  NEXT:C8=C8-1:IF K<>0 THEN 5080
5320  J9=10:IF (31-PT) < 10 THEN J9=31-PT
5330  FOR J1=1 TO C2:FOR J2=J9 TO 1 STEP -1
5340  IF VN(HH(2,J1,1)<>J2 THEN 5380
5350  IF ABS(HH(2,J1,1)-CT(C8))< 3 THEN 5380
5360  J3=PT+VN(HH(2,J1,1))
5370  IF J3 <> 21 AND J3 < 32 THEN K=J1
5380  NEXT:NEXT:IF K<>0 THEN 5080
5390  FOR J2=J9 TO 1 STEP -1: FOR J1=1 TO C2
5400  IF VN(HH(2,J1,1)=J2 THEN K=J1:J1=C2:J2=1
5410  NEXT:NEXT:IF K<>0 THEN 5080
5420  GOTO 5130
```

Listing 13b: computer table play, Amstrad.

```
4900 IF GF=1 THEN GF=3:RETURN
4910 IF C2>0 THEN 4930
4920 IF GF=0 THEN PRINT #1," I'm out of cards.
":M=200:GOSUB 1100
0 5160
4930 IF C2>1 THEN 4960
4940 IF PT+VN(HH(2,1,1))<32 THEN K=1:GOTO 5070
4950 GOTO 5120
4960 IF C8>0 THEN 5180
4970 K=0:FOR J2=4 TO 1 STEP-1:FOR J1=1 TO C2
4980 IF VN(HH(2,J1,1))=J2 THEN K=J1:J1=C2:J2=1
4990 NEXT:NEXT:IF K<>0 THEN 5070
5000 FOR J=1 TO C2
5010 IF VN(HH(2,J1,1))=10 THEN K=J1:J1=C2
5020 NEXT:IF K<>0 THEN 5070
5030 FOR J2=6 TO 9:FOR J1=1 TO C2
5040 IF VN(HH(2,J1,1))=J2 THEN K=J1:J1=C2:J2=9
5050 NEXT:NEXT:IF K<>0 THEN 5070
5060 K=C2
5070 PRINT#1," I play ";N$(HH(2,K,1));S$(HH(2,K,2))
5080 PT=PT+VN(HH(2,K,1))
5090 GOSUB 6400:GOSUB 5900:GOSUB 5800:GOSUB 6000
5100 M=200:GOSUB 1100:CLS #1
5110 RETURN
5120 IF GF=2 THEN GF=4:RETURN
5130 IF C1=0 THEN 5160
5140 PRINT#1," I can't play. Your go."
5150 M=250:GOSUB 1100:CLS #1
5160 IF GF=1 THEN GF=3:RETURN
5170 GF=1:RETURN
5180 K=0:FOR J1=1 TO C2
5190 J2=PT+VN(HH(2,J1,1)):IF J2=15 OR J2=31
THEN K=J1:J1=C2
5200 NEXT:IF K<>0 THEN 5070
5210 FOR J1=1 TO C2
5220 IF HH(2,J1,1)<>CT(C8) THEN 5250
5230 J2=PT+VN(HH(2,J1,1))
5240 IF J2<32 AND 31-J2<>VN(CT(C8)) THEN K=J1:J1=C2
5250 NEXT:IF K<>0 THEN 5070
5260 IF C8<2 THEN 5310
5270 C8=C8+1:K=0:FOR J1=1 TO C2:ZR=0:CT(C8)=HH(2,J1,1)
5280 GOSUB 6060:IF ZR<>0 THEN K=J1:J1=C2
5290 HT=0:IF PT+VN(HH(2,K,1))>31 THEN K=0
5300 NEXT:C8=C8-1:IF K<>0 THEN 5070
5310 J9=10:IF (31-PT)<10 THEN J9=31-PT
5320 FOR J1=1 TO C2:FOR J2=J9 TO 1 STEP -1
5330 IF VN(HH(2,J1,1))<>J2 THEN 5370
5340 IF ABS(HH(2,J1,1)-CT(C8))<3 THEN 5370
5350 J3=PT+VN(HH(2,J1,1))
5360 IF J3<>21 AND J3<32 THEN K=J1:J1=C2:J2=1
5370 NEXT:NEXT:IF K<>0 THEN 5070
5380 FOR J1=1 TO C2:FOR J2=J9 TO 1 STEP -1
5390 IF VN(HH(2,J1,1))=J2 THEN K=J1:J1=C2:J2=1
5400 NEXT:NEXT:IF K<>0 THEN 5070
5410 GOTO 5120
```

Listing 14a: player's table play, BBC.

```
5500  DEFPROCT
5510  IF C1=0 THEN 5570
5520  PRINT TAB(2,26);"Your turn."'"  Type GO if you can't play."
5530  INPUT A$
5540  PRINT TAB(0,26);SPC(80)
5550  IF A$<>"GO" THEN 5590
5560  IF PT + VN(HH(1,1,1)) < 32 THEN 5520
5570  IF GF>0 THEN GF=GF+2:ENDPROC
5580  GF=2:ENDPROC
5590  PROCN:IF EF=1 THEN PRINT TAB(2,26)"You do not hold card ";A$:
      PROCDEL(250):PRINT TAB(0,26);SPC(80):GOTO 5520
5600  IF PT+VN(HH(1,K,1)) <= 31 THEN 5620
5610  PRINT TAB(2,26);"Too large.  Try again.":PROCDEL(250)
     :PRINT TAB(0,26); SPC(80):GOTO 5520
5620  PT=PT+VN(HH(1,K,1))
5630  PROCP:PROCV:PROCS:PROCW
5640  ENDPROC
```

Listing 14b: player's table play, Amstrad.

```
5500 IF C1=0 THEN 5560
5510 PRINT#1," Your play. Type GO if you can't."
5520 GOSUB 3700:A$=UPPER$(A$)
5530 CLS #1
5540 IF A$<>"GO" THEN 5580
5550 IF PT+VN(HH(1,1,1))<32 THEN 5510
5560 GF=GF+2
5570 RETURN
5580 GOSUB 3800
5590 IF EF=1 THEN PRINT#1," You do not hold card
     ";A$:M=200:GOSUB 1100:CLS #1:GOTO 5510
5600 IF PT+VN(HH(1,K,1)) <=31 THEN 5620
5610 PRINT #1," Too large. Try again.":GOTO 5520
5620 PT=PT+VN(HH(1,K,1))
5630 GOSUB 3900:GOSUB 5900:GOSUB 5800:GOSUB 6000
5640 RETURN
```

Listing 15a: miscellaneous, BBC.

```
5700  DEFPROCLL
5710  ZJ=0:FOR A=1 TO 4: IF HN(A)=11 AND HS(A)=HS(5) THEN ZJ=1
5720  NEXT:PROCL ENDPROC

5800  DEFPROCS
5810  D$=STR$(PT):IF PT=0 THEN D$="     "
5820  PRINT TAB(2,18);D$
5830  ENDPROC

5900  DEFPROCV
5910  C8=C8+1:CT(C8)=AA:PROCCB(C8)
5920  PRINT TAB(CP(C8) MOD 40 + 2,CP(C8) DIV 40 +3);SG$(BB)
5930  PRINT TAB(CP(C8) MOD 40 + 1,CP(C8) DIV 40 +1);N$(AA)
5940  ENDPROC

6000  DEFPROCW
6010  HT=0:ZP=0:ZR=0:IF C8=1 THEN ENDPROC
6020  J7=0:FOR J9=C8-1 TO 1 STEP -1:J7=J7+2
6030  IF CT(J9)<>CT(C8) THEN J9=1:GOTO 6050
6040  ZP=ZP+J7
6050  NEXT:IF ZP > 0 OR C8 < 3 THEN 6140
6060  PROCWA:ENDPROC

6100  DEFPROCWA
6110  FOR J9=C8 TO 3 STEP -1:FOR J8=1 TO J9
6120  ST(J8)=CT(C8-J8+1):NEXT
6130  FOR J8=1 TO J9-1:FOR J7=J8+1 TO J9
6140  IF ST(J7) < ST(J8) THEN S1=ST(J7):ST(J7)=ST(J8):ST(J8)=S1
6150  NEXT:NEXT
6160  ZX=0:FOR J8=1 TO J9-1:IF ST(J8)->ST(J8+1)-1 THEN ZX=1
6170  NEXT:IF ZX=0 THEN ZR=J9:J9=3
6180  NEXT
6190  HT = ZP+ZR:ENDPROC
6200  DEFPROCX
6210  PRINT TAB(2,26);W$(PL);" score ";HT;" for ";ZH$(ZH)
6220  PROCDEL(300):SP=PL:SC=HT:PRINT TAB(0,26);SPC(80)
6230  PROCR:IF PT<>15 THEN HT=0
6240  ENDPROC

6300  DEFPROCY
6310  FOR J7=1 TO C8
6320  PROCCD(J7):NEXT
6330  C8=0:PT=0:GF=0
6340  PROCS:ENDPROC

6400  DEFPROCZ
6410  AA=HH(2,K,1):BB=HH(2,K,2)
6420  HH(2,K,1)=HH(2,C2,1):HH(2,K,2)=HH(2,C2,2)
6430  HH(2,C2,1)=AA:HH(2,C2,2)=BB
6440  C2=C2-1:PROCCD(16+C2)
6450  ENDPROC
```

Listing 15b, miscellaneous, Amstrad.

```
5700 ZJ=0:FOR A=1 TO 4
5710 IF HN(A)=11 AND HS(A)=HS(5) THEN ZJ=1
5720 NEXT:GOSUB 3100:RETURN

5800 D$=STR$(PT):IF PT=0 THEN D$="     "
5810 LOCATE 2,18:PRINT D$
5820 RETURN
5900 C8=C8+1:CT(C8)=AA:N=C8:GOSUB 1800
5910 SUIT=BB:DEN=AA:GOSUB 2100
5920 RETURN

6000 HT=0:ZP=0:ZR=0
6010 IF C8=1 THEN RETURN
6020 J7=0:FOR J9=C8-1 TO 1 STEP -1:J7=J7+2
6030 IF CT(J9)<>CT(C8) THEN J9=1:GOTO 6050
6040 ZP=ZP+J7
6050 NEXT:IF ZP>0 OR C8<3 THEN 6140
6060 FOR J9=C8 TO 3 STEP -1:FOR J8=1 TO J9
6070 ST(J8)=CT(C8-J8+1):NEXT
6080 FOR J8=1 TO J9-1:FOR J7=J8+1 TO J9
6090 IF ST(J7)<ST(J8) THEN S1=ST(J7):ST(J7)=ST(J8):ST(J8)=S1
6100 NEXT:NEXT
6110 ZX=0:FOR J8=1 TO J9-1:IF ST(J8)<>ST(J8+1)-1 THEN ZX=1
6120 NEXT:IF ZX=0 THEN ZR=J9:J9=3
6130 NEXT
6140 HT=ZP+ZR:RETURN

6200 PRINT#1," ";W$(PL);" score ";HT;" for ";ZH$(ZH)
6220 SP=PL:SC=HT
6230 GOSUB 4300:CLS#1
6240 RETURN

6300 FOR J7=1 TO C8
6310 N=J7:GOSUB 2000:NEXT
6320 C8=0:PT=0:GF=0
6330 GOSUB 5800
6340 RETURN

6400 AA=HH(2,K,1):BB=HH(2,K,2)
6410 HH(2,K,1)=HH(2,C2,1)
6420 HH(2,K,2)=HH(2,C2,2)
6430 HH(2,C2,1)=AA:HH(2,C2,2)=BB
6440 C2=C2-1:N=C2+16:GOSUB 2000
6450 RETURN
```

Listing 16a

```
500   PF=0:IF DL=1 HP=2 ELSE HP=1
510   I5=4:SK=HP;PROCG
520   FOR J=1 TO 4:HN(J)=HH(HP,J,1):HS(J)=HH(HP,J,2)
530   FOR X=15 TO 29:PRINT TAB(0,X);SPC(40):NEXT
540   PF=PF+1:HN(5)=TU(1):HS(5)=TU(2)
550   IF HP=1 THEN D$="YOUR"
560   IF HP=2 THEN D$="MY  "
570   IF HP=3 THEN D$="CRIB"
580   PRINT TAB(2,11):D$
590   FOR IJ=1 TO 4:PROCCB(IJ+15)
600   PRINT TAB(CP(IJ+15) MOD 40 +2,CP(IJ+15) DIV 40 + 3);SG$
      (HH(HP,IJ,2))
610    PRINT TAB(CP(IJ+15) MOD 40 +1,CP(IJ+15) DIV 40 +1);N$(HH
      (HP,IJ,1))
620   NEXT:PROCLL
630   PRINT TAB(8,16);"This hand scores;"
640   IF ZQ >0 THEN PRINT TAB(10);"Fifteens    ";ZQ
650   IF ZR >0 THEN PRINT TAB(10);"Runs        ";ZR
660   IF ZP >0 THEN PRINT TAB(10);"Pairs,&c    ";ZP
670   IF ZF >0 THEN PRINT TAB(10);"Flush       ";ZF
680   IF ZJ >0 THEN PRINT TAB(10);"One for his Nob"
690   IF HT = 0 THEN PRINT TAB(10);"Nothing" ELSE PRINT TAB(10);
      "Total ";HT
700   IF PF < 3 THEN SC=HT:SP=HP
710   IF HP=3 THEN SC=HT:SP=DL
720   IF HT >0 THEN PROCR
730   IF WIN=1 THEN 900
740   PRINT TAB(10,27);"Press RETURN to continue."
750   INPUT " "X
760   FOR IJ=1 TO 4:PROCCD(IJ+15):NEXT
770   HP=2+(HP=2):IF PF =2 THEN HP=3
780   IF PF< 3 THEN 510
790   FOR X=15 TO 29:PRINT TAB(0,X);SPC(40):NEXT:PROCE
800   GOTO 150

900   IF SP=1 THEN D$="YOU " ELSE D$="I   "
910   FOR K=1 TO 10:PRINT TAB(30,17);"      ":PROCDEL(100)
920   PRINT TAB(30,17);D$;" WIN":PROCDEL(100):NEXT
930   PRINT TAB(9,27);:INPUT "Want another game ";A$
940   IF A$="Y" OR A$="y" THEN CLS:GOTO 120
950 STOP
```

Listing 16b

