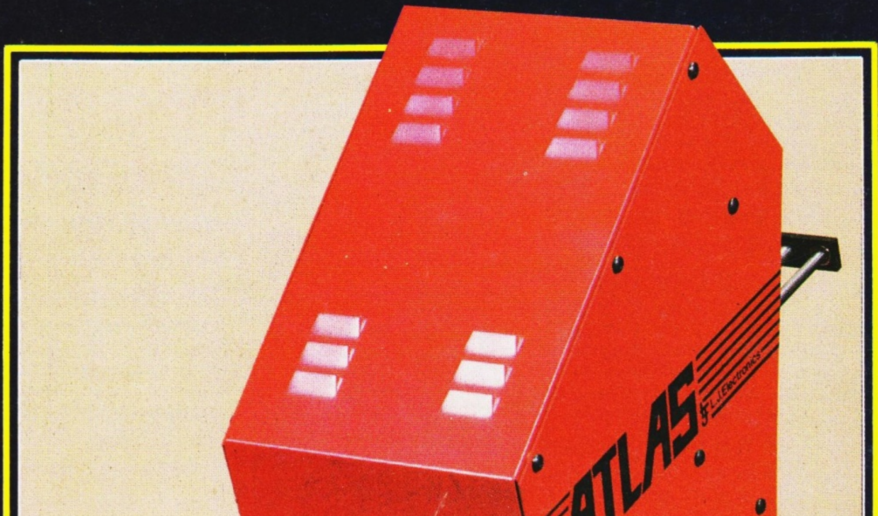
AN ARGUS SPECIALIST PUBLICATION

June 1985 £1.00

COMPUTING Today

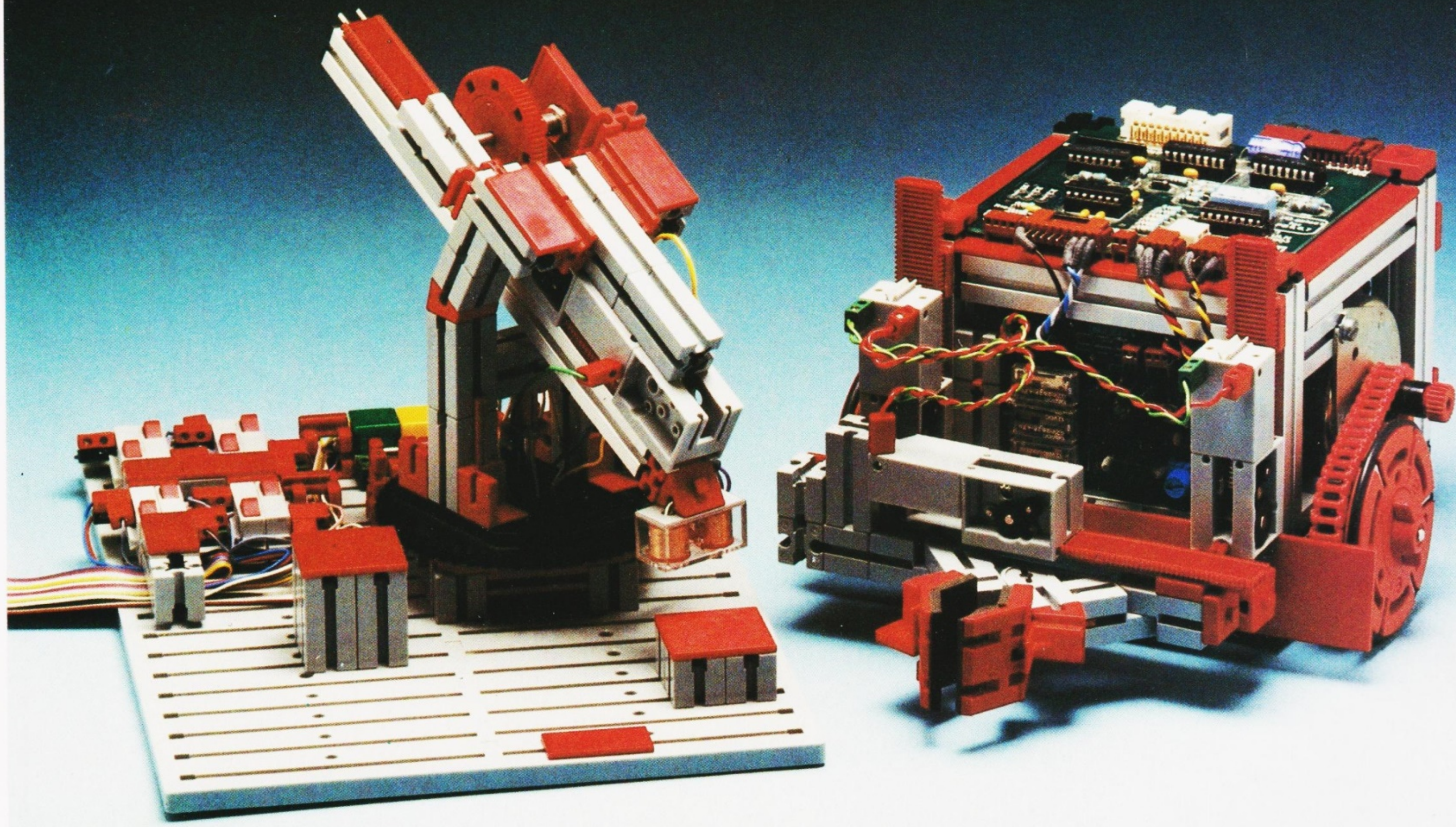


ROBOTICS
One step forward, two steps back?

THE ALVEY PROJECT
Closing in on the fifth generation

POLAND
Micros in eastern Europe

AN ACADEMIC APPROACH
Pointing out the pitfalls of analysis



COVER

Robotics is a subject that enjoys regular but brief 'flares' of popularity, but is there something fundamentally wrong with our approach to mechanisation? Mimicking human systems is not the way, suggests Bill Horne. Cover story: page 39.

CONTENTS

VOL 7 NO 6 JUNE 1985

REGULARS

SERIES

GENERAL FEATURES

SPECIAL FEATURES

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

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, London 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.

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 ....

NEWSNEWS

PROGRAMMERS WANTED FOR DEVELOPING COUNTRIES

Voluntary Services Overseas is looking for 10 computer programmers to take up posts for two years or more in the less developed countries.

Volunteers, male or female, will be required to train local people and will probably work in medical research, census information or financial records.

One post is for a programmer at the Lilongwe Agricultural Development Department in Malawi. The job, after an initial period working on the financial side, will be with the evaluation and planning section, where the volunteer will implement programmes on agricultural survey data, streamline existing programmes and train staff.

The university of Assiut, Egypt, has asked for a programmer to design and run programmes processing community health data, train four graduates to take over the work and advise generally on the BASIC language. Assiut is the only medical faculty in Upper Egypt and has an excellent record in promoting community approaches to health care.

Brian Passman of VSO's Business and Social Development Department commented: "One does not automatically think of aid workers being involved in computing but there is a need for this skill too. Volunteers should have a relevant qualification, two years' experience, as well as tact and sensitivity. They should share VSO's concern to combat exploitation in any form and to enlarge people's independence by helping them to develop their skills. VSO is interested in applications from minority groups.

VSO recruits people between 23 and 65 for posts in 40 less-developed countries. They work for a two-year period and received accommodation and a small allowance based on local rates of pay. Applicants should have no dependents though couples may be accepted if there are posts for both of them. Further details from Enquiries, 9 Belgrave Square, LONDON SW1 or Tel: (01) 235 5191. □

A GROWING MARKET

Propagating plants can be a time consuming process, especially if, like Blooms Nurseries of Bressingham, Norfolk, you're producing in the region of five million blooms annually.

So to help them Blooms have installed an NCR Decision Mate V PC to monitor 'work in progress' on each plant and seedling from the first moments of growth to the moment of sale.

It has become essential to know what stage of growth certain plants have reached, what treatments they've received and importantly, what treatments they should be given. The DMV has therefore come to play a vital role performing a number of specific tasks essential to the blossoming of their business.

Pesticide control is just one such area and John Adlam, Crop Services Manager, who has spent the last year programming the DMV, can now quickly and easily identify the most efficient pesticide for a particular batch or field, given data such as type of plant, number of hectares and what stage of growth the plants have reached. The correct mixture of fertiliser and cost per acre is computed in a similar manner.

All this is a step towards the development of a scheme to computerise the whole growing and selling process. Blooms hope to link up the DMV to their NCR I-9020 mainframe, interfacing it with the main stock file and updating it automatically. Blooms expect the scheme to take two years before it is fully functional. □

BP LAUNCHES EDUCATIONAL RESOURCE ON RML MICROS

Teachers who want to locate information on cross-curricular materials can now do so easily and quickly on a microcomputer, following the launch of BP Educational Services Electronic Catalogue.

Based on the British Library's Schools Information Retrieval (SIR) software package running on Research Machines Ltd range of microcomputers, the catalogue gives fast access to details of more than 200 British Petroleum educational resources and films.

"Until now, teachers who wanted to identify suitable materials for interdisciplinary projects would have had to read through assorted conventional printed catalogues. With a computer program they can do the job more quickly and with greater efficiency and flexibility," said Mike Fischer, Managing Director of Oxford-based Research Machines Ltd.

BP Educational Service is offering the BPES Electronic Catalogue free to any teachers' centre or media resource centre in the country. The catalogue, which comes with complete instructions, is free of copyright and can be duplicated by the centre for distribution to local schools and colleges. A free starter pack is also available to centres which have no previous experience with BP's range of industry-related educational materials.

"This is the first time BP's wide range of educational resources can be accessed by computer. Initially, we are releasing the electronic catalogue to run on the RML 380Z, RML 480Z, and RML CHAIN Network. RML machines are running in more than 75 per cent of UK secondary schools and colleges," said Dave Barnett, Manager BP Educational Service. "We plan to issue versions of the catalogue later in the year to run on other micros."

The BPES Electronic Catalogue, which has been developed in conjunction with the Teaching Resource Materials Service, Ampthill, Bedfordshire, requires no computer expertise. Its easy-to-follow instructions allow access to BPES resource files, and can be searched by title, subject, skill and other combinations of headings. Permanent record printouts can also be distributed to other users. Every six months, BPES will update the service.

A free copy of the BPES Electronic Catalogue is available to teachers centres and media resources by contacting BP Educational Service, PO Box 5, Wetherby, West Yorkshire, LS23 7EM; or telephone Boston Spa (0937) 843477. □

SCARED OFF BY SALES TALK

Everybody knows, or should know, that emissions from VDUs are harmless. So why is 'umbrella' marketing outfit, ASTAR, flogging radiation-proof aprons for users of VDUs?

In an extremely HighTech looking brochure entitled 'New Products For Safety In The Modern Office', ASTAR (who also handle the Flat Card memory featured elsewhere in our News) have managed to produce a masterful example of applied sales-psychology full of questions and suggestive conjecture, which looks to have been designed with the intention of scaring people into buying their products.

Here are a few prime examples of how the brochure describes the computing environment:

> "Video display terminals emit various forms of radiation, including X-rays, microwaves, and infrared light. they also cause a strong electrostatic charge to develop in the air in front of the screen."

> "... a strong charge" (*strong* charge, mind you) "...of up to 50,000 volts per square inch."

> "Some authorities believe that the nervous system may be directly affected by ion imbalances in the human body, leading to tension, irritability, and depression."

> "Aspects of VDT (Video Display Terminal) operator health which are currently the subject of serious scientific study include: eyestrain, blurred vision, double images, headaches, dizziness, nausea, neck and back pains, facial rashes, tensions and other stressrelated problems, pregnancy dangers.'

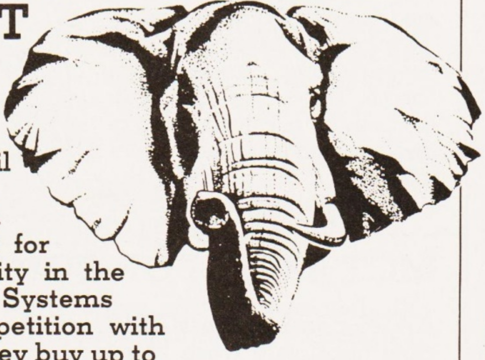
Conveniently enough, the **Royal Society for the Prevention of Accidents** (RoSPA) has produced a twelve-page booklet entitled 'VDUsers', which more-or-less explains why you *don't* need a radiation-proof apron. The booklet emphasises that radiation from VDUs does *not* cause epilepsy or skin rashes, nor are pregnant women at risk from exposure to VDUs. In fact, the booklet points out that radiation levels from VDUs are not only below the stringent safe legal limits, but that they are even below the totally harmless levels present in the general environment.