```
500 PF=0:IF DL=1 THEN HP=2 ELSE HP=1
510 I5=4:SK=HP:GOSUB 2600
520 FOR J=1 TO 4:HN(J)=HH(HP,J,1):HS(J)=HH(HP,J,2):NEXT
530 FOR X=15 TO 24:LOCATE 1,X:PRINT SPC(39):NEXT
540 PF=PF+1:HN(5)=TU(1):HS(5)=TU(2):LOCATE 2,11
550 IF HP=1 THEN D$="YOUR"
560 IF HP=2 THEN D$="MY  "
570 IF HP=3 THEN D$="CRIB"
580 PRINT D$
590 FOR IJ=1 TO 4:N=IJ+15:GOSUB 1800
600 SUIT=HH(HP,IJ,2):DEN=HH(HP,IJ,1)
610 GOSUB 2100
620 NEXT:GOSUB 5700
630 LOCATE 8,16:PRINT"This hand scores;"
640 IF PF<3 THEN SC=HT:SP=HP
650 IF HP=3 THEN SC=HT:SP=DL
660 IF ZQ>0 THEN PRINT TAB(10);"Fifteens    ";ZQ
670 IF ZR>0 THEN PRINT TAB(10);"Runs        ";ZR
680 IF ZP>0 THEN PRINT TAB(10);"Pairs &c    ";ZP
690 IF ZF>0 THEN PRINT TAB(10);"Flush       ";ZF
700 IF ZJ>0 THEN PRINT TAB(10);"And one for his Nob."
710 PRINT:IF HT=0 THEN PRINT TAB(10);"Nothing.
    " ELSE PRINT TAB(10);"Total ";HT
720 IF SC>0 THEN GOSUB 4300
730 IF SC=61 THEN 900
740 INPUT #1,"       Press Enter to continue.";X:CLS #1
750 FOR IJ=1 TO 4:N=IJ+15:GOSUB 2000:NEXT
760 HP=2+(HP=2):IF PF=2 THEN HP=3
770 IF PF<3 THEN 510
780 FOR X=15 TO 24:LOCATE 1,X:PRINT SPC(39):NEXT:GOSUB
    2430
790 IF SC<61 THEN 150
900 IF SP=1 THEN D$="YOU " ELSE D$=" I "
910 FOR K=0 TO 10:LOCATE 30,17:PRINT"
    ":M=100:GOSUB 1100
920 LOCATE 30,17:PRINT D$;"WIN":GOSUB 1100:NEXT
930 INPUT #1,"       Want another game ";A$
940 IF UPPER$(A$)="Y" THEN CLS:GOTO 120
950 GOTO 950
```

THE SHOW PHASE

The show phase routine is tacked on to the end of the main program. Its main loop is executed three times, once for each hand, the non-dealer first, then the dealer, and then the crib hand, scored by the dealer. HP determines which hand will be processed. PF counts iterations.

The loop is then entered. PROCG is called with SK=HP and J5=4 to copy four cards from the hand under review to arrays HN, HS. The turnup card is added. The required screen area is cleared, and the hand is identified. The cards are displayed in positions 16 to 19. PROCLL is then called to check for a jack of the turnup suit, setting ZJ=1 if a suitable jack is found. PROCLL also calls PROCL to make a full score check. The result is displayed, with reasons and total. PROCR is called to update the scoreboard, and the win routine may then be entered. The display is held for inspection until Enter is pressed. It is then erased, and the routine loops to deal with the next hand.

The Win routine follows. It is quite simple, Those who want a firework display are welcome to design one . . .

PROCEDURES

PROCS	5800	Table total
PROCT	5500	Table play, player
PROCU	4900	Table play, computer
PROCV	5900	Card to table
PROCW	6000	Table score, pairs
PROCWA	6060	Table score, runs
PROCX	6200	Score update
PROCY	6300	Clear table cards
PROCZ	6400	Cards to table

SUMMARY

The program is now complete, and when you have it running you may feel like probing the details for possible improvements. In particular, you may feel that the 'decision' routines could be 'beefed up', though Bob Stafford's originals — used without relevant change — are something to marvel at.

There are minor points. It has been suggested that a GOSUB followed immediately by a RETURN is pointless. It could be just a GOTO, but the advantage of that is small, and readability suffers. We have cut out variables using 'I' as far as possible, because they can by confused with 1. No doubt we will see other possible improvements as soon as the text goes to press, but that always happens. . .

Important notice: following complaints regarding the quality of the Amstrad Cribbage listings in the August edition, we have reproduced listings 5b to 11b. We apologise to all readers who were affected by this.

Listing 5b

```
2500 DD=53 FOR A=1 TO 52 PC(A)=A NEXT
2510 FOR A=1 TO 2 FOR B=1 TO 6 GOSUB 1700
2520 HH(A,B,1)=NN HH(A,B,2)=SS NEXT NEXT
2530 GOSUB 1700 TU(1)=NN TU(2)=SS
2540 RETURN

2600 FOR A=1 TO J5-1 FOR B=A+1 TO J5
2610 IF HH(SK,A,1)=HH(SK,B,1) AND(HH(SK,A,2)=HH(SK,B,2))THEN 2640
2620 F=HH(SK,A,1) HH(SK,A,1)=HH(SK,B,1) HH(SK,B,1)=F
2630 F=HH(SK,A,2) HH(SK,A,2)=HH(SK,B,2) HH(SK,B,2)=F
2640 NEXT NEXT RETURN

2700 FOR N=16 TO 21 GOSUB 1800 GOSUB 1900 NEXT
2710 FOR N=9 TO 14 GOSUB 1800
2720 SUIT=HH(1,N-8,2) DEN=HH(1,N-8,1) GOSUB 2100
2730 NEXT
2740 N=15 GOSUB 1800 GOSUB 1900
2750 RETURN
```

Listing 6b

```
2800 PRINT#1," I'm putting cards in the crib"
2810 LS=0 SP=0 ZJ=0 HF=2
2820 FOR IJ=1 TO 2 C=0 FOR I=1 TO 6
2830 IF I=I( THEN 2850
2840 C=C+1 HH(C)=HH(2,I,1) HS(C)=HH(2,I,2)
2850 NEXT GOSUB 3100
2860 IF HT<LS THEN LS=HT SI=I(
2870 N=IJ+15 GOSUB 2000 GOSUB 1900 GOSUB 1900
2880 NEXT
2890 IF SK=0 THEN SK=6
2900 F=1 GOSUB 3400 SI=0
2910 LS=0 FOR IJ=1 TO 5 C=0 FOR I=1 TO 5
2920 IF I=IJ THEN 2940
2930 C=C+1 HH(C)=HH(2,I,1) HS(C)=HH(2,I,2)
2940 NEXT GOSUB 3100
2950 IF HT<LS THEN LS=HT SI=IJ
2960 NEXT IF SK=0 THEN SK=5
2970 F=2 GOSUB 3400
2980 N=21 GOSUB 2000 N=20 GOSUB 2000
2990 CLS #1
3000 RETURN
```

Listing 7b

```
3100 ZF=0 RT=0 FOR A=1 TO 4 RT=RT+HS(A)=HS(1) NEXT
3110 IF RT=4 THEN ZF=4+HS(5)=HS(1)
3120 FOR A=1 TO 4 FOR B=A+1 TO 5 IF HH(A)<>HH(B) THEN 3140
3130 F=HH(A) HH(A)=HH(B) HH(B)=F
3140 NEXT NEXT
3150 ZP=0 ZO=0 ZR=0
3160 IF (HN(2)=HN(1)+1 AND HN(3)=HN(2)+1 AND HN(4)=HN(3)+1 AND HN(5)=HN(4)+
1)THEN ZP=5
3170 IF VN(HN(1))+VN(HN(2))+VN(HN(3))+VN(HN(4))+VN(HN(5))=15 THEN ZO=2
3180 ZZ=0 FOR A=1 TO 3 FOR B=A+1 TO 3 FOR C=B+1 TO 4 FOR D=C+1 TO 5
3190 IF (HN(B)=HN(A)+1 AND HN(C)=HN(B)+1 AND HN(D)=HN(C)+1)AND ZR=0 THEN ZZ
=ZZ+4
3200 IF VN(HN(A))+VN(HN(B))+VN(HN(C))+VN(HN(D))=15 THEN ZO=ZO+2
3210 NEXT NEXT NEXT NEXT
3220 ZR=ZR+ZZ ZZ=0
3230 FOR A=1 TO 3 FOR B=A+1 TO 4 FOR C=B+1 TO 5
3240 IF (HN(B)=HN(A)+1 AND HN(C)=HN(B)+1 AND ZR=0 THEN ZZ=ZZ+3
3250 IF VN(HN(A))+VN(HN(B))+VN(HN(C))=15 THEN ZO=ZO+2
3260 NEXT NEXT NEXT
3270 ZR=ZR+ZZ
3280 FOR A=1 TO 4 FOR B=A+1 TO 5
3290 IF VN(HN(A))+VN(HN(B))=15 THEN ZO=ZO+2
3300 IF HN(A)=HN(B) THEN ZP=ZP+2
3310 NEXT NEXT
3320 HT=ZP+ZO+ZR+ZF+ZJ
3330 RETURN
```

Listing 8b

```
3400 HH(3,F,1)=HH(2,SI,1)
3410 HH(3,F,2)=HH(2,SI,2)
3420 HH(2,SI,1)=HH(2,7-F,1)
3430 HH(2,SI,2)=HH(2,7-F,2)
3440 RETURN
```

Listing 9b

```
3500 FOR J=1 TO 6 PC$(J)=N$(HH(1,J,1))+S$(HH(1,J,2)) NEXT
3510 FOR QP=3 TO 4 IF QP=3 THEN Q$="first" ELSE Q$="second"
3520 PRINT #1," Your ";Q$;" discard to the crib?"
3530 GOSUB 3700 A$=UPPER$(A$) CLS #1
3540 GOSUB 3800 IF EF=0 THEN 3570
3550 PRINT#1,"You do not hold card ";A$
3560 M=150 GOSUB 1100 CLS #1 GOTO 3520
3570 HH(3,QP,1)=HH(1,K,1)
3580 HH(3,QP,2)=HH(1,K,2)
3590 GOSUB 3900 NEXT
3600 RETURN

3700 A$=INKEY$ IF A$=""THEN 3700
3710 B$=INKEY$ IF B$=""THEN 3710
3720 A$=A$+B$ RETURN

3800 EF=1 FOR J2=1 TO C1
3810 IF A$=PC$(J2) THEN K=J2 EF=0
3820 NEXT RETURN

3900 AA=HH(1,K,1) BB=HH(1,K,2)
3910 N=C1+8 GOSUB 3000
3920 IF K=C1 THEN 3990
3930 FOR C=K TO C1-1
3940 HH(1,C,1)=HH(1,C+1,1) HH(1,C,2)=HH(1,C+1,2)
3950 N=C+8 GOSUB 1800
3960 SUIT=HH(1,C,2) DEN=HH(1,C,1) GOSUB 2100
3970 PC$(C)=PC$(C+1) NEXT
3980 HH(1,C1,1)=AA HH(1,C,2)=BB PC$(C1)=""
3990 C1=C1-1 RETURN
```

Listing 10b

```
4100 GOSUB 4200
4110 IF TU(1)<>11 THEN RETURN
4120 FOR I=1 TO 4 GOSUB 4200 M=100 GOSUB 1100 NEXT
4130 SP=0L SC=2
4140 PRINT#1," Jack turned up. ";S$(0L);" score 2 to dealer."
4150 M=200 GOSUB 1100 CLS #1
4160 GOSUB 4300
4170 RETURN

4200 N=15 GOSUB 3000 GOSUB 1800
4210 SUIT=TU(2) DEN=TU(1) GOSUB 2100
4220 RETURN
```

Listing 11b

```
4300 SC(SP)=SC(SP)+SI
4310 IF SP=1 THEN Y=9 ELSE Y=1
4320 LOCATE 15,Y PRINT STR$(SI);SP;
4330 IF SC(SP)<120 THEN SC=61 WIN=1 GOTO 4350
4340 SC=SC(SP) IF SC>60 THEN SC=SC-60
4350 IF SP=1 THEN GOSUB 4400 ELSE GOSUB 4500
4360 RETURN

4400 F3=B2 IF F3=0 THEN M1$=CHR$(144) M2$=CHR$(231) GOSUB 4800 GOSUB 4700
4410 B2=F2 F2=SC F3=SC M1$=CHR$(231) M2$=CHR$(144)
4420 GOSUB 4800 GOSUB 4700 RETURN

4500 F3=B1 IF F3=0 THEN M1$=CHR$(144) M2$=CHR$(231) GOSUB 4600 GOSUB 4700
4510 B1=F1 F1=SC F3=SC M1$=CHR$(231) M2$=CHR$(144)
4520 GOSUB 4600 GOSUB 4700 RETURN

4600 IF F3<31 THEN Y=3 X=F3+2+(F3-1)\5 ELSE Y=4 X=68-F3+(F3-31)\5
4610 RETURN

4700 FOR I=1 TO 3 LOCATE X,Y PRINT M2$
4710 M=30 GOSUB 1100
4720 LOCATE X,Y PRINT M1$
4730 GOSUB 1100
4740 NEXT RETURN

4800 IF F3<31 THEN Y=7 X=F3+2+(F3-1)\5 ELSE Y=6 X=68-F3+(F3-31)\5
4810 RETURN
```

ALGORITHM ANGLES

Bill Horne

Relocatable programs.

Relocatable programs are nothing new, but interest in the subject has been reopened by a characteristic of the AMSTRAD CPC464 and CPC664. When use is made of the extension ROM facility, each ROM fitted can claim an area of RAM for its own use. For example, the MAXAM ROM reserves 260 bytes, and if used on its own moves HIMEM from &AB7F to &AA7B, while the disc system takes 504 bytes, giving HIMEM as A67B. With MAXAM and discs, HIMEM is reduced to A577.

This can be an embarrassment. The normal idea is to put machine code below the original HIMEM, moving HIMEM to a point below the code, so that the machine code program is protected. One approach to this is to use;

```
MEMORY (HIMEM - N)
```

- where N is the number of bytes in the machine code program. This is fine at the first run, but causes trouble if you run again, because HIMEM is then lowered, and it is now lowered still further. This can be avoided by doing a full reset (CTRL/SHIFT/ESCAPE) and reloading the program, but that in itself can cause problems, especially during program development.

On the other hand, setting HIMEM to a specific value may not match up with the original HIMEM for the system in use. this arises with some commercial programs, which are written to run on the basic machine, and thus overlap the area reserved for extension ROMs. Since these ROMs are not used by the program there is no immediate hassle, but if the programs are copied to disc everything is liable to go haywire.

AUTOMATIC RELOCATION

The best option, in terms of simplicity, is to assemble the machine code at run time, but that is not as easy as it sounds. A MAXAM ROM makes it easier, because the source code can then be embodied in the BASIC program, and ORG can be based on the value of HIMEM for the system. It is still necessary to alter the entry call, however. The entry address has to be calculated from the original HIMEM, and the result used as the 'address expression' in the CALL statement. MAXAM can be made to deal with data transfers.

Where the machine code is set up by the BASIC program, using DATA statements as a source, things become rather more tricky. First, the FOR loop used to poke the machine code into store must have its limits adjusted on a basis of HIMEM, so that the program goes into the right place. Then all absolute addresses internal to the program itself must be adjusted.

For the Z80 this affects all LD XX,NN type instructions, where XX indentifies a register and NN is an absolute address; all absolute calls and jumps; and the contents of registers used to define absolute addresses in accessing data. In practice, only a small proportion of the code bytes need to be changed, especially if the program is moved by a multiple of 256 bytes, in which case only the upper byte of the address needs to be altered. This is achieved by using only the upper byte of HIMEM as read:

```
A = HIMEM:B = INT (A/256)
```

However, it must be remembered that the index registers can be useful in this context. But how can the program know where it is located, so that IX or IY can be set up to point to the right data area? What about this:

```
L1  CALL  L2
L2  POP   HL
    LD    DE,DISP
    ADD   HL,DE
    PUSH  HL
    POP   IX
```

The CALL puts the address of L2 on to the stack, and POP HL takes it off again. An appropriate displacement is added, via DE, and the result is passed to IX by way of the stack. Any questions?

We now have IX pointing to a data area which is in a known position relative to the position of the rest of the program, and we can therefore access the data without problems. We might go further, using JP(IX) to create a relative jump, but that would be unnecessary except in large programs, since with small programs relative jumps can usually be used. If they need to span more than 127 locations, use 'staging jumps', the original jump going to a further jump and so on.

ALTERNATIVES

There is an alternative approach for small programs which has some rather useful characteristics.

The CPC464 sets up its main stack area below &C000, and provides room for 128 pushes or return addresses. These are rarely used in full, and it is therefore possible to consider placing a small program from &BF00 upwards. The word 'small' must be emphasised, because the nesting of subroutines in the BASIC interpreter can produce quite a big stack on occasion, but some sixteen locations can be used without risk. They can be used for code or data, allowing at least some addresses to be fixed.

A particular point about using this area is that its contents survive a general reset! This proved to be a boon when a short program at &BF00 was being used in some strictly illegal experimentation, which entailed very frequent resets.

SUMMARY

Making a program relocatable is not a simple matter, unless you can assemble it at run-time, but some of our readers may see that as a challenge. Others may well point out that everything is different with another processor, but we have quoted the Z80 here because we were talking about the AMSTRAD.

MAXAM makes life easier, and the more we use that little beast - this article was typed with its aid - the more we like it. But remember we are talking about the ROM version. The loadable form takes up additional memory, and could make machine code relocation even more necessary...

PROGRAMMING FOR SPEED AND RELIABILITY

Jim McCartney

Diskette systems are economical and have enough storage capacity and speed for many data processing applications. But how do you get the best out of them?

By diskette, I mean floppy disk in most cases, but then some of them aren't floppy any more, and by diskette machines, I mean machines which use removable diskettes as their means of off-line data storage, and read them by head contact with the magnetic surface. In these machines, the read head is moved by electro-mechanical devices, and the heads are in physical contact with the disk, and if there is one thing as certain as death and taxes, it is that mechanical systems wear out. The other feature of diskette systems is that they are a lot slower than flying head hard-disk systems. I am not fond of the term diskette, which sounds like a member of a rather old-fashioned female pop group, but I can't think of anything better.

WEAR AND TEAR

Although the reliability of diskettes and diskette systems has improved greatly over the past few years, a heavily-used system is still prone to mechanical failure. Diskettes themselves fail and are chucked away, and drives develop a bit of slack in the mechanism, and are replaced under maintenance contract. In fact, the huge majority of maintenance calls — after the run-in period — are disk drive faults. This means that all over the country at this moment, hundred, maybe thousands, of accountant, DP people and the like are coping with lost or corrupted data. While this is jolly good for the sales of hard-disk systems in the long run, we might consider how to minimise the agony in the meantime.

As I said above, mechanical systems wear out in use, and it must follow, as doth night the day, that if you use them less they will wear out less frequently. This means that you need to give thought to programming in such a way as to reduce the number of disk accesses, and in particular to random accesses through several files (sequential access causes less head movement). There is another payoff; the whole system works faster, pro rata.

DATA CORRUPTION

Three main causes of data corruption are: faulty disks, dirty heads, and misplaced blocks. The first two are fairly easy to guard against, because they will generally produce unreadable blocks and will show up when you take copies or even sooner, depending on your error-checking in the DOS. The chances are that you can recover this sort of error by cleaning the heads, re-trying the copy, and, if all else fails, copying the good files only. With any luck, your recent data will be on these. If all these fail, you have lost your day's work, and that's why you take copies anyway; don't grumble!

The misplaced block is a more pernicious and subtle fault. How or why it occurs I do not know; but it will happen very occasionally on an old drive that a block of one file gets written over a block of another. As I said, I don't know how it happens, but a file dump will show that there is a block of foreign data in the middle of your data file. The data is in fact written well enough, and checksums will show no errors, but you now have two corrupt files. Now unless you read this file back there is no way you are going to find out that your system is fouled up: you will make your copies quite happily. Only when it comes to the week-end or month-end report, will it become apparent that your data files are so much garbage; by this time all your data copies are faithful copies of identical garbage. If you can keep your head while all about are losing theirs, and blaming it on you, you will see that this could have been avoided by better programming.

Briefly, the principle needed is to use some sort of data validation to check out files on which work has been done.

MINI LEGACY

At least some of the problem of frequent access arises from established practice on mini-computers, where 64 megabytes might be considered OK for disk storage, instead of perhaps 1% of that amount, and it has been customary until recently to make do with relatively tiny amoungs of RAM space (known to afficionados as "core", and formerly very, very expensive), compared for example with the lavish quantities used by spreadsheets on micros where it is not unusual for the RAM space to exceed the single disk capacity. This meant that people who knew how to program minis tended to use the same techniques on micros, and to teach others likewise.

Programming a computer with a vast amount of fast access disk space, and very little program space, means that you write everything to disk pretty well as soon as you get hold of it, and read everything off in the same sort of way. It also means that you have good sort utilities provided, and when a data file nees to be read in a different way, why then you just go and sort it right there on the disk.

RECORDS STAY PUT

McCartney's First Principle of data processing on diskettes is that whenever you write a data record on diskette, it stays in that physical location until you have done with it. Avoid sorts on diskette at all costs. Index files may need to have their records shifted about, but do this by downloading the whole file plus the sort keys, and sorting in RAM. Then write back the entire index file at once. The ratio of disk space to RAM on most micros usually permits this without too much bother — for example,

allowing an 8-byte key and a 2-byte index number, you can sort some thousands of record keys in a 64k RAM and still have space for the program.

Physical sorts on floppy disk are both excruciatingly slow and cause excessive mechanical work. I will not attempt to compare degrees of awfulness; just avoid them. If all else fails and you must execute a physical sort on a very large number of short records, for example, try the following:

1. Read a batch of records into RAM from disk A:, say 1000 at a time, and do a tag sort on them;
2. Write the sort batches to B; as serial files if convenient;
3. From n batches on B:, execute a n-way merge back to A:. That is, open all the batches on B: as independent files, and read the first record on each. Compare, write the lowest one to A:, and read the next record from the batch used.

Repeat till done. Be patient.

RECOVERABLE DATA STRUCTURES