We rest our case...

Radiationproof aprons available from: Astar International (UK) Ltd., Heathrow Business Centre, Terminal 2, Heathrow Airport, London TW6 1EU. Tel. (01) 936 9177

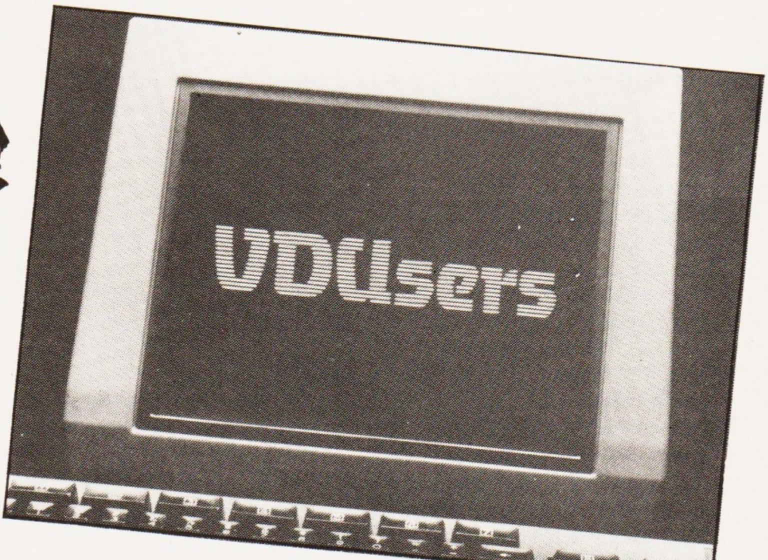
VDUsers booklet available from: Sales Department, RoSPA, Cannon House, Priory Queensway, Birmingham B4 6BS. Price: 70p. □

WIN KENYA SAFARI WITH ELEPHANT



Purchasers of 10disks Elephant packs are on the trail of a Kenyan safari. Starting this month, they will be eligible for an entry opportunity in the Elephant Memory Systems Kenya Safari competition with each 10disk pack they buy up to 30th June.

There are four super two-person safaris to be won with luxury accommodation and travel (on a 'Jumbo' jet?), all expenses paid. Entry is by means of a special puzzle card. Entrants have to complete the puzzle, write a slogan and return the card, together with the 10disk pack top, to Dennison. They will automatically receive a free Elephant 'T'-shirt as well as qualifying for entry to the competition. Judging of entries and the presentation of prizes will take place later this year. □



TATUNG DROP PRICE ON EINSTEIN

The price of the Tatung Einstein has been slashed by £150 to £350, pushing the machine into the same competitive arena as the BBC micro and the Amstrad CPC464.

Whispers from the Tatung camp suggest that further developments are on the way, including a new version of the Einstein, and a new peripheral that will allow the machine to run MSX software.

On the software side, Kuma Computers has announced a new communications package which will allow Einstein owners, armed with a suitable modem and Prestel/Micronet/Telecom Gold account, to Save, View, and Print viewdata screens.

Meanwhile, Xitan Ltd., one of Britain's leading software distributors, has been apointed to distribute CP/M software for the Einstein. Also, a version of the Microtext Authoring system - a computer based training package developed by the National Physical Laboratory - is to be made available shortly by Hampshire software house, Transdata. □

NEW GROUP TO MAKE ROBOTICS ACCESSIBLE

An authoritative robotics specialist group has been set up by The British Computer Society to lead the way through the increasingly hostile jungle of computing and robotics.

Chaired by Peter B Scott, robotics expert and author of a leading introductory textbook on the subject, the ten-strong organising committee will act as the focus for the Robotics specialist group, which will be open to everybody.

It is intended that the group will provide a major forum for the exchange and discussion of knowledge in all areas related to the use of computing in robotics. It is also planned to foster research and application of such technology, with attention to its social consequences, and to promote professional practice through education and training.

Committee Chairman, Peter Scott, comments: "Already robotics can be seen to be racing ahead into currently unimagined regions, and of all the disciplines which are aiding this development, computing is without doubt the most vital. Within the foreseeable future, the current cost of a comparatively cheap robot controller will buy power greater than that of today's largest mainframe computers. So it becomes vital that The British Computer Society, as the one chartered institution ideally suited to take a credible and objective stance on such a computer-driven technology, should be seen to do so, and in a leading way."

The committee has already taken these words to heart, and has managed to secure funding for a one-day seminar on the subject 'Robotics applications and implications' with accommodation for several hundred people at Imperial College, London on Wednesday 29 May. Some of the leading UK experts on robotics and allied fields will provide a highly accessible and comprehensive overview of all aspects of robotics. Such high-level seminars have been offered before, but the cost to each delegate of this one? £15.00, including refreshments.

Says Peter Scott: "We should all be *talking* to each other about robotics. Seminar costs should be such that representatives of small and medium sized firms *are able to justify* attendance to hear from and discuss with experts at first hand."

Further information can be obtained from Ms Catherine Wiles, Secretary, BCS Robotics Committee, Centre for Robotics, Imperial College, London SW7 2BX. □

INDEPENDENT USER GROUP FOR CPC 464

An independent postal user group dedicated to the Amstrad CP 464 has been set up by Nick Godwin, organiser of the ZX81 group, ZX EXCHANGE.

The group seeks to encourage home programwriting as a creative activity, by exchanging the experiences and expertise of amateur programmers of all ages from all over the country. it will also offer opportunities to make direct postal contact between amateur program writers who have particular interests in common.

The main forum for the group will be a newsletter, to be produced every two months. This will carry reviews of utility software and publications, information about magazines, books and other publications of special interest to CPC 464 users, and will report on other user groups and special deals that are available. It will also carry routines. The inaugural issue is now available and includes, among other routines, a simple technique to display a disc unspoilt by any interference pattern and not requiring the use of SIN and COS.

The group will appeal to anyone who is interested in writing software for the Amstrad computer, whether or not he or she has previous programming experience. Full details can be obtained by writing to the address below, enclosing two second-class postage stamps. A copy of the inaugural newsletter can be secured by enclosing a cheque or postal order for 50p, payable to Nick Godwin at 4 Hurkur Crescent, Eyemouth, Berwickshire, Scotland, TD14 5AP. □

'FLAT CARD' STORAGE NOW AVAILABLE

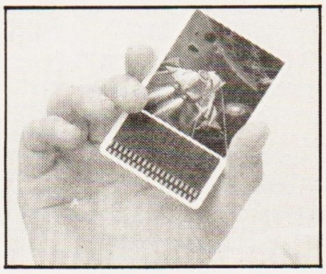
A new storage medium looks set to emerge as the convenience package for tomorrow.

The new medium, a flat plastic rectangle the size of a credit card, has been developed by Japanese company, Astar.

The Astron Card, as it's called, is a solid-state memory device that packs 32Kbtes of ROM into a surface no more than 3mm thick.

Astar see the card as an alternative to prepackaged ROM and cartridge software - and it certainly has advantages over these more conventional media. The card is small, thin and robust, alleviating most of the common 'storage' problems, and the surface of the card is ideal for carrying logos and other artwork associated with the software.

Astar say that by the end of this year four types of card will be available: Custom made masked ROM; small quantity EPROM; EEPROM and high-speed CMOS RAM with a built-in Lithium battery. □



MCT BASIC FOR COMMODORE 64

A C16-compatible BASIC for the Commodore 64 is now available from Micro Component Trading Co. (MCT) of Norwich.

The new BASIC contains a number of extensions not found in V2. or its contemporaries, and is in the form of a cassette complete with Turbo loader.

Incorporating a number of sprite and hires graphics commands, MCT BASIC is set to retail at £10.95 including VAT.

A ROM version will be available shortly, but a price has yet to be announced.

MCT are at Group House, Fishers Lane, Norwich, Norfolk NR2 1ET. Tel 0603 666966. □

BRITISH FIRMS AGREE NETWORK STANDARD

Following a survey of British microcomputer manufacturers, a report hs been published outlining a number of recommendations which could lead to the standardisation of local area networks and protocols.

The survey, commissioned by the British Microcomputer Manufacturers' Group (BMMG) in conjunction with the Department of Trade and Industry, highlights the need for an agreed LAN standard before such a standard is imposed by a single major manufacturer (IBM?).

More urgently, the report pinpoints the need for a definition of the 'local area network', emphasising that the success of any standard would require a promotional campaign to enhance manufacturers' awareness of it.

The report describes a six-phase implementation plan which would give manufacturers already producing LANS to their own design the chance to 'phase-in' the new standard. □

SCHOOLS COMPUTER PROJECT "MISGUIDED"

The heavily publicised Government sponsored M.E.P. project has been misconceived and has been wasteful, said Brian Drinkall, Managing Director of PMSL Mentor Ltd., the Bradford computer based training company.

"The computers installed in most of our schools are little more than toys. They're fun and you can teach children about computers with them, but they really aren't much use as teaching aids across the syllabus, and they don't prepare the youngsters for using computers in the real world".

Commenting on reports that the French government plans to install 120,000 micros of the size used in business, like the Apricot, Drinkall commented, "This is exactly the right way to go. Using modern techniques of computer based instruction such as Mentor II, teachers will give their pupils tremendous learning advantages across the full range of subjects."

Pointing out the difference between the French approach and ours, Drinkall summed up, "It's a bit like the British taking their kids out to lunch and restricting them to hamburgers. It's exciting, it's fun and they learn how to behave in public. Meanwhile, the French children are being offered the full choice of the 'menu', acquiring taste and learning to discriminate. Unless we do something about it, we will become a nation restricted by cheap, fast educational fodder when a little more consideration could result in a more varied and beneficial diet". □

MANAGEMENT'S JOB TO STOP FRAUD

Greater awareness by management of the additional risks posed by computerisation would significantly minimise computer fraud and misuse, says the Audit Commission in a report recently published.*

Its survey, involving public and private sector participants, revealed that few cases of computer fraud demonstrate ingenious technological skill; most take advantage of inherent weakness in particular procedures.

The report emphasises that while new entrants in organisations are increasingly familiar with computing, middle and senior management are invariably not. The new generation is ready to grasp the potential whereas others may fail to grasp the implications, it warns. To counter fraud and identify the safeguards required, the report urges management to adopt a more systematic approach.

Analysing the seventy-seven cases submitted in the survey, it highlights the following points:

- In all fraud the opportunity rather than the actual financial amount is the determining factor.
- Disturbingly, a large number of frauds are perpetrated by long serving employees in trusted positions, who divide almost equally between clerks and managers/supervisors.
- Three quarters of the reported frauds resulted from unauthorised submission and alteration of data. In many cases the opportunity was blatant and weaknesses were known.
- Use of computers for private work - by far the most publicised misuse - accounted for 22 per cent of reported cases. The increasing use of desk-top microcomputers will exacerbate this trend.
- In one instance of theft of computer-produced output, presigned bank giro cheque forms with a maximum face value of £931,000 were stolen.

Existing internal controls and audit systems led to the discovery of over half the cases, which underlines the need for such procedures. However, the fact that one third of all reported cases were discovered by other means, sometimes anonymous tip-offs, gives cause for concern. For the auditor to function effectively in the future, the Commission stresses that he or she will need to be fully "computerate", with equal accounting and computing knowledge.

Clearly, more computer frauds are committed than are discovered. Whilst recent reports have estimated enormous losses through fraud among financial institutions (which did not respond to the Commission's survey), the Commission believes the implications for the many organisations which did - and for their auditors - are in no way invalidated.

For a copy of the report or further information contact John Cheetham or Andrew Slaughter on 01627 0383. □

* *Computer Fraud Survey.* Published by HMSO price £3.30.

ECONOMATICS TAKE FISCHER-TECHNIC PRODUCTS

Distribution of the Fischer-form and Fischer-technik product range, including the exciting new computer controlled robot kits, has been taken over by ECONOMATICS (EDUCATION) LIMITED.