If you don't do physical sorts, you must build more elaborate data structures in order to get things into order. McCartney's Second Principle of data processing on diskettes states that all data structures (such as indexed random access, linked lists and the like) must be fully recoverable from data in the records, preferably by the methods used to build them in the first place. This means that if the process is interrupted, it can be restarted.

For example, consider the subset of invoice records which belong to a particular customer. In an infallible system, it would not be necessary to have any data in the record which actually referred to the customer, provided that the customer record could refer to the invoice in some way, for example through a linked list, or as a member of a sorted range. In practice, systems rarely prove infallible, and you must have the means to rebuild broken chains, lost pointers, and missing links. Give this a menu option "Rebuild Data Structure" or similar, so that the user can run it when his ledgers don't balance, or his sales analysis run is hit by a power failure or the like. If you are really crafty, you can arrange for the software to do this automatically — see some hints below.

Without going into details of the programming, this means that a transaction record must include pointers to records in every file which relates to it; for example a stock movement record should contain pointers to a material file, a source file, and a destination file. The pointers should be actual record numbers for quick access, not keys

LINKED LISTS

We will deal in the following examples with linked lists, (or multilists) because this is the structure which is most useful for non-sorted transactions. Consider stock records. Each record is a record of the movement of a particular amount of a certain product. We will not bother for the moment about where the stuff is going too, coming from, how it gets here and how much it costs. We just need to know how much of what came in or went out. We then need two files, a static (product) file and a dynamic (movement) file. In addition, we need some control data, which might be held in the first record of each file, or in a small independent control file.

Now suppose we put in movements for product P. We can construct a chain of records most easily by setting a pointer from the product record to the last movement record, and from the last movement record to the previous movement record, and so on back, see Fig. 1. The grave disadvantage of this is that the list has to be read backwards; this is viewed with disfavour in commerce. We therefore elaborate the structure a bit more by pointing to the last record, which in turn points to the first record, which then points forward, see Fig. 2. The dotted lines show the shifts in pointers when a new record is linked in.

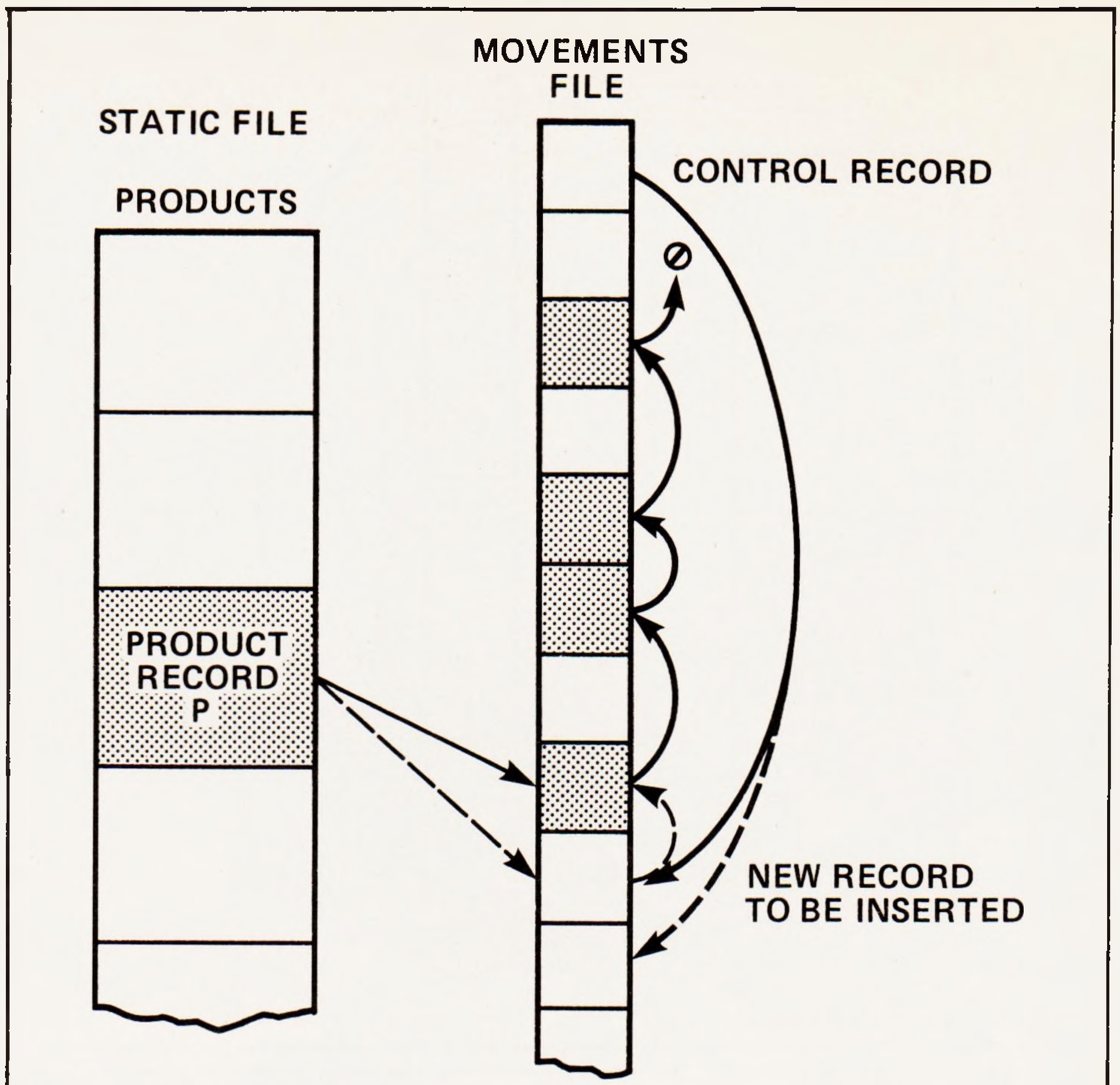


Fig. 1 Constructing a chain of records.

DISKETTE ACCESSES

Now tot up the number of disk accesses which are necessary to do this "on line":

1. **READ** the control record to find the first free record;
2. **READ** the product record to confirm input and find the last movement record;
3. **READ** the last movement record, to find the link to the free space from 1;
4. **WRITE** the last movement record, with a link to the free space from 1;
5. **WRITE** the new movement record, in the free space, with a link to the first movement from 3;
6. **WRITE** the product record, with a pointer to the new record;
7. **WRITE** the control record, with the next free record.

This gives seven accesses per new record. Every record which is updated is both read and written, and in addition the new record is written.

The situation gets more complicated if you want to identify a record with other static files as well.

All this and perhaps more is necessary only for on-line processing, in which data is incorporated in the system as soon as valid input is obtained. This is really irrelevant to the micro.

THE ZIP-UP METHOD

You can avoid constant reference to a control record by using the LOF variable, present in most micro operating systems now, provided you are putting records in in sequence. LOF gives the file size in bytes; divided by the record length, you get the last record number. You will perhaps need a control record for other purposes, but not this one. A control record is needed to tell you where you left off last time.

The product record should contain two

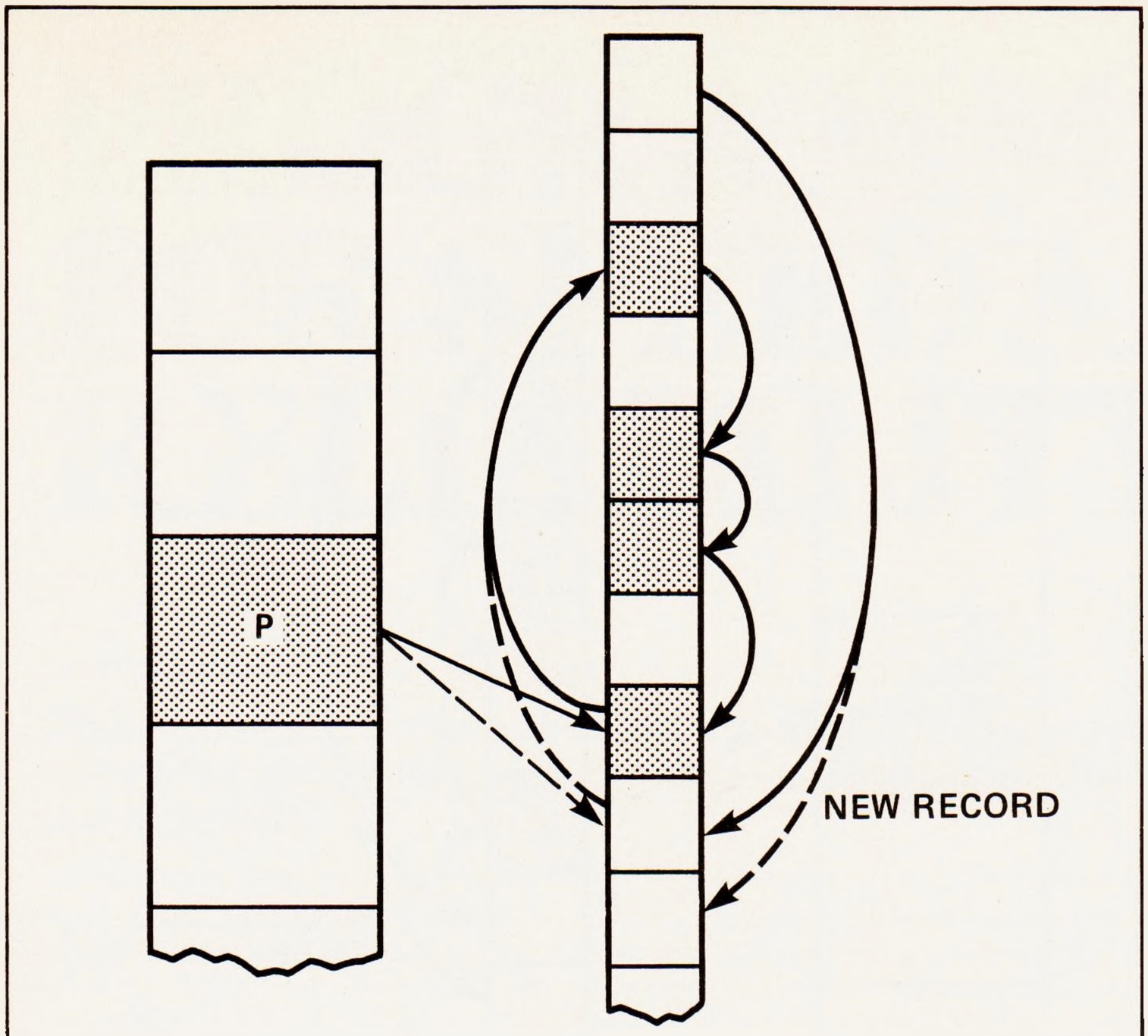


Fig. 2 Elaborating the chain.

pointers, one to the start of its own movement list, and the other to the end. The movement record must contain the record number of the product.

During input of a data batch:

1. READ the product record (necessary for validation);

2. WRITE the movement record. You work out where it goes from LOF.

When input is finished, zip up the data structure as follows:

a) create an integer array as large as the highest product record number, and having two columns, for the first and last movement records present in the batch list.

b) read the batch BACKWARDS from the last record to the previous last record logged in the Control. You can revalidate the records at this point (see below) to make sure they were written properly. Link it into a chain from the array pointers. This involves two accesses:

3. READ the record. If the product array is empty, put the record number in both columns of the array, and skip the next step. The columns correspond to the first and last members of the array chain;

4. WRITE it back together with the pointer to the first (earliest) member of that product chain, which is in the array. Then replace that array data with the current record number. Finally, link the array chain and the product record chain together, only for non-zero array elements, i.e. those product records which have actually been used. (Fig 3.) This requires 4 accesses per product record:

1. READ the product record, to get the last movement in the old chain;

2. WRITE it back with the last movement (from the array) in the new chain;

3. READ the last record in the old movement chain;

4. WRITE it back with the pointer (from the array) to the first movement in the new chain.

Finally write the new control record.

Suppose we enter a batch of 100 movement records, and access 20 products; the total number of accesses is now:

4 * number of movement records used	= 400
4 * number of product records used	= 80
2 * control record	= 2
TOTAL	= 482

This is a useful reduction on the 700 which would be needed for the first scheme.

Note another useful feature: the entire data structure can be rebuilt by using the zip-up for the full set of all movement records.

THE PRE-EXTENDED CHAIN METHOD

At some cost of storage space, we can get a dramatic reduction in disc access, and save time used in day-end processing.

The principle here is that the last record in a chain will point to an empty record, which will be used for the next record to be added. This saves two accesses for each record; see Fig. 4.

The basic technique is as follows:

1. INCREMENT the control record (don't

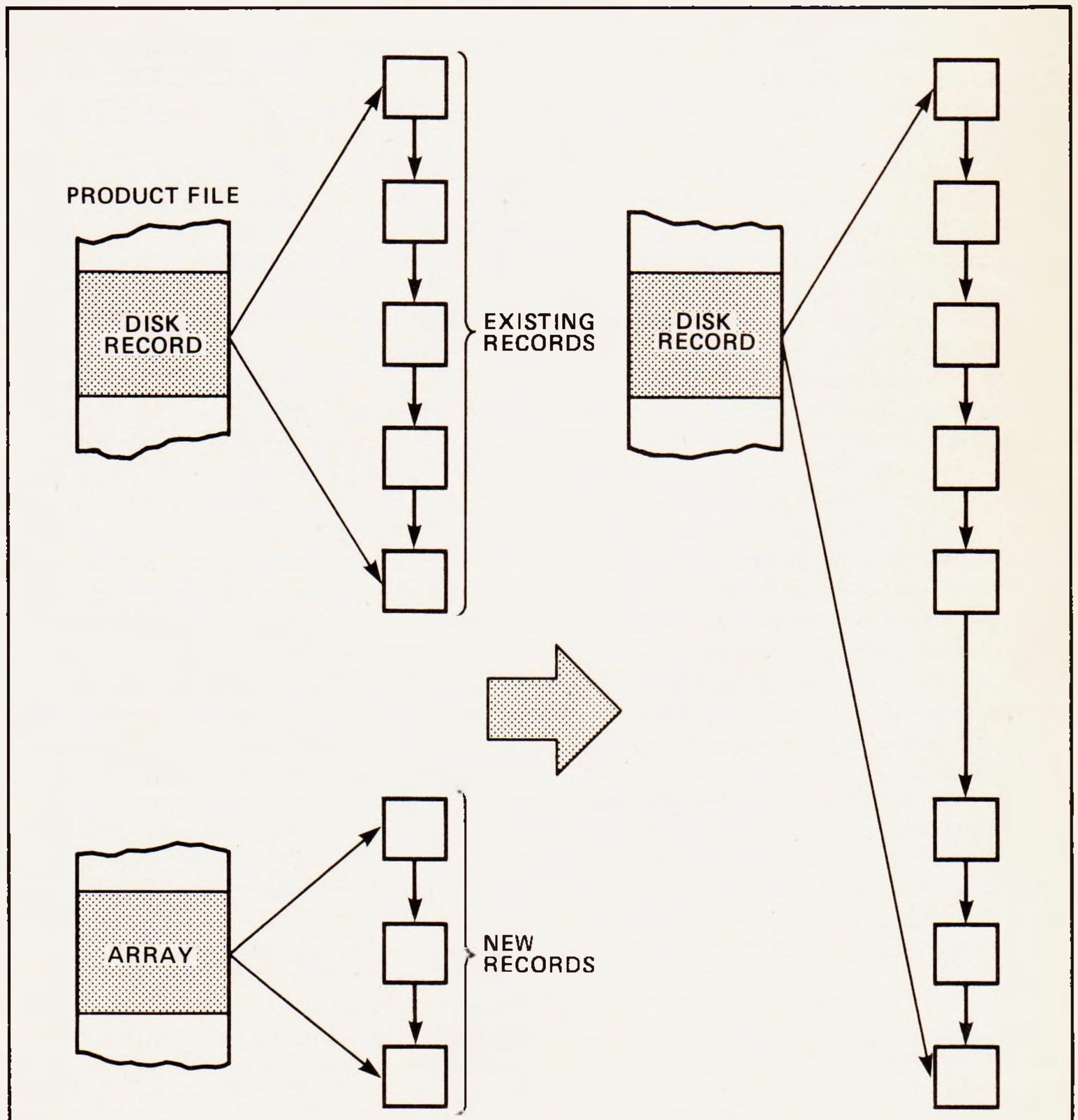


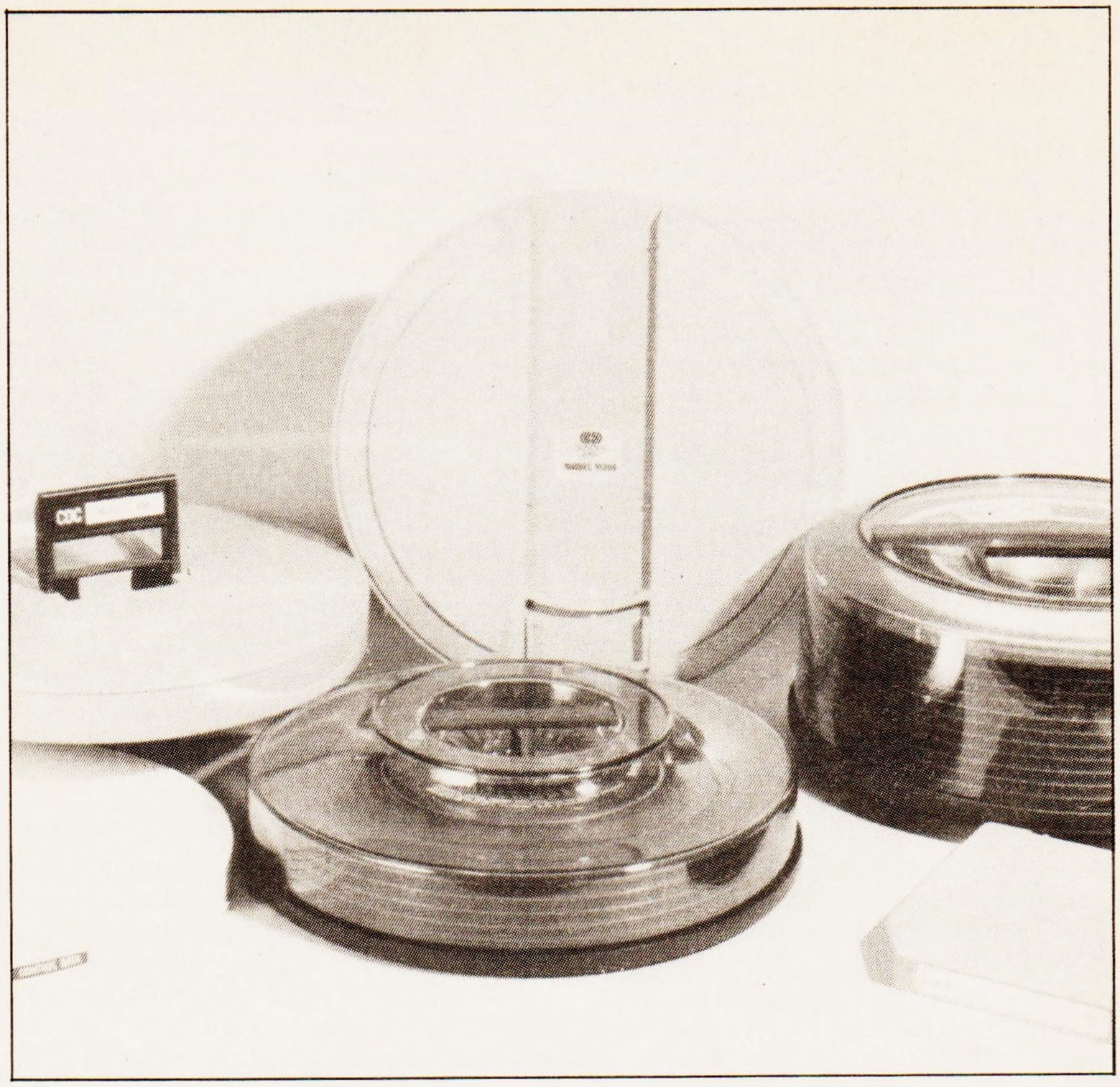
Fig. 3 Linking the array chain and the product record chain.

read it; just increment a corresponding variable and write it);
2. READ the product record, for the empty record and for input validation;
3. WRITE the movement record in the empty space, with a pointer to the next empty record obtained from 1;
4. WRITE back the product record with the same pointer.
It is necessary, if a product record has not been used at all, to assign it an empty record as well as a live record on the first access.

Now instead of working on the disk all the time, use an array and a control variable as follows:
a) create a one-dimensional array corresponding to the product records;
b) in step 1., just increment the number — don't write ti back;
c) in step 4., write back to the array, not the record;
d) in step 2., refer to the array first to find any existing empty record pointer.
Finally when entries are finished, write back any non-zero product pointers from the array to the product records, literally blank out the empty records by writing nulls except for the product pointer (necessary for recovery) and write back the control record.

The number of accesses is now:

2 * number of movement records used	= 200
2 * number of product records used	= 40
2 * control record	= 2
TOTAL	= 242



. . . but then again, with a few Megabytes to play with . . .

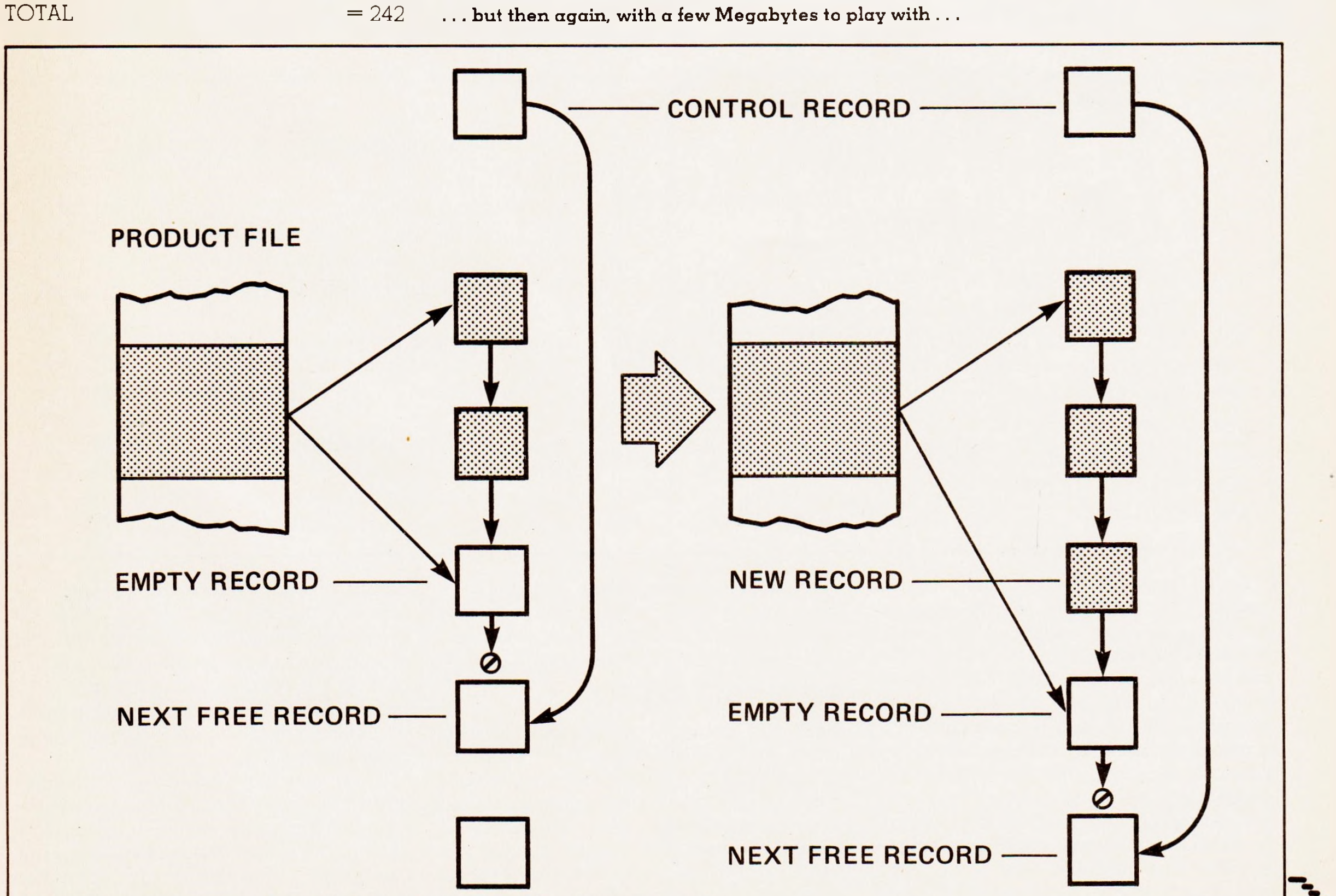


Fig. 4 Using the last record in a chain to point to the site for the next record.

Listing 1 For the XYZ999