Already established as a major distributor of technical equipment for education, Economatics will now be supplying both the retail trade and other educational distributors with this product range.

Fischer-technik is the precision construction system with graded sets for different age groups. A simple version allows very quick assembly, while more complex sets allow very sophisticated models to be built. The latest addition to the range is the new robot kit, which can be used to build up 10 different computer controlled models ranging from simple traffic lights through to a two axis robot.

ECONOMATICS, a Sheffield-based company have many years' experience in developing manufacturing and supplying technology teaching equipment. From a sound base of industrial automation consultancy and service, the growth of the education equipment company has reflected the important changes in the UK School Curriculum.

The emphasis is on the technology of 'control' and the material is being constantly updated to include modern industrial techniques. Probably the fastest growing area is in the field of robotics and here Economatics have produced the British Broadcasting Corporation Buggy, a small mobile robot used to illustrate the use of the microcomputer in a control sitution. Further developments now include a Pen for the Buggy to enable it to draw shapes programmed by the computer and a Grabarm which makes the Buggy a working robot. □

EMPLOYMENT PROJECT FOR YOUNG DISABLED

A new employment project for the disabled has been announced by THORN EMI Computer Software, part of THORN EMI Information Technology Division.

Launched in association with COMET (Concerned Micros in Employment and Training), established by British Microcomputing Awards, the scheme will provide sponsorship by THORN EMI Computer Software of a trainee programmer role within the company's Altergo Products Division for a disabled young person.

The sponsored place will be awarded by competitive entry for which criteria will include academic achievement and programming suitability. Entrants should have at least a GCE 'A' level standard or equivalent of education and be able to demonstrate logical ability. Experience in computing would be useful but is not essential.

A COMET Award Panel will shortly be announced who will select and interview applicants. This panel will consist of specialists in the computing and disabilities fields. The successful applicant will be announced at the British Microcomputing Awards Ceremony on June 12th 1985 at the Park Lane Hotel.

The scheme is now open for applications and anyone interested in applying should initially contact: The National Bureau for Handicapped Students, 40 Brunswick Square, London WC1N 1AZ (Tel: 01-499 5195). Applications close on April 24. □

FERRANTI DEMONSTRATES INFERENCE ART

Ferranti Computer Systems Limited will be demonstrating 'Inference ART' on the Scientific Computer/Symbolics and Miles 33/LMI stands at the Artificial Intelligence and Simulation of Behaviour (AISB) Conference at Warwick on 10-11 April.

Inference ART is a comprehensive expert system development environment which integrates the latest techniques for the representation of knowledge and reasoning, and is considered to be well ahead of similar products. It has been used to develop expert systems for demanding realtime applications, including a system for NASA to control tracking data during space shuttle re-entry.

Ferranti, recently appointed as the main distributor in Europe and the Middle East, will provide all support, training and consultancy for users of Inference ART and is setting up a network of sales outlets. Ferranti has had experience of developing practical expert systems and is established as a supplier of artificial intelligence products. □

MAKING MONEY

Bulls, Bears, and Microcomputers is the title of a new book from Sigma Press.

Its author is none other than *Computing Today* contributor G T Childs, and the book is intended to show how your micro can help make the most of your investments.

Chapters include an introduction which outlines how computes can be linked used in a financial context; Risk and Reward ('where should you put your money?'); Jargon (a quick guide to investment language); Buying, Selling and Following shares and Organising your Portfolio. The book continues with several more chapters outlining the investment options open to you.

I was a little put off, though, by a sentence in the opening chapter that reads '. . . a person with a double-barrelled name and an expensive public school education may be a totally unfit person to invest in the market as he or she may never save a penny they earn.' Now, I have neither an expensive public school education, nor a double barrelled name. However, I cannot save for toffee. Maybe this book could provide me with the necessary incentive! (Send your donations to the Assistant Editor c/o *Computing Today*. Watch-out for a review in the near future.

BIBBY GROUP GOES INTO ROBOTICS

Cyber Robotics, the educational and industrial robot manufacturer, has moved from its Cambridge base to the West Midlands following its recent takeover by the giant Bibby Group plc.

It is now operating from the J Bibby Science Products Limited factory at Stone, in Staffordshire.

With an injection of capital plus new management, Cyber Robotics is now poised to make a renewed impact on the expanding robotics market.

The company's new top man is electronics graduate Paul Ritson, 31, whose previous job was with Dainichi - Sykes Robotics, supplying industrial robots for the re-equipping of production lines.

His first action on joining Cyber was to boost demand for the company's £650 educational robot by offering £200 worth of free extra software to buyers.

Cyber Robotics will be launching a new product at the Automan '85 Show, May 14-17, National Exhibition Centre. □

64s FOR P.M.

Commodore Business Machines (UK) Ltd recently presented the Prime Minister, the Rt. Hon. Margaret Thatcher MP, with the one millionth and two millionth Commodore Computers to be manufactured in Corby in March.

In the Prime Minister's private room at the House of Commons, 17 year-old Carmella Polcaro and 19 yearold Robert Gray presented Mrs. Thatcher with the computers, both Commodore 64s, on behalf of the workforce at Commodore UK's Corby factory.

Commenting on the presentation, Commodore UK's Acting General Manager, Arthur Scott, sad, "We are indebted to the Government for the support it has given us in setting up our new manufacturing facility in Corby.

"We are equally grateful to Mr. Powell for his continued encouragement and interest in our activites.

"Today was an opportunity for us to celebrate the production of two million Commodore computers in Britain and to present two of them to the Prime Minister in recognition of the investment which helped to make it possible." □

GOVERNMENT SOFTWARE GUIDELINES FOR EDUCATION

A book* published today for the government education departments' Microelectronics Education Programme (MEP) is likely to become an international standard reference work in the field of educational computing.

Education Software: a creator's handbook, edited by Cambridge-based Ken Alexander and Diana Blanchard of the MEP Software Unit, distils into one comprehensive volume the many lessons the Programme has learnt about producing software for schools to use.

Since its inception five years ago MEP has attracted world-wide interest in the range, quality and educational integrity of its teaching and learning materials. Overseas visitors are always interested in the approach the Programme has adopted towards ensuring that its published software is of the highest quality.