```
100 'FILE INSPECTION UTILITY "DATADUMP"
110 ON ERROR GOTO 370
120 PRINT CHR$(27)"E": L=0: BL$="          "       'ESC code clear screen
130 PRINT "FILE CONTENTS PRINTING UTILITY 'DATADUMP'": PRINT
140 FILES: PRINT
150 INPUT "FILE NAME (ALT-C to terminate):  ",FI$
160 INPUT"Line length : ",LIN%
170 INPUT"Start at line : ",START%
180 INPUT"Finish at line: ",FIN%
190 INPUT"Do you want text, hex, or both? (t/h/b):",Q$:
      IF Q$<>"t" AND Q$<>"h" AND Q$<>"b" THEN 190
200 PRINT:PRINT"PRESS ANY KEY TO STOP...."
210 OPEN "R",£1,FI$,LIN%: FIELD £1, LIN% AS B$
220 LPRINT CHR$(15)      'condensed print for EPSON
230 LPRINT "DATADUMP.BAS utility on file:    "FI$
240 LPRINT "Line (record) length = ";LIN%;"  Starting byte = ";
    (START%-1)*LIN%+1
250 LPRINT
260 FOR J% = START% TO FIN%:
      A$=INKEY$:
      IF A$<>"" THEN 350
270   GET£1,J%:
    IF Q$="t" THEN 300
280   LPRINT LEFT$(STR$(J%)+BL$,7);:
      FOR K%=1 TO LIN%:
      B%=ASC(MID$(B$,K%)):
      HX$=RIGHT$(("0"+HEX$(B%)),2)+" ":
      LPRINT HX$;:
      NEXT: LPRINT
290 IF Q$="h" THEN 320
300   LPRINT LEFT$(STR$(J%)+BL$,7);:
      FOR K%=1 TO LIN%:
      B%=ASC(MID$(B$,K%)):
      IF B%<33 OR B%>126 THEN B%=46
310   LPRINT CHR$(B%)"  ";:
      NEXT: LPRINT
320 IF (J%-START%+1) MOD 24 = 0 AND Q$="b" THEN LPRINT CHR$(12):
      GOTO 340
330 IF (J%-START%+1) MOD 48 = 0  THEN LPRINT CHR$(12)
340 NEXT J%
350 LPRINT: LPRINT
360 RUN
370 IF ERR=62 THEN B$=CHR$(255): RESUME NEXT
380 PRINT"ERROR: ";ERR;" LINE: ";ERL
390 END
```

Again, provided the movement records contain the pointers to the product records, you can regenerate the data structure, but not by the same method. You can use the zip-up method because you have got product pointers in the empty records — if you didn't, you might have garbage or perhaps old records from a previous accounting period, which would cause havoc.

The weakness of this method is that data is lost if the write-back is not executed — e.g. because of power loss or Control-C exit. This is not a severe penalty for the increase in speed achieved, and reduced mechanical wear on drives and discs. All popular word processors and spreadsheets have the same weakness, and they get along fine, after all.

RE-VALIDATION

As I said above, it is necessary to read back the transactions you have written during the day to ensure that they actually have been written, and are to be found where expected. This insurance is necessary regardless of which method you have used to link them. It is also general practice in accounting and commercial applications to produce a batch list, giving details of each transaction entered, and checking that the totals are OK. These two operations should be combined in a final read-through of the list after linking-in. It is desirable to read only from the start of the current batch. If a batch list can be read and printed and tots up OK, then the list on disk is also OK.

If you think that there is still a possibility that some minor item might not be written right, remember the principle that to err is human, but a comptuer will foul things up completely. In other words, computers don't make small clerical errors, but if they do make a mistake, they will devastate a

large area of data, typically a disk block or more. It is important not to get too anthropmorphic about computers. If you do come across a clerical error — typically transposed digits or a wrong key struck or missing — it is a simple matter to arrange to correct the record and re-run the batch list.

The production of a batch list, which also serves as an audit trail, means another read, but the read is sequential and as mentioned above causes least distress to the mechanical systems. If a printout is not needed, you can do internal validation checks; read each record and validate the files in much the same way as you validated the input (except don't look up associated files). The same principles apply; you are not looking for small clerical errors, but gross absurdities. For example, invoice amounts are typically £0.01 to £1,000,000. A piece of text read from the diskette as a floating-point number may register any value from 1 E −99 to 1 E 99 positive or negative; the invoice amount is a relatively tiny range compared with this. The same applies to record pointers and any other numeric data you have in the record. If these are stored as logical integers, Look out for negative or excessive values or backward-pointing links, or the mysterious number 8224 provided that you have fewer records than that. (There is no real point in looking for 8224 specifically, but if you see it several times in a printout it's a month's salary to a worn-out floppy that your data is so much garbage — see if you can figure out why.)

You can combine a validation with the back-reading of records in the zip-up method, but this is not fully watertight; the records have to be written back again, and in in any case, a printout would be in reverse order.

What to do, then, if you find corrupt data? It is a rare enough event; the standard procedure of going back to the last copy will cope with it very well. But it is a symptom of underlying failure; it is a good idea to get the system on which it happened checked out thoroughly. When the maintenance man comes it is also a help to have some printed evidence of what happened; hence the corrupted batch list is useful. Even if you don't produce this list for accounting purposes, it is a good idea to have a program capable of printing it. You will of necessity have something like this for program development work anyway. A file dump utility will do very well. The accompanying short listing is an invaluable tool for MSDOS files.

DATA RECOVERY

Data recovery does not mean the correction of corrupted files such as mentioned in the last paragraph; as I said this is rare enough not to warrant any special deletion of corrupt sections of data, but common enough and harmful enought to need identification. But if you are working with links and pointers or indices in RAM, and experience a power failure, or just get accidentally dropped out of the program, your data structure on disk may be all mixed up. This is common enough to make it worth guarding against and to build in automatic recovery; the technique is to have a flag in a control file on the data disk set when you take the data into RAM for organising; then reset the flag when you put it back. At start-up, the system should then inspect flags and re-organise any corresponding data structures. Make sure, as I said before, that all records have the necessary field built in to enable this to be done.

PERIOD-END PROCEDURES AND CLEARDOWNS

At the end of accounting periods, you need to bring forward totals, and some of your transaction data. It is a very good idea to transfer the lot to a new master diskette at this point, and archive the old data diskette. Reorganisation and revalidation of all data should take place during the month-end run which will work, for example, with input from the old month on drive A: to output to start the new month on drive B: Archive copies then form a valuable audit trail, and your master disks, changed each month, don't get clapped out. If you happen to be doing it on a weekly basis, your archives may get to be large and expensive, but you don't need to keep them for ever, and can recycle old disks after a suitable time.

ARTICLE SUBMISSIONS

Computing Today welcome articles submitted by readers, but potential contributors are asked to observe the following guidelines:

- Feature articles should be typewritten with double line-spacing if possible, and should ideally cover between 8 and 16 sides of A4.
- Program listings when absolutely necessary should be kept short. When listings are supplied they must be accompanied by text describing the algorithms employed and, if possible, a flowchart.

Articles should be sent, along with an accompanying letter describing the nature of the article, to:

SUBMISSIONS
COMPUTING TODAY
1 Golden Square
London W1R 3AB.

Please ensure that your name and address is printed clearly on your covering letter.

LEARN UNIX

Mark Woodley

Part two: the UNIX file system

One of the great advantages of the UNIX system is the simplicity with which it is implemented. The most important role of the operating system is to provide a filestore that not only handles files, but also manages the use of the computer's input and output devices. The inputs and outputs are an essential part of any computer system because without them the computer would be useless. And it is right here that the powerful simplicity of the UNIX system becomes most apparent.

A standard protocol is used within the whole filing system. Any file is communicated character-by-character and is terminated by the end-of-file character. There are no other restrictions, for example, the lines of a file may be demarcated with carriage return, line feed.

Devices are treated in exactly the same way. The operating system will automatically provide the necessary interface, so that a device behaves as if it were a file. The implications of this are tremendous, because without changing a program, its inputs and outputs can be redirected anywhere. Programs that were designed to write their output to a particular file could just as easily be made to send their output to the printer by means of a simple operating system command. In effect, the opening and closing of files is handled by the operating system, and the program just talks about file numbers relating to open files, called 'file descriptors'.

FILE TYPES

There are three types of file in the UNIX file system; these are Ordinary Files, Directories and Special Files.

Ordinary files can contain the characters of a document, the source code of a program, or the executable (binary or object) version of a program. The user is free to structure the file to suit the particular application in hand.

Directories are like ordinary files except the information they contain holds special meaning to the operating system. Directories hold the names of other files, be they ordinary files, special files or even other directories, but they do not contain the files themselves. Instead they contain a pointer called a 'link', that points to file information on the media used to access the file.

Special Files are provided for every input and output device. They are interfaced to behave in the same way as ordinary files, but information is sent directly to and from the device.

FILE SYSTEM STRUCTURE

Because directories can hold other directories, a natural structure forms in the filing system. Each user has a directory of his or her own files, within which can be created as many other directories and subdirectories as wished, without limit, provided there is enough room available. Users are classed together by the system administrator and their directories are accessed through class directories. The class directory contains all of the user directories for each group. Groups could be made up of separate project teams, classes of students or perhaps members of a department. The class directories are kept in a high-level directory called **usr**, through which all of the user's files can be accessed. **usr** is one of several directories that occupy a higher level 'root' directory called / or 'root'. In all, 'root' contains the following directories:

The set of **usr** files;

The system commands, in 'bin';

The library (lib) files used to keep essential information, such as compilers and their libraries;
The general 'etc' files such as the password file and the 'message of the day';

Temporary 'tmp' files, generated by the system that have a short life;

The system's full complement of special files, that provide interfaces for peripheral devices.

So the resulting hierarchy is as shown in Fig 1. with more files in each directory.

PATHNAMES

To find a file in the tree structure, you have to start at the root and follow a path through the directories until you reach the file you want. The full name of the file is then written in terms of the path that you would have to follow to find it. This is called the pathname of the file, and is written as a series of component files separated by slashes. For example, if there were a user with directory 'jim' in class 2, that had a program called 'test', then its pathname might be:

/usr/gp2/jim/test

FILE INFORMATION

Associated with each file in the file system are the following data:

1. First of all there is a record of the file type, so that the operating system can tell between an ordinary file, a directory, and a special file.

2. Except for special files, there is also a record of the time and date of creation, the time and date of the file's last access, and the time and date of the file's last modification.

3. If it is an ordinary file, there is a count of the number of directories in which it appears.

4. Then there is a special user identification number, which can be used to find the owner's user-name.

5. There is also information relating to the length of the file in bytes, and the physical address of the file on the media.

6. For all files, there are a set of bits that represent the file's access permissions. These describe for each of three types of user, the types of access available. In UNIX there are three relationships that a user can have with a file. He can either be a 'user' (owner) of the file, a member of the user's 'group', or an 'other' member of the user population. For each class of user there are read, write and access permissions. From the most significant bit down, the following bits are used to indicate a file's access permissions:

owner:	read access
	write access
	execute access
group:	read access
	write access
	execute access
other:	read access
	write access
	execute access

You are allocated user groups by the system administrator, and these are not necessarily related to the class you are in.

The meanings of the access permissions vary with the file's type. For ordinary and special files they are the same. Read access means that you read from a file, write access means that you can write to and modify a

file, and execute means that if it is an executable program, you have the right to run it.

It is slightly different for directories. Although read and write access appear to be unchanged, writing to a directory, by altering the files within it, can only be done using special system commands. And now, execute access indicates whether a user is granted access to files in the directory.

There is also a tenth bit which is set when the computer grants a user the access privileges of the file's owner if the file is executable. The program is able to access files with the access rights of the program's owner, that its user would not normally have access to, so that certain files can only be altered by the program.

FILE ACCESS COMMANDS

When you create a file, the Shell will normally allow all types of access unless you specify otherwise. A subroutine called **'umask'** that is part of the Shell can be called up to change this. An octal number is used to represent the access bits.

umask <octal number> <return>

For example, if we wanted to set up a mask that allowed user read, write and execute to all our files, and group read access our access bits should look like this:

access bits	user	group	other
	r w x	r w x	r w x
in binary	1 1 1	1 0 0	0 0 0
and in octal	7	4	0

— so umask will look like this:

umask 740 <return>

You can also use a command called 'chmod' that changes the access modes of a file; and this takes this form:

chmod <octal number> <file name> <return>

These commands can also be used in more advanced applications when you want to change the tenth bit.

GENERAL FILE COMMANDS

The 'cat' command, short for concatenate, can be used to list files. The 'cat' command sends the files mentioned in its arguments to your terminal screen.

Files can be output to the system line printer in the same fashion using the 'pr' command:

pr<filename><return>

— and to copy one file to another, there is the 'cp' command:

cp file1 file2<return>

— which copies the contents of file1 to file2.

To rename a file, there is the move 'mv' command'

mv file1 file2<return>

Here 'mv' renames file1 as file2.

You can delete files with the remove, 'rm' command. For example:

rm <filename><return>

You may find some of these abbreviations unusual, and so it may take a little time before you learn them all.

DIRECTORY MAINTENANCE

You can list your directory of files with the list directory command 'ls'

If your directory was /usr/guest, then 'ls' might produce the output shown in fig. 2.

The first entry gives the access permissions for 'user', 'group' and 'other' members of the user population, in that order. The last file for example, has user read, write and execute access, with read access to other users. The letter 'c' at the start of an entry indicates that the file is a directory.

The other entries from left to right are: the number of links to the file (which will always be one for a new file); the owner of the file; the time and date of the file's last access or modification; and the filename.

Each directory contains two entries '.' and '. . .', where '.' is the shorthand for the current directory and '. .' is the shorthand for its last, parent directory. In this example, '.' is /usr/guest and '. .' is /usr. These shorthands provide a convenient way of moving up or down the directory hierarchy.

You use the 'cd' command to change the current directory. If we wanted to switch to the directory of another user we could type:

cd . ./fred<return>

— rather than:

cd /usr/fred<return>

Note that the path to the current directory can always be found with the 'pwd' command:

pwd<return>

— and the system will reply with something like:

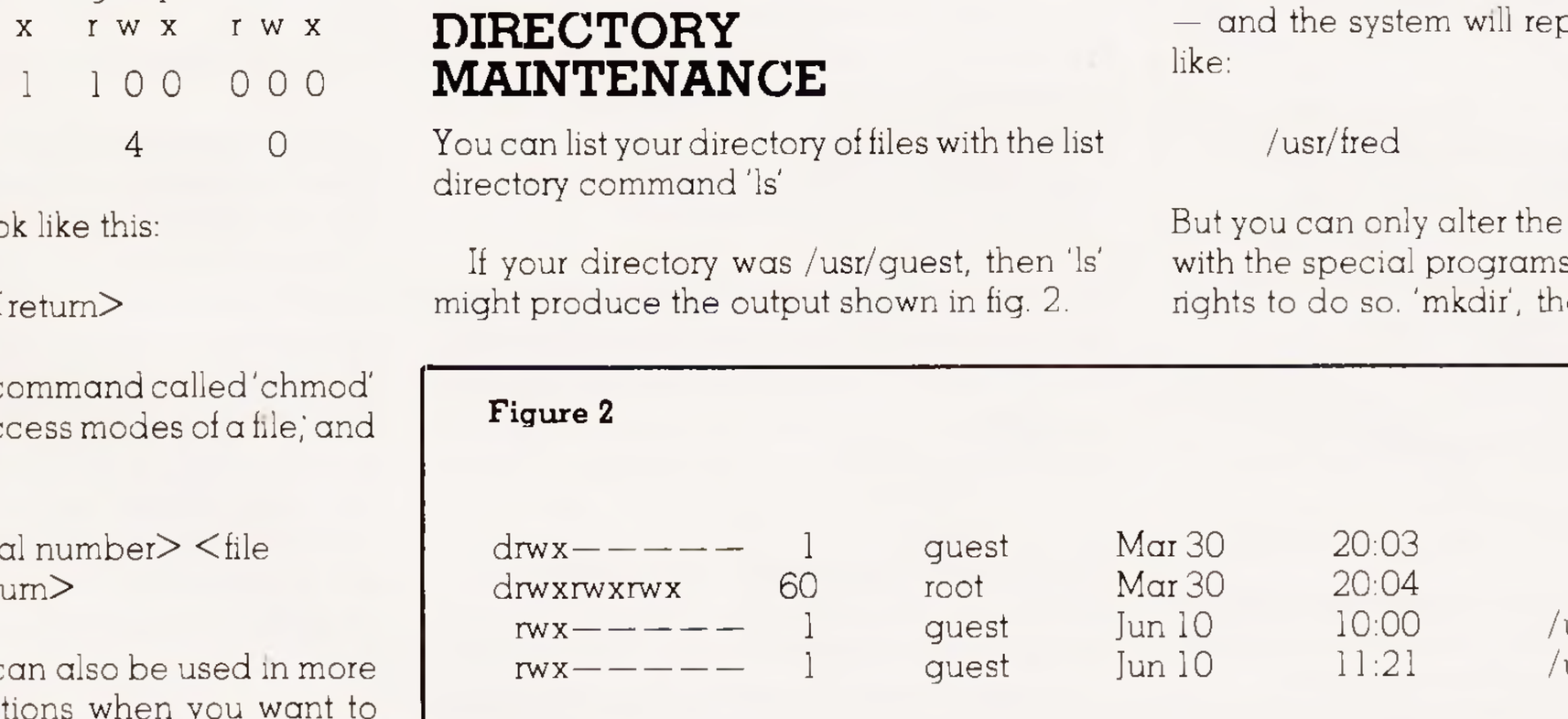
/usr/fred

But you can only alter the files in a directory with the special programs that have access rights to do so. 'mkdir', the 'make directory'

Figure 2

```
drwx------   1   guest   Mar 30   20:03
drwxrwxrwx  60   root    Mar 30   20:04
 rwx------   1   guest   Jun 10   10:00   /usr/guest/file1
 rwx------   1   guest   Jun 10   11:21   /usr/guest/file2
```

Figure 1: file system structure.



command creates an empty directory file in the current directory.

mkdir project<return>

— will make a directory called project, with the access rights specified by 'umask'.

You can now create files in that directory by changing it and proceeding as normal and you can create further subdirectories.

When you want to delete a directory, there is a special rm command, 'rmdir':

rmdir project<return>

Now if you want a copy of another user's file, the easiest way to go about it is to establish a link to it (note that you cannot form a link with a directory).

To establish a link within your directory to the required file, type:

ln <filename><return>

-- and the file will appear in your directory, with the number of links incremented.

Figure 3 shows how UNIX commands tie in with MS-DOS and CP/M.

Figure 3

UNIX	MS-DOS	CP/M
cat	type	type
lpr	↑Ptype↑N	↑Ptype↑P
cp	copy	pip
mv	ren	ren
rm	del	era
ls	dir	dir

UNIX supports the same 'wild cards' as MS-DOS and CP/M. An asterisk (*) can be used to represent anything before and after a dot, and a question mark (?) can be used to represent any single character. So:

ls ?.*<return>

— will give all the directory entries for files in the current directory that have a single character prefix, and any extension.

In addition, any one of a range of characters can be selected by specifying the range inside square brackes. So:

[a—c]

— would represent either 'a' or 'b' or 'c'.

ADVANCED SHELL USE

The Shell can do much more than perform single line commands containing arguments. It lets you redirect inputs and outputs, and run several programs concurrently, it can recognise a whole set of structures for command lines and it lets the user use working variables.

REDIRECTING INPUTS AND OUTPUTS

We'll start by looking at the way inputs and outputs are redirected. The devices which are normally geared up to receive the program's inputs and outputs are called the standard input and standard output devices respectively. When you are using UNIX for the first time, from a terminal, you can expect the standard input to be the terminal's keyboard and the standard output to be its display.

The following symbols are written as arguments to a command to redirect standard inputs and outputs.

><file>	send standard output to <file>
>><file>	append standard output to <file>
<<file>	read standard input from <file>

So, for example, to send your directory to the line-printer you could type:

ls >/dev/lpr<return>

To add a list of your directory files to the end of a file called 'text', instead of from the keyboard, you could type:

ls >>text<return>

— and to get an editor to read a prepared set of instructions from a file, instead of from the keyboard, you could type:

ed text <file<return>

You can of course redirect different things in the same command, for example, you could copy one file to another with either:

cat file1 >file2<return>
cat <file1 >file2<return>

You can specify inputs within a command in 'here documents'. In the command there will be the following redirection symbols:

<<<char>
text
<char>

Whichever character you choose, its purpose is to demarcate the text.

As an example, to create a file you could type:

cat <<!This is an example file!
>egfile<return>

When inputs and outputs are redirected, the user does not have to worry about opening and closing files, all of this is handled by the system automatically. Once a file has been opened by the system it is then referred to by a number, called a file descriptor. The following file descriptors can be assumed to exist when any program is run.

0 Standard Input
1 Standard Output
2 Standard Error

The extra Standard Error is used as the channel for error messages. Normally, this is set up as the standard output device, but it can be changed (the system actually opens it for input and output.)

The following redirections can be used with file descriptors, but there will be a more in-depth discussion these when we look at system programming.

>& <fd>	define <fd> for standard output
<& <*fd*>	define <fd> for standard input
>& —	specify no standard output
<& —	specify no standard input

COMMAND SEPARATORS

We have already come across one of the UNIX command separators, which is <return>. But as well as the <return> key we can use other things to separate commands, in a more convenient way. It is possible to group several commands on the same line like this:

<command1> <arg> <arg> <arg>
; <command2> <arg> <arg>
<arg>

Once you have pressed <return>, you won't get the next Shell prompt until all of the commands have been executed in sequence.

It is often convenient to run programs in a particular sequence, usually where the outputs of one command are the inputs of another. For example, you could get a sorted list of users currently logged on to the system using the 'sort utility in this way:

who >temp; sort <temp<return>

To do this, we create a temporary file (called 'temp') to store the output from 'who' and then this file is sorted using the sort utility.

Recent versions of UNIX, however, offer a much more elegant solution, which does not involve the use of temporary files. The means by which this is achieved is called a 'pipe'. You set up a 'pipeline' between two commands, such as who and sort, and the outputs from the first become the inputs to the second. Commands that are piped together are separated by the pipe symbol '|'. We could thus rewrite the previous example, like this:

who| sort<return>

When the Shell sees the pipe symbol it invokes a situation that is then controlled by the operating system, rather than the Shell. The operating system sets up a buffer in memory, and the two programs take turns to write to and read from the buffer, until they are both finished. It works very quickly, because of the intervention of the operating system.

MULTITASKING

The Shell can also be made to pass the execution of a command to the operating system, so that the Shell does not have to wait for the command to finish before it can give its next prompt. The command is effectively disconnected from the terminal.

We would set up such a task from the Shell,

by terminating the command with an ampersand:

who¦ sort¦ lpr &<return>

Now we can continue typing in more commands to our Shell while this task runs on in the 'background'. Any program that is running under the operating system is termed a 'process' and for each process there is a unique process number. The process number for the command will be displayed at the terminal when the command is invoked.

You can keep track of the processes that are running with the 'process status' 'ps' command. This normally gives a list of your processes with their numbers.

The newly spawned process can be killed at any time with the 'kill' command. If the process number was 872 for example, then we could kill this process by typing:

kill 872<return>

— but this will only work if the process belongs to you.

Normally, any background processes that are running when you log off will be automatically killed. This can be very inconvenient. In a previous example, our printout might never appear, or may stop half-way through. To get around this obstacle we can use the 'no hang up' **nohup** command, and this prevents the process from being killed. Our previous command would then be written:

nohup who¦ sort¦ lpr & <return>

But a command involed in this way can still be killed with a special form of kill:

kill –9 872<return>

Also if the process you are running is not required to finish urgently, you can nominate it to be scheduled at a low priority, with the 'nice' command. It works like nohup and can be used in conjunction with nohup like this:

nice nohup who¦ wsort¦ lpr &<return>

SHELL VARIABLES

The shell maintains certain variables that you can use in commands. Each one is preceded by a dollar sign. Here are most of them:

$0 The name of the command that the Shell is currently running.

$£ The number of arguments to that command, in decimal.

$n Can be used to represent the nth argument.

$? Programs can return values to the shell when they terminate, this variable gives the termination value (or the 'exit status') of the last command that the Shell ran. This is normally zero if the command was successful.

$! the process number of the last background process that the Shell ran.

depending on the string variable employed.

That's plenty to be getting on with, but in the final part I will look at how UNIX works at the gut level. Until then, here are the majority of UNIX commands, in summary.

$$ The process number of the Shell itself.

$PS1 Shell prompt, normally $.

$PS2 The Shell's 'please continue' prompt, normally >.

$MAIL A flag used to indicate whether mail is present in mbox. Zero means there is no mail.

$HOME The pathnames of your home directory.

SEARCH PATHS

$PATH: I mentioned last month that the Shell could be made to search specific directories for its commands. The Shell knows which directories to search through, and in which order, because it has variable holding a 'search path'. $PATH is the search path that the Shell is using. It consists of directory pathnames, separated by colons:

. : /usr/bin : /bin

In this example, the search starts at the current directory, moves on to /usr/bin, and if the command has not yet been found, moves on to /bin.

$CDPATH: When you change directories with 'cd' the change directory command, you don't have to give the full path name. This is because 'cd' has its own search path, which it uses to find the directory you specify. This path is CDPATH.

COMMAND STRUCTURES

The variables, particularly those for arguments, can be used to build up structured commands that look similar to the structures used in Pascal and PDL. The real advantage of writing structured commands is that they can be built to from macro programs (called Shellscripts) in a file, and then executed by invoking the Shell with:

sh <<script file>

There is a whole host of command structures, but to use them I suggest you refer to the system documentation.

There is a conditional statement that will execute one of two lists of commands, depending on the exit status of another command:

```
if command-list
    then command-list
    else command-list
fi
```

A 'while' loop works in the same way:

```
while command-list do
    command-list
done
```

A 'for' loop can be used to select commands that apply to particular arguments:

```
for shell-variable do
    command-list
done
```

This has more complicated forms, but I won't mention them here. Finally, there is a selection structure:

```
case string in
    string) command-list ;;
    string) command-list ;;
      . . . etc
esac
```

This will perform a particular command-list, depending on the string variable employed.

That'ts plenty to be getting on with, but in the final part I will look at how UNIX works at the gut level. Until then, here are the majority of UNIX commands, in summary.

Learn Unix continues in the next edition of *Computing Today*.

COMMANDS

adb	Machine code debugger.
ar	Archive and library maintainer.
at	Specify execution time of a command.
awk	Pattern scanner and processor.
basename	Remove filename affixes.
cal	Calendar
calendar	Reminder service
cat	Used to list and concatenante files.
cb	C program 'beautifier'.
cc	C Compiler.
cd	Change directory.
chgrp	Change the group associated with a file.
chmod	Change access permissions on a file.
chown	Change the owner of a file.
cmp	Compare files.
col	Filter special characters.
comm	Select or reject common lines in two files.

cp	Copy file.
crypt	Encrypt document.
date	Read or set current time and date.
dd	Convert and copy a file.
deroff	Remove typesetting commands from a file.
df	List free disk space.
diction,explain	List wordy sentences.
diff	Describe the differences between two files.
du	Describe user's disk usage.
echo	Display a message.
ed	Line editor.
eqn,neqn	Typesetter for mathematics.
expr	Evaluate arguments.
/etc/haltsys	Shutdown system in single user mode.
/etc/shutdown	Shutdown system in multi-user mode.
expr	Evaluate mathematical expression
file	Describe type of file.
find	Find a file.
format	Format disks and tapes.
fsck	Check integrity of file system.
f77	Fortran 77 compiler
grep	Find patterns in a file.
join	Links two databases together.
kill	Terminate a process.
ld	Link editor.
learn	CAI on UNIX
lex	Lexical Analyser generator.
lint	Reports on the quality of a C program.
ln	Link to file.
login	Enter system.
look	Find particular lines.
lorder	List relations between object libraries.
lpr	Print document.
ls	List directory.
mail	Send mail.
make	Maintain program groups.
man	Display entry from manual.
mesg	Accept/Reject messages.
mkdir	Create directory.
mv	Rename file.
newgrp	Log in to a new group.
nice	Assign low priority to a task.
nm	List symbol table of object file.
nohup	Allow programs to run after logging off.
nroff,troff	Document preparation/typesetting aids.
od	List program. (Octal dump)
passwd	Change password.
pr	Format document.
prof	Display profile data.
ps	Display current processes.
ptx,pubindex	Generate a document index.
pwd	List current directory.
ranlib	Convert archives to random libraries.
refer,lookbib	Insert literature references.
rm	Remove file.
rmdir	Remove directory.
sdb	Debugger for C, Pascal, f77.
sh	Shell.
size	Report size of object file.
sleep	Suspend execution for a time period.
sort	File sort.
spell	Check spelling in file.
strip	Remove symbols and relocation bits.
stty	Set terminal.
style	Comment on a document's style.
su	Become superuser.
sync	Update files before power-off.
tabs	Set terminal tabs.
tail	List last part of file.
tar	Tape archive.
tbl	Format tables for nroff and troff typesetters.
tee	Copy piped output to screen.
test	Evaluate true/false expressions.
time	Time a command.
touch	Update date last modified of a file.
tr	Copies fiels and substitutes specified characters.
tsort	Topological sort.
uniq	List repeated lines in a file.
units	Unit conversion program.
uucp	Unix to Unix copy.
vi	Visual display editor.
wall	Send message to all users.
wc	Word count.
who	List who is on the system.
write	Send message.
yacc	"Yet another compiler-compiler".

COMPUTING Today

NEXT ISSUE OUT FRIDAY SEPTEMBER 13th

WARGAMES

Computer assisted wargaming is something that has countless attractions for hardened campaigners. By building the rules and conditions of a particular scenario into a program, much of what some would consider the 'tedium' of wargaming can be left to a micro, leaving the interesting bits for the Generals.

OMNI READER

Optical character recognition is a field that has been researched for many years with very little commercial success. However, the Omni Reader could prove that character recognition could be made cheap enough for home users.

LITTLE GIANT

We review the ACS PX1000 text processor – a dedicated wordprocessing unit that weighs in at just 14oz.

All this and more in the next edition of

Computing Today

Articles described here are in an advanced state of preparation but the circumstances may dictate changes to the final contents.

THE EVOLUTION CONTINUES..

ACT MICRODEALER

xi APRICOT

CPU	8086
MEMORY	256K RAM
LANGUAGES	Microsoft BASIC, Personal BASIC
MASS STORAGE	No cassette drive Integral Sony 3½" 315K microfloppy disk drive Integral 5 or 10 Mb hard disk
OS	MS-DOS 2.11 with GSX bundled CP/M-86 (not yet available) Concurrent CP/M-86 (not yet available)
KEYBOARD	QWERTY, cursor, numeric pad, function keys
INTERFACES	RS-232C, Centronics, Microsoft mouse
DISPLAY	Monitor (supplied)
GRAPHICS	80 by 24 text with block graphics 800 by 400 high-res graphics under GSX
SOUND	No

Notes. The Apricot xi is a development of the award-winning Apricot, and replaces one of the latter's disk drives with an integral hard disk, providing vastly increased storage with faster access. Memory may be expanded in 128K increments to a maximum of 768K. The languages and operating systems mentioned above come bundled (except for Concurrent CP/M) and four software tools are also bundled, including an asynchronous package for use with the optional modem card.

HAMPSHIRE

TIMATIC SYSTEMS LTD
The Market, Fareham.
Tel: (0329) 239953

For the complete range of Apricot hardware and software. Also dealers for Zenith, Memotech. For future information call or ring anytime.

SCOTLAND

SIRIUS
is alive and well and supported at
ROBOX
(Office Equipment) Ltd,
The Scottish Computer Centre
Anderson Centre, Glasgow
041-221 8413/4
34 Queen Street, Edinburgh
031-225 3871

WEST MIDLANDS

Q data limited
The Black Country's specialist in micro-computing. Full range of ACT Apricots and IBM personal computers.
The Limes, High Holborn, Sedgley, West Midlands.
Tel: Sedgley (09073) 62331

CBM MICRODEALER

Notes: The Commodore 64 is a popular micro with a great deal of games software available. There is also some business software available.

The Commodore 715B is the top model in the 700 range of business machines.

NASCOM MICRODEALER

NASCOM 3

CPU	2 MHZ Z80
MEMORY	8K or 32K inbuilt RAM (expandable to 60K)
LANGUAGE	Full Microsoft BASIC
MASS STORAGE	Single or twin 5.25" disc drives 350K capacity per drive
OS	NAS-DOS or CP/M 2.2
KEYBOARD	Full size QWERTY
INTERFACES	RS232 and 16-bit parallel
DISPLAY	40 or 80 column 25-line display
GRAPHICS	High resolution graphics with 8 foreground and 8 background colours (400 x 256 pixels) Double density graphics with 2 colours (800 x 256 pixels)
SOUND	No

LANCASHIRE

EV COMPUTING
700 Burnage Lane
Manchester M19
Tel: 061-431 4866
80-BUS SOLUTIONS

SHARP MICRODEALER

SHARP MZ-3541

CPU	Z80A (two), 80C49
MEMORY	128K RAM, 8K ROM
LANGUAGE	Sharp BASIC
MASS STORAGE	Twin integral 5¼" floppy disk drives, total capacity 1.28 Mb
KEYBOARD	QWERTY, cursor, numeric pad, function keys
INTERFACES	RS-232C, Centronics, interface for extra external floppy disks
DISPLAY	Monochrome monitor, colour optional
GRAPHICS	80 by 25 text, 640 by 400 high-resolution graphics
SOUND	Single channel

Notes: The Sharp MZ-3541 is aimed at the businessman. RAM is expandable to 256K, while two disk drives may be added externally to complement the integral pair. Colour is only possible with the optional graphics expansion RAM. One Z80 handles the main CPU activities while the other handles peripheral activities. The third processor handles the keyboard. The availability of CP/M means a ready supply of business software.

LONDON

SHARPSOFT LTD.

Specialists in all Sharp software and hardware.

Sharpsoft Ltd, Crisallen House, 86-90 Paul Street, London EC2. Tel: 01 - 729 5588.

COMPUTAMART

AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...

HERTFORDSHIRE

NEWBRAIN & SANYO
HARDWARE & SOFTWARE
Printers: Epson, Canon, Juki etc. Monitors, Tape Recorders, Books, Expansions, CP/M. **Sanyo 550/555 Computers.** Access/Mail Order. Ask for details.
ANGELA ENTERPRISES
Tel: Stevenage (0438) 812439 anytime

SOUTH LONDON

CROYDON COMPUTER CENTRE
Authorised Acorn Service Centre
29a Brigstock Rd., Thornton Heath, Surrey. Tel: 01 - 689 1280
BBC, Acorn, Electron, Genie, Oric, Kaga Microvitek Zenith Monitors, OKI 80, 82A + 84 Printers. Paper, Ribbons, Software etc. BUY-HIRE.

CHESHIRE

Computer Junk Shop
We Buy, Sell, Break Computers & Peripherals
10 Waterloo Rd, Widnes, Halton. Tel: 051 420 4590

LONDON

LEABUS
legal and buslnes software
Specialists in wordprocessing systems (multi-lingual wordprocessors etc) based on the Apricot Computers.
Open 9am-6pm. Telephoe anytime.
114 Brandon Street, London SE17 1AL.
Telephone: 01 708 2756.

TO ADVERTISE IN COMPUTAMART RING CAROLINE ON 01 437 0699

MIDDLESEX

SCREENS MICROCOMPUTERS
6 Main Ave., Moor Park, Northwood, Middx.
Tel: Northwood (09274) 20664
Telex: 923574 ALACOL G.
Official Dealers for: Acorn, Atari, Amstrad, Apricot, Commodore, Dragon, Einstein, Memotech, Oric, Psion, Sirius, Sanyo & Sinclair.
Open 6 days per week

NORFOLK

ANGLIA COMPUTER CENTRE
88 St Benedicts Street, Norwich.
Tel: (0603) 29652/26002.
Open: 6 days 9am-5.30pm.

SUSSEX

GAMER
24 Gloucester Road, Brighton.
Tel: 0273-698424.
Open: Mon-Fri 10am-5.30pm, Sat 9am-5.30pm.

TYNE AND WEAR

HCCS ASSOCIATES
533 Durham Rd., Low Fell, Gateshead. Tel. Newcastle 821924.
Open: 6 days 9am-5.30pm (Sat 10am-5.30pm). Specialists in: Acorn, BBC, Video Genie, VIC 20.

COMPUTING TODAY

Lineage: 40p per word.

Semi display: £9.00 per single column centimetre
Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

01-437 0699

Send your requirements to:
CAROLINE FAULKNER
ASP LTD, 1 GOLDEN SQUARE,
LONDON W1.

SOFTWARE

TURBO PASCAL
Extended Pascal for PC DOS, MS DOS, CP/M£86 and CP/M-80, includes full screen editor, floating point arithmetic, full string handling feature, random access data files, complies faster than IBM or MT + Pascal, requires less than 35K of disk space, 250 page manual and FREE spreadsheet program written in Turbo Pascal
"ONLY £54.95"
All prices fully inclusive for prepaid orders.
CONGUIN SOFTWARE,
14 GOODWOOD CLOSE, MORDEN,
SURREY SM4 5AW.
No callers please Phone 0524 381423

COLOUR GENIE owners quiz game £2.99 coming soon — Quiz Master £4.99, Fires of Mordor, 41 Pexwood Rd, Tormorden, Lancs.

CONFUZION
BY INCENTIVE
£6.95

DISKS

DISKS
3M—TDK—BASF

SS DD 40T	14.64
DS DD 40T	19.99
SS DD 80T	21.25
DS DD 80T	25.25

All Prices Box 10 And Include VAT & P-P.
Send cheque stating:
Qty, Brand and Type to:
CAROUSEL TAPES
"Disks", 3 Park Parade,
Stonehouse, Glos GL10 2DB.
Tel: 0453-82-2151

ALARMS

BURGLAR ALARM Equipment. Please visit our 2,000 sq. ft. showrooms or write or phone for your free catalogue. CWAS Ltd., 100 Rooley Avenue, Bradford BD6 1DB. Telephone: (0274) 731532.

REPAIRS

HOME COMPUTER REPAIRS
Look at our fantastic prices on repairs!

BBC B	£27.50
VIC 20	£20.00
COMMODORE 64	£27.50
DRAGON	£30.00
ORIC/ATMOS	£25.00
ZX SPECTRUM	£17.25
ZX INTERFACE	£17.25
ZX MICRODRIVE	£17.25

PLUS OTHERS!
The above prices are inclusive of parts, labour, P&P. All repairs carry 6 months warranty on replaced parts. Extended warranties, peripheral repairs, upgrades etc. etc. All available. Ring for full details (0234) 213645.
ZEDEM COMPUTER LTD
2 Kimbolton Rd, Bedford.

TO FILL THIS SPACE RING 01 437 0699

MISC

FOR MEN! FREE DETAILS
VASECTOMY
Everlasting alternative to the Pill.
No fuss. No waiting lists. One visit. Low cost. In
BIRMINGHAM LONDON PLYMOUTH
EDINBURGH MANCHESTER SOUTHAMPTON
GLOUCESTER NEWCASTLE SWANSEA
LEEDS NORWICH TAUNTON
LIVERPOOL
Write or phone for booklet (plain cover)
LONDON 01-388 2585
LEEDS 0532 440685
MANCHESTER 061 832 4260
Name_____
Address_____
_____ CT 9
Marie Stopes House,
108 Whitfield St., London W1P 6BE
Caring clinics since 1925
MARIE STOPES SEND NOW

COURSES

LEARN TO USE your computer on a weekend/holiday course. Details from:- Jaysoft MICRO Developments, 2 Wester Row, Greenlaw, Berwickshire.

COMPUTING TODAY
CLASSIFIED ADVERTISEMENT — ORDER FORM

If you have something to sell now's your chance! Don't turn the page — turn to us!
Rates of charge: 40p per word per issue (minimum of 15 words) + 15% VAT. Please state classification and post to: **COMPUTING TODAY, CLASSIFIED DEPT., 1 GOLDEN SQUARE, LONDON W1.**

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss/Ms) .....
(delete accordingly)
Address .....
.....
.....
Signature ..... **Date** .....
Daytime Tel. No. .....

Please place my advert in **COMPUTING TODAY** for . . . . . issues commencing as soon as possible.

OVER 220 AMSTRAD CASSETTE TITLES IN STOCK

OVER 110 *NOW* AVAILABLE ON DISC

CPM SOFTWARE
Macro 80, Microsoft Basic, Microsoft Basic Compiler, Turbo Pascal, BBC Basic (Z80), Purchase Ledger, Payroll, Database, other titles on request.

● TAPE TO DISC TRANSFERS ●

HARDWARE
CPC464 3" Disc, Timatic 5¼" 2nd Disc @ £149, CPC664 now available and Timatic 5¼" 2nd Drive @ £149. 3" 2nd Drive £99.

RS232 INTERFACES
Full spec. dual RS232 £59, Full Board includes RS232 and Rom Software, 8 bit parallel printer port, BBC compatible user port £89.

Also available Maxam Assembler & Quma Assembler.

Mail order welcome. Please send sae for full list to:

TIMATIC SYSTEMS LTD

NEWGATE LANE
FAREHAM, HANTS PO14 1AN
Tel: FAREHAM (0329) 239953

FAREHAM MARKET
FAREHAM, HANTS
Tel: FAREHAM (0329) 236727

SPECIAL OFFER

3½" FUJI Microdisks

SS/DD **£29.95**
DS/DD **£39.95**

5¼" FLOPPY DISKS

Suitable for use on nearly all single and double-sided 5¼" disk systems.
- Replacement Guarantee
- Hub rings
- Boxed in 10s

PRICES PER BOX

	1-4	5-9	10+
DS/DD	9.90	9.00	8.60

99P EACH
(Boxed in 10s)
EXCL VAT

PRICES PER BOX

	1-4	5-9	10+
SS	29.95	29.50	29.10
DS	39.95	39.30	38.70

Full Lifetime Guarantee

REPLACEMENT GUARANTEE
If any disk should fail, return it for free replacement. .

SEE10 LIBRARY CASES
(Hold 10 disks)

	1-4	5-9	10+
5¼"	1.99	1.85	1.75
3½"	1.99	1.85	1.75

DELIVERY

	1-4	2-4	5+
Disks (per box)	75p	50p	40p
Library Cases (Free with disks)	60p	30p	25p

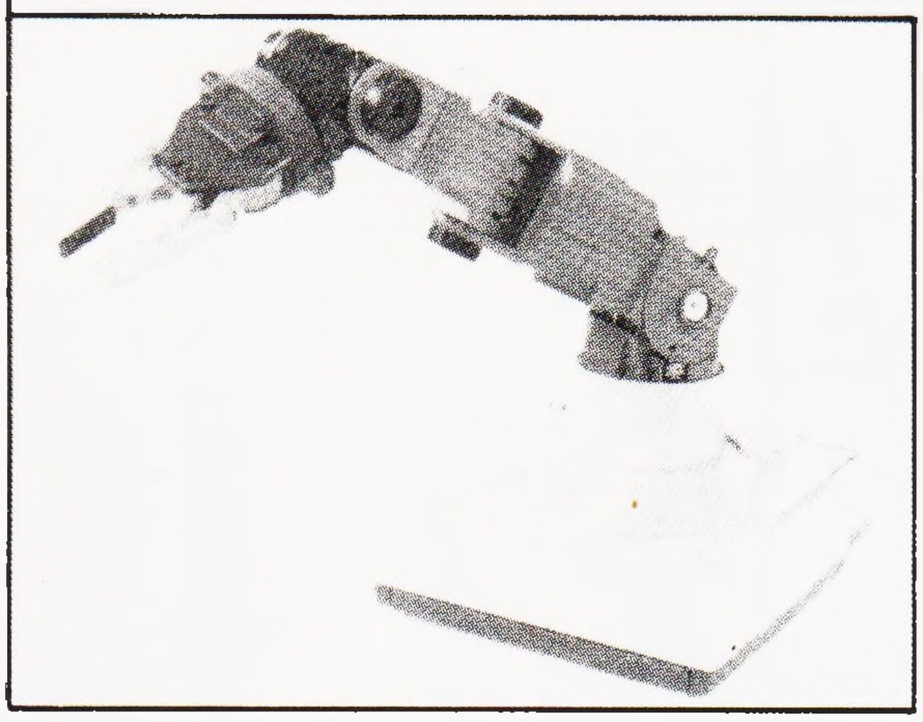
HOW TO ORDER

To total order value add Delivery, then add 15% VAT and send to:

IDS Computer Supplies

P.O. BOX 436, BANCROFT, MILTON KEYNES MK13 0QX. Tel: (0908) 310896.

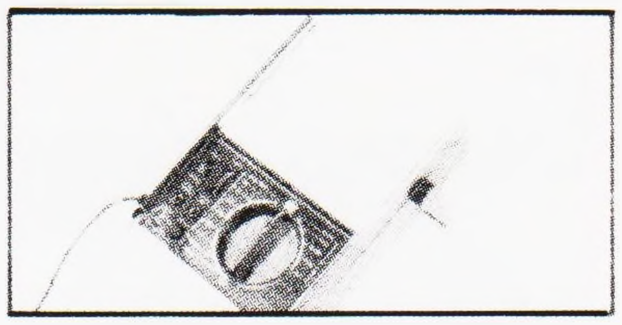
Make more things happen with Memoco.



Memoco Electron Robotic Arm

12 Axis of movement. Arm raise and lower. 270 degree rotation left or right. 90 degree Elbow movement left or right. 90 degree wrist movement either side of centre. 270 degree wrist rotation in either direction. Claw open and close. Fitted with motor control circuit. Switched from 5 volt TTL. Controlled by computer. Separate motor driver power supply.

With position feedback **£129.95**
BBC B Interface card **£49.00**
Spectrum Interface card **£79.00**
Commodore 64 Interface card **£49.00**



200 in 1 Electronic Lab Kit