There are six sections in 'Educational Software': origination and design; coding; field trialling; software support; publication; software evaluation. Each section draws on the expertise of contributors from many areas of education and industry and represents some of the most authoritative views currently available.

The 36 pages of appendices give some useful contacts, examples of program listings, a model publishing agreement, machine limitations, etc. There is a glossary of terms that are peculiar to the fields being discussed.

Copies of the book are available for reference in the Programme's 14 regional information centres.

In the foreword to the book Richard Fothergill, director of the Microelectronics Education Programme, says: "One of the most difficult and yet exciting aspects of our modern age is the rapid and seemingly continual advance of technology. it is difficult because change always presents problems as we learn to accommodate to it, and exciting because the advances present us with new opportunities to help children and students learn. ☐

*Educational Software: a creator's handbook'. Edited by Ken Alexander and Diana Blanchard, MEP Software Unit. 276 pages, 240 × 165mm, illus. Distributed by Tecmedia Limited, 5 Granby Street, Loughborough, LE11 3LD. Price £25 plus postage and packing (£2.88 in the UK) with an introductory price to UK schools of £10 plus postage and packing.

STOP PRESS

By the time you read this the **Amstrad 664** will have been in the shops for a week or two, but this item is being written the afternoon of the press launch of the product — Such is the company's, and **Alan Sugar**'s (The MD) disdain for other computer manufacturers that present empty boxes at press launches.

I case you haven't seen it already, the 664 is the 464 with a 3" disc drive where the cassette unit is on the 464, a slightly modified case to accomodate this (and different colour scheme) plus a few modifications to the software. And herein lies a cautionary tale . . .

To the user, the modifications to the software will initially seem trivial. Besides slightly improved disc handling routines and some extra graphic commands, you won't notice a lot. However, a good deal of work has gone into the firmware; for a start, **Locomotive Software** — the people that actually wrote the firmware — say that the BASIC is faster; not much faster (a comparative figure of 13.1 seconds instead of 13.2 on an unspecified benchmark was floated), but every little helps.

However, the crunch comes for software written in machine code. As a result of the re-vamp, although the software is largely the same, it's all moved around in the computers memory map. In particular, the keyboard map is not

Amstrad MD, Alan Sugar, and the new Amstrad CPC664.

where it used to be! Now, at the time that the 464 originally appeared, Amstrad made a manual on the firmware available to programmers, and this, amongst other things, gave the location of the 'Firmware Jumpblock', ie, a table that told the programmer the address of the various useful bits of information.

Through the jumpblock, the programmer can get a keyboard *scan*, ie it can tell the last key pressed, but not a keyboard *map*, ie it doesn't tell you which other keys are being pressed. This resulted in many people digging around in the ROM to find the keyboard map, when writing their software package using these addresses. So unfortunately, their software will not work on the 664.

Amstrad, on the other hand, and all those writers who have observed the rules written into the firmware manual, are laughing, because their programs *will* work on the 664. The trouble is that it's the most creative programming that will fall foul of the change, although Amstrad do say tht any program which drives the machine direct, ie doesn't use any of the firmware, will have no problems.

The press launch provided an interesting insight into the way Alan Sugar, Amstrad's MD, works. According to Sugar, the company motto is something along the lines of: "Our consumer is the truck driver and his wife". *(Sounds like a certain chocolate bar — Ed.)* By this he means that people want computers that they just plug in and switch on — no messing about with different little boxes that need masses of interconnecting wire.

Sugar also stressed that he sees the computer market as a consumer electronics market, just like any other consumer electronics market, and that "Our business has always been based on the material content of the product," by which he means that some people's prices are too high for what it costs to put their computers together — rather than his prices being exceptionally low.

44% of Amstrad's business is outside the UK, and in the computer line, considerably more; in 1984, Amstrad claim (and we have no reason to doubt this claim) to have sold 200,000 computers, and in 1985 they say they'll make and sell 600,000. This implies that by the end of 1985 there will be about 300,000 Amstrads in use in the UK.

Sugar was very keen to kill speculation that there would be another computer launched during the year. Although he says he expects to see other computer products appearing he wouldn't say *what* (he says that he won't announce the products until they're about to go into the shops).

One final point regards Compact Disc — nothing to do with computers. Here Sugar notes that this is a market just waiting for a product of the 'truck driver and his wife' price — and here's a non-truck driver that would buy one!

Dave Bradshaw

BACKNUMBERS

JANUARY 1984
TRS-80 programmer's aid, Apple music, Electron review, TRS-80 screen editor, calendar program.

FEBRUARY 1984
Using MX-80 graphics, Colour Genie monitor, non-random random numbers, ZX81 Forth, Program recovery on the Commodore 64.

MARCH 1984
Easycode part 1, BBC poker, Spectrum SCOPE review, Genie utilities, Spectrum Centronics interface.

APRIL 1984
MEMOTECH MTX500 review, Genie BASIC extensions, Brainstorm review, Disassembly techniques, Recursion.

MAY 1984
Debugging, Spectravideo SV318 review, Extending the Commodore 64's BASIC part 1, Z80 text compactor.

JUNE 1984
Adler Alphatronic review, Digithurst's Microsight review, Commodore search and replace, CP/M directory, Interrupts.

JULY 1984
Commodore BASIC extensions reviewed, The Art of Islam, a fast sort, Brother HR5 review, Random Thoughts, extended palette on the Dragon.

AUGUST 1984
Apricot xi review, BBC Mode 7 screen editor, Genie sprites, Microdrive-file line editor, TRS-80 screen scroller.

SEPTEMBER 1984
CUBE's Beebflex, Electron drawing utility, MTX real time clock, Commodore SX64 review, BBC disassembler, TRS-80 Fastsave.

OCTOBER 1984
AMSTRAD CPC464 review, Dragon sprites, Commodore 64 adventures, BBC Draughts, Nascom screen dump.

NOVEMBER 1984
Apple IIc review, Epson PX8 review, MTX utilities, Z80/TRS-80 memory move routine; 16-page Business supplement.

DECEMBER 1984
Acorn Bitstick package review, Art and the AMSTRAD, BBC Draw, Psion Organiser review, Koala Pad review.

JANUARY 1985
BBC Commodities, Tatung Einstein review, Fujitsu Micro 16 review, Commodore 64 prettyprint, MTX500 Life, Nascom string-save.

FEBRUARY 1985
The Intelligent Computer, Dragon interrupts, BBC Machine-code monitor, Tasword 464 review, Spectrum/BBC cassette volume meter, Sakata SCP800 printer/plotter review, Spectrum ON ERROR, TRS-80 mail list, BBC passwords; Deficiency, Abundance, Perfection.

If you've lost, lent or had stolen one of those precious back copies of Computing Today then now is your chance to fill the gap in your collection. The list of issues given here represents the few remaining copies that we have available to help complete your library of all that's good in features, programs and reviews.

If you want one of these issues, it's going to cost you £1.40 (including postage and packing) but we think that's a small price to pay for the satisfaction you'll get. Ordering could hardly be made simpler — just fill in the form, cut it out (or send a photocopy) together with your money to:

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

If you wait until next month to do it, the chances are that we'll have run out of the very issue you wanted!

A COMPUTING Today

BACKNUMBERS

Please send me the following Backnumbers

ISSUE

At £1.40 each. I enclose £ ....

NAME ....

ADDRESS ....

....

....

.... POSTCODE ....

Signature ....

I enclose a cheque/PO for £..... (Payable to ASP Ltd)
I wish to pay by credit card

Access ☐ Barclaycard ☐

Insert Card No.

If you wish to pay by Access or Barclaycard, just fill in your card number and sign the form, do not send your card.

Please allow 21 days for delivery.

TALKING SHOP

Don Thomasson

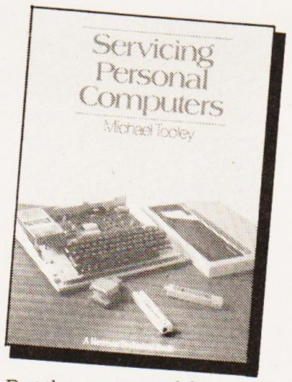
Among the books which have recently come to us for review, one of the more unusual - and therefore more interesting - was **Servicing Personal Computers,** by Michael Tooley (Newnes Technical Books). Even the first brief glance through its pages gave us food for thought.

It can be quite difficult to get a faulty computer repaired, and this creates a temptation to attempt. Some manufacturers disclaim all responsibilty for a machine if there is the slightest evidence that the screws holding the outer case together have been disturbed, so anyone who attempts to effect a cure should be reasonably confident that he can bring the work to a successful conclusion.

To do this, he may need a combination of knowledge and skills ranging from simple electrical matters to a detailed understanding of the software involved. He will need at least a circuit diagram, and preferably a technical manual, though - as Mr Tooley points out - some technical manuals contain very little useful information.

Mr Tooley's objective is to provide a broad spectrum of information likely to be useful to anyone who wants to cure an ailing computer, but it is not entirely clear whether he is writing for a private user or a professional repairer. A section on test equipment describes gear with total cost running into thousands of pounds. There is even a diagram suggesting how the workshop should be laid out, with data on fire precautions and the like.

This conjured up a picture of someone reading the book and then setting himself up to offer a repair service. A tyro might well get away with this, providing he was cautious enough to avoid making matters obviously worse, because many faults can be cured almost by accident. Removing socketed chips and putting them back again has solved many a problem. Other defects are more difficult to trace. a single 'dud' store location may affect one program but not others, leading to a suspicion that the fault lies in the program. A thorough store check is always worth while, especially if it runs automatically on a soak test basis.



But there are problems which can only be solved by patience allied with experience. I remember a program that had a habit of going dumb, leaving the screen shimmying idly, with no response to anything but Reset - which, of course, lost the evidence. After some six weeks of spare time fiddling - the fault became a challenge - I noticed that location 0072 had been set to &72, fortunately without causing the program to halt. Suspicion fell on the line listed below;

```
FOR J7=0 TO LEN(DS$) - 1:
POKE DS+J7,ASC(MID$(DS$,
J7+1,1))
```

The idea was straightforward enough. The string DS$ was to be POKED into location from DS on, these being in the screen RAM. What it was doing was POKEing a byte from DS$ into a location based on DS$, not DS. Changing DS to PP throughout removed the bug completely.

It did not, however, remove the real fault, which I must admit was never traced. It could have been a bug in the interpreter, which could have been checked by the fact that the fault appeared on another machine of the same type - and it did, I believe. In that case, I had no wish to redesign the interpreter.

Another type of problem about which we hear a good deal arises from compatibility of hardware and software bought from different sources. None of the suppliers want to know about the problem, and the unfortunate purchasers are left with junk on their hands. It might be necessary to get several people together to find a solution, and nobody has the time for that sort of thing.

□ □ □

Would Servicing Personal Computers have helped my problem or one of these compatibility bugs? When the computer lies doggo and won't give a helping hand, more than a broad spectrum of knowledge is needed. Lateral thinking and imagination come into play, and those are difficult skills to teach.

The worrying thing about the book is that it provides relatively basic information regarding such matters as circuit symbols and computer architecture, not to mention the circuits of memory cells, which aims it at people of very limited experience, yet there are sections on the repair of disc drives, printer and monitors, which call for expert attention.

However, the book contains a lot of interesting data. It may scare off those who are likely to get themselves and their equipment into trouble, and if it inspires others to study the subject more deeply it will have achieved its object. Perhaps the ideal reader might be a television service engineer who wants to move up a step.

The lack of adequate servicing facilities is a brake on the development of the computer world. So is the lack of real sales support. Looking round my populous home area, I have found remarkably few dealers, and among the 'ones I did find there was an offhand attitude. Technical Manuals? You should write to the manufacturer. 'Deliver the equipment? Haven't you got a car?' Go back for questions about apparent faults, and you get a shrug of the shoulders. Is it any wonder that small computer sales are below expectations?

Where servicing is concerned, Mr Tooley has the right idea, up to a point, but he may not have hit the target as precisely as he would have liked. In 216 closely-packed pages he has covered a great deal of ground, but if the book were four times the size there would still be vital information left out. As an appetizer, the book is fine, but it needs following up, with expansion of some of the points barely mentioned in passing.

With such a vast volume of books appearing on the market each month - we can barely scratch the surface in our reviews - it seems a great pity that there is not a set of really sound tutorials on different aspects of computers. The theory is that books linked to a particular type of machine will sell better than anything of general interest. That may be true, but one wonders if the general field has yet been properly exploited.