Includes all parts to make 200 projects such as Radio · Rain Detector · Burglar Alarm. Covers projects using Transistors · Integrated Circuits · Seven segment displays. Light Sensitive circuits and many more. All components built into fitted workcase with cover. Comprehensive manual. Completely safe.

Normal Price £34.00
Our Price **£24.95**

Range Doubler Multitester 43 Ranges

50,000 Ohms per volt DC. 10,000 Ohms per volt AC. 4.25" Colour coded mirrored display giving accurate reading without parellex error.

Normal Price £27.00
Our Price **£15.45**

Ni-Cad Battery Charger

A.A., C. D and PP3. Charges up to five batteries at a time.

Price **£6.95**

Battery Eliminators

Mains to DC regulated 300 mA 6v, 7.5v, 9v selectable. Suitable for most battery operated equipment.

Price **£6.95**

QuickShot™ II Joystick **£9.50**

1½–4½ v DC Motors Stall Current 300 mA
Dimensions: Length o/a: 45 mm; Dia: 24 mm; Shaft length: 5 mm; Shaft Dia: 2 mm
Price – **£1.50** each or 10 for **£12.50**

ALL PRICES ARE INCLUSIVE OF VAT – FOR ORDERS UNDER £5 ADD £1.00 POSTAGE AND PACKAGING. FOR INFORMATION ONLY PLEASE SUPPLY S.A.E. TRADE ENQUIRIES WELCOME

MEMOCO ELECTRON

15 WINDSOR STREET
MELTON MOWBRAY, LEICS
TELEPHONE (0664) 63544
DEPT CT

ATARI ST

520ST

POWER WITHOUT THE PRICE

THE NEW ATARI 520ST

Under the new leadership of Jack Tramiel (former boss and founder of Commodore Business Machines), Atari Corporation have marked their entry into the world of business/personal computers with a machine which leaves the competition standing. Tramiel's slogan 'Power Without the Price' has been implemented in the manufacture of the new 512K Atari 520ST colour computer which offers the user amazingly high performance at an incredibly low price. Launched as a work-station, this new system incorporates seven software packages as well as the 520ST computer with 512K RAM, mouse controller, high resolution monochrome monitor (640x400), 95 key keyboard (with 18 key numeric keypad), MIDI interface, GEM and a 500K 3½ inch disk drive, all for the package price of only £651.30 (+VAT = £749). Dubbed the 'Mac beater' and the 'Jackintosh' (after Atari's Chief, Jack Tramiel), Atari's new machine has been directly compared with the Apple Macintosh RRP £2595 (+VAT = £2985) which offers similar features and capabilities but at a much higher price. Favourably reviewed by the UK's highly critical specialist computer press, the 520ST is likely to make a great impact in this country as a sophisticated alternative to an IBM PC, APRICOT or APPLE MACINTOSH. Unlike its overpriced competitors, the Atari 520ST can be linked up to a colour monitor to unleash a choice of up to 512 colours. The addition of colour brings out the full potential of graphics packages such as GEM.

USER FRIENDLY GEM OPERATING SYSTEM

The power of the ST is harnessed and made user friendly by the new operating system 'GEM' from Digital Research. GEM stands for Graphics Environment Manager and allows a user friendly colour or B/W graphics interface which closely resembles that of the Macintosh. This similarity extends to the use of moveable resizeable windows, icons to represent objects such as disks and disk drives, and the use of pull down menus and a mouse. The advantage of all this is that the computer becomes extremely easy to use. GEM has now been implemented for the Acorn, ACT, Atari, IBM, ICL and Olivetti. Software written for GEM on one computer should also run under GEM on another computer. This will enable the market to quickly produce a large library of standard interchangeable software.

FREE SOFTWARE AND FUTURE EXPANSION

The Atari 520ST comes supplied with seven free software packages as listed below. 1) TOS - Tramiel Operating System based on CPM 68K. 2) GEM Graphics Environment Manager by Digital Research (DR) giving a WIMP (Window, Icon, Mouse, Pull down menu) environment. 3) DR GEM Paint for creating graphics masterpieces. 4) DR GEM Write for word processing. 5) Logo learning language to enable you to write your own programs easily using turtle graphics. 6) DR Personal Basic a powerful user friendly version of the Basic programming language. 7) BOS operating system giving you access to dozens of business applications packages already available on the market. Designed with future expansion in mind, the ST also features a host of convenient interfaces to the outside world and an impressive list of accessories is planned. Atari will soon be releasing a 1000K (1MB) 3½ inch disk drive, and a 15MB hard disk storage system as well as a mass storage compact disk (CD) player capable of storing an entire 20 volume encyclopedia on one disk. A full range of inexpensive printers are planned including dot matrix, daisywheel and thermal colour printers. With its unbeatable graphics, speed and software at a price which is far below that of any comparable personal computer currently on the market, the ST is all set to do battle with the competition. To receive further details of the ST from Silica Shop, just fill in the coupon below with your name and address details and post it to us.

Silica Shop Price: £651.30 - £97.70 VAT = £749.00 This price includes:

- ***512K RAM***
- ***B/W MONITOR***
- ***MOUSE***
- ***500K 3.5" DISK DRIVE***
- ***GEM***
- ***KEYBOARD (95 KEYS)***

£749

ATARI 520ST SPECIFICATION

MEMORY
512K RAM (524,288 bytes)
16K ROM expandable to 320K
Port for addni 128K plug-in ROM cartridges
200K TOS operating system

GRAPHICS
Individually addressable 32K bit-mapped screen with 3 screen graphics modes:
320x200 pixels in 16 colours (low resolution)
640x200 pixels in 4 colours (med resolution)
640x400 pixels in monochrome (high res)
16 shades of gray in low res mode
512 colours available in low/medium res
8 levels of each in red, green and blue

ARCHITECTURE
4 custom designed chips:
GLUE Chip - MMU Memory Mngmnt Unit
DMA Controller - Graphics Processing Unit
16/32 bit Motorola 68000 processor at 8MHz
eight 32 bit data registers
eight 32 bit address registers
16 bit data bus/24 bit address bus
7 levels of interrupts/56 instructions
14 addressing modes/5 data types

DATA STORAGE
High speed hard disk interface
Direct memory access 1.33 Mbytes per second
CD (Compact Disc) interface
Built in cartridge access
Dedicated floppy disk controller

DISK DRIVE
500K (unformatted) 3½" sided floppy disk drive
349K (formatted) storage capacity

SOUND AND MUSIC
Sound Generator
Frequency control from 30Hz to above audible
3 voices (channels) in wave shaping sound in addition to a noise generator
Separate frequency and volume controls
Dynamic envelope controls
ADSR (Attack, Decay, Sustain, Release)
Noise generator
MIDI interface for external music synthesizers

KEYBOARD
Separate keyboard microprocessor
Standard QWERTY typewriter styling
Ergonomic angle and height
95 keys including 10 function keys
Numeric keypad — 18 keys including ENTER
One touch cursor control keypad

MONITOR
12" screen - high res monochrome monitor
640x400 monochrome resolution

VIDEO PORTS
Display - Low Resolution - 40 columns
Med/High Res - 40/80 plus cols
Medium res RGB (Red/Green/Blue) output
High resolution monochrome (Black & White)

COMMUNICATIONS
Bidirectional centronics parallel interface for printers, or modems capable of input/output
RS232C serial modem/printer interface
VT52 Terminal Emulation Software
Maximum Baud Rate up to 19,200
High speed hard disk interface
Floppy disk controller (Western Digital)
2 joystick ports (one for 2 button mouse)
MIDI interface for external music synthesizers

GEM WIMP ENVIRONMENT
WIMP - Window Icon Mouse Pop-down menus
Two button mouse controller
Icons/Pull down menus/Windows
GEM VDI - Virtual Device Interface
GEM AES - Application Environment Services
GEM BBT - Bit Block Transfer
Real time clock & calendar

SOFTWARE
GEM environment
with user friendly Macintosh style operation
TOS - Tramiel Operating System
Atari's own system based on CP/M 68K with hierarchical directory & file structure plus a host of MS-DOS & UNIX command structures
BOS - Business Operating System
to run any standard BOS business programs
GEM desktop
with GEM PAINT graphics mgmt system and GEM WRITE word processor
Personal BASIC and DR Logo
originally written by Digital Research (DR) Very much like those on other machines except for the extensive use of pull down menus, mouse control and windows

VARIOUS
Dimensions: 470mmx240mmx60mm
Replaceable external power supplies
Expansion: 3½" floppy disk drives 500K/1.000K (two drives can be connected)
3½" 15Mb hard disk
CD (compact laser disc)
Dot matrix & daisywheel printrs (black)
Thermal dot matrix (colour)
RGB & monochrome monitors

LANGUAGES
BASIC & LOGO supplied
Many others will soon be available, including Assembler, BCPL, C, Cobol, Compiled Basic, Lisp, Modular-2 and Pascal

Note: Some of the above specifications are pre-release and may therefore be subject to change.

MACINTOSH v F16 v 520ST

"Imagine a Fat Mac - the 512K Apple Macintosh - but with a bigger screen, a far bigger keyboard with numeric keypad, cursor and function keys, and colour. That gives you some idea of what the Atari 520ST is like, except for two important things. First the Atari seems faster. Second the Atari system is about one third of the price."
June 1985 - Jack Schofield - **PRACTICAL COMPUTING**

FEATURES OF BASIC SYSTEM	APPLE MACINTOSH	APRICOT F1e	ATARI 520ST
Price includes B/W Monitor	YES	NO - extra £200	YES
Keyboard size mm (LxDxH)	330x147x50	450x167x28	470x240x60
Keyboard size ins (LxDxH)	13x5¾x2	17½x6½x1	18½x9½x2½
3½" D/Drive (Unformatted)	500K	500K	500K
3½" D/Drive (Formatted)	399K	315K	349K
WIMP (Window, Icon, Mouse, ..)	Apple	ACT - Activity	GEM
Real-time Clock	YES	YES	YES
Polyphonic Sound Generator	YES	NO	YES
RS232 Serial Port	YES	YES	YES
Centronics Parallel Printer Port	NO	YES	YES
Dedicated Floppy Disk Controller	NO	YES	YES
Hard Disk DMA Interface	NO	YES	YES
Full stroke keyboard	YES	YES	YES
Number of keys on keyboard	59	92	95
Numeric Keypad	NO	YES (16 Keys)	YES (18 keys)
Cursor Control Keypad	NO	YES	YES
Function keys	NO	10	10
16-bit processor	68000	Intel 8086	68000
Processor running speed	8MHz	4.77MHz	8MHz
RAM size	512K	256K	512K
Number of graphics modes	1	4	3
Number of colours	Monochrome	16	512
Max Screen Resolution (pixels)	512 x 342	640 x 256	640 x 400
Mouse included	Single Button	NO - extra £95	Two Button
Replaceable External Power Pack	NO	NO	YES
Cartridge Socket	NO	NO	YES
Joystick Ports	NO	NO	YES (two)
MIDI Synthesiser Interface	NO	NO	YES
Monitor Size	9"	8" - extra £200	12"
RGB Video Output	NO	YES	YES

System Cost with: Mouse - Monochrome Monitor - 512K RAM - 500K Disk Drive

Price of basic system (exc VAT)	£2595+VAT	£595+VAT	£652+VAT
+ Mouse	Included	£95+VAT	Included
+ Monochrome Monitor	Included	£200+VAT	Included
+ Expansion to 512K RAM	Included	£295+VAT	Included
Price of complete system (exc VAT)	£2595+VAT	£1185+VAT	£652+VAT
PRICE rounded down including VAT	**£2,984**	**£1,362**	**£749**

PRESS COMMENT

"Atari's new corporate image as an aggressive low cost computer maker is likely to mirror that of Commodore where Mr. Tramiel established the maxim that 'Business is war'."
August 21st 1984 **FINANCIAL TIMES**

"This is the only personal computer I know of that comes with a Midi interface as standard."
Peter Bright March 1985 **PERSONAL COMPUTER WORLD**

"The (GEM) version running on the Atari 68000 machines will have the additional advantage of leaving the PC version standing."
April 6th 1985 **PERSONAL COMPUTER NEWS**

"It would seem that GEM offers the ideal operating system."
March 7th 1985 **POPULAR COMPUTING WEEKLY**

"I found it (GEM) extremely easy to use and was very impressed with the way in which it disguises the unfriendly hardware and operating systems lurking under the surface."
Peter Bright Feb 1985 **PERSONAL COMPUTER WORLD**

"The electronics in the machine are a work of art ... The heart of the 520ST is a Motorola 68000, one of the most powerful 16-bit processors around and in many respects it is close to being a 32-bit chip ... when the machine appears in the shops, I'll be at the front end of the queue to buy one."
Peter Bright June 1985 **PERSONAL COMPUTER WORLD**

"This machine is significantly more powerful than an IBM PC ... If it's possible to design a sure-fire winning machine, this is it."
May 11th 1985 **PERSONAL COMPUTER NEWS**

"... the use of GEM makes the new range of Atari computers so similar to the Macintosh (with the added attraction of colour), that they are already being called 'Jackintoshes'."
May 2nd 1985 **COMPUTING**

"The new Atari ST computers truly represent to the consumer what Jack Tramiel is saying — easy-to-use computing power without the price."
March 1985 **ANALOG COMPUTING**

"It (the ST) uses the most modern technology that is affordable, in a package that gives a professional impression."
May 23rd 1985 **POPULAR COMPUTING WEEKLY**

"The Atari ST is one of the most elegant designs I have seen ... Atari has used an original and elegant method of memory management which should make the ST faster than any other PC on the market - in any price bracket ... The 64K dollar question is would I go out and spend money for one? To which the only answer is 'Try and stop me!'."
John Lambert July 1985 **ELECTRONICS & COMPUTING**

"The 520ST is technically excellent ... The 520ST hardware is the new standard by which others will be judged."
July 1985 **YOUR COMPUTER**

SILICA SHOP

WE ARE THE UK's No1 ATARI SPECIALISTS

At Silica we have been successfully dedicated to Atari ever since their products first appeared on the UK market. We can attribute our success largely to the Atari specialisation which we practice and to the user back-up we provide. Rest assured that when you buy a piece of Atari hardware at Silica you will be fully supported. Our mailings giving news of software releases and developments will keep you up to date with the Atari market and our technical support team and sales staff are at the end of the telephone line to deal with your problems and supply your every need. With our specialist bias, we aim to keep stocks of all the available Atari hardware, software, peripherals and accessories. We also stock a wide range of Atari dedicated books and through us, the owners on our list can subscribe to several American Atari dedicated magazines. We can provide a full service to all Atari owners and are now firmly established as the UK's NUMBER ONE Atari specialists. Here are just some of the things we can offer to our customers:

- **FREE POST & PACKING ON MAIL ORDERS**
- **FREE NEXT DAY SECURICOR DELIVERY**
- **INFORMATION MAILING SERVICE**
- **TECHNICAL SUPPORT TEAM**
- **HIGHLY COMPETITIVE PRICES**
- **AFTER SALES SUPPORT SERVICE**
- **REPAIR SERVICE ON ATARI PRODUCTS**

If you would like to be registered on our mailing list as an Atari computer owner, or as a person interested in buying an Atari machine, let us know. We will be pleased to keep you up to date with new Atari developments free of charge. So, return the coupon today and begin experiencing a specialist Atari service that is second to none.

SILICA HOTLINE 01-309 1111

SILICA SHOP LTD, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

SEND FOR FREE ATARI ST LITERATURE

To: Silica Shop Ltd, Dept CT 0985, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

PLEASE SEND ME FREE LITERATURE

ON THE NEW ATARI 520ST COMPUTER

Mr/Mrs/Ms: Initials: Surname:

Address:

Postcode:

Do you already own a computer?
If so, which one do you own?