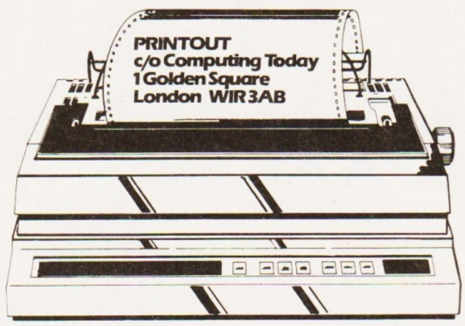
One problem is that it can sometimes be extraordinarily difficult to explain in print a procedure which you can carry out yourself without thinking. There is a danger that the explanation may become too complex, making something essentially simple look complicated. If anything, Mr Tooley makes complicated things look simple, so perhaps we should attempt a joint effort!

□ □ □

In one sense, Servicing Personal Computers aligns with our current policy to provide general information, rather than listings for particular computers. What we want to do is to provide information that people need. If we worry too much about the use they may make of the information, we will get nowhere, so we must not be too critical of Mr Tooley's offering. He clearly has a wide knowledge of his subject, and has used the pages at his disposal to good advantage.

Servicing Personal Computers
Michael Tooley
Newnes Technical
Books £17.95

PRINTOUT



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

CAN YOU HELP?

Dear Sir,
I started my company just over three-and-a-half years ago, with the aid of a 16K Video Genie and printer.

During this period, I bought a word processing program (Word 4 Word) from a company called Premier Publications in Croyden. This has proved to be an absolute boon, and is an extremely versatile word-processing package.

Here's the snag: I very recently acquired a Model I, level II TRS80 with twin SS/SD discs. Naturally, or rather unfortuantely, the cassette version overwrites part of the DOS and consequently will not work properly. I have tried to load it in higher memory but I am not very familier with machine code, and so cannot understand where the fault lies.
Any suggestions?

Yours sincerely,
David Bell.

Can anybody help Mr Bell?

CBM 1520 GROUP

Dear Sir,
I am trying to start a user group for the CBM 1520 printer/plotter with the aim of exchanging programmes, hints/tips etc. I would be greatful if you could mention this on the letters/news page of your magazine.

Anyone interested should contact me at the above address (S.A.E.) or on: Prestel Mailbox: 782279612, Compunet Courier: SCB 1

If anyone has any programs they can send them to me on a disc or cassette and after I have collected a number of programmes it will be returned to them with the new programmes recorded on it.

Regards,
Steven Birks.

AVERAGES AND TRENDS

Dear Sir,
As a professional statistician in industry, I was interested to see the article 'Averages and Trends' in Computing Today March 1985. It was useful to give some publicity to the dangers of misinterpreting statistics.

I was however concerned about the veiled criticism of statisticians, especially in the Contents page summary. Anyone can generate statistics by putting numbers through a computer program. Wrong conclusions are often reached due to a lack of understanding of the underlying theory. The role of the statistician is to ensure that data are analysed according to the appropriate mathematical model, and interpreted properly in the correct context. Unfortunately, statisticians are not always consulted.

May I point out some typographical errors in the definitions of mean? The Σ sign is missing in the arithmetic and harmonic definitions, and in the geometric the 1/N should be a superscipt to donate a power.

The diagram showing the mode, median and mean is misleading; they should be drawn much closer together, and the mode should, of course, be exactly the highest point of the frequency curve.

Finally, referring to the penultimate paragraph, ninety-five percent of a Normal distribution lies within plus or minus two, not three, standard deviations from the mean.

I hope these comments have been helpful.

Yours faithfully,
D.H. Williams

Thankyou for your comments, Mr Williams. The 'veiled criticism' you refer to was not intended to malign statisticians, but to encourage readers to beware of accepting 'interpreted' data outright.

BENCHMARKS (AGAIN!)

Dear Sirs,
I would like to comment on the Benchmark Test article in your February issue. I have no complaints about the article, but the timings seem to be erroneous. I own a Beeb and an Amstrad, and tried the benchmarks on them, using the Beeb's TIME command to time the Amstrad and vice versa. My results, (see table), disagree with yours, especially with the Beeb.

Your magazine's timings seem to show the Beeb as nearly three times as fast as the Amstrad machine, but in reality it is only slightly faster:

Yours sincerely,
Colm Buckley

P.S. Comparing your timings to those given in Personal Computer World, BM8 seems to have a serious irregularity, with the discrepancies ranging from 1000% to 0% (BBC and Amstrad), and your timings average 1/7 of the time given in Personal Computer World for this Benchmark.

		Yours	Mine
BBC	BM1	0.8 sec.	1 sec.
	BM2	3.1	3.1 secs.
	BM3	8.3 sec.	8.2 sec.
	BM4	8.7sec.	8.7 sec.
	BM5	9.1 sec.	9.1 sec.
	BM6	13.7 sec.	13.9 sec.
	BM7	5.3 sec.(!)	51 sec.
Average		8.7875 sec.	14.55 sec.
Amstrad	BM1	1.1 sec.	1.1 sec.
	BM2	3.3 sec.	3.3 sec.
	BM3	9.2 sec.	9.2 sec.
	BM4	9.6 sec.	9.6 sec.
	BM5	10.2 sec.	10.2 sec.
	BM6	19.2 sec.	19 sec.
	BM7	30.3 sec.	30.2 sec.
	BM8	34.2 sec.	34.2 sec.
Average		14.6375 sec.	14.59 sec.

Apart from Benchmark 8 (see April issue for explanation!) we submit that the discrepancies arise through the use of different timing methods. Our Benchmarks are timed manually (stopwatch).

The point is that Benchmark timings are very general estimates of a machine's performance, therefore timing differences of 1/200th of a second can, for all intents and purposes, be ignored. Interestingly enough, your average timing for the BBC micro is only significantly different from ours because of the enormous difference between BM8 -stressing the dangers of accepting averages without question!

Oh, and if our average benchmarks for the BBC and Amstrad are 8.7875 and 14. 6375 respectively, again, it is stretching the point a little to suggest that we make the Beeb three times faster than the Amstrad! Almost twice as fast is nearer the truth - even though it isn't, really (?).

COMPATIBILITY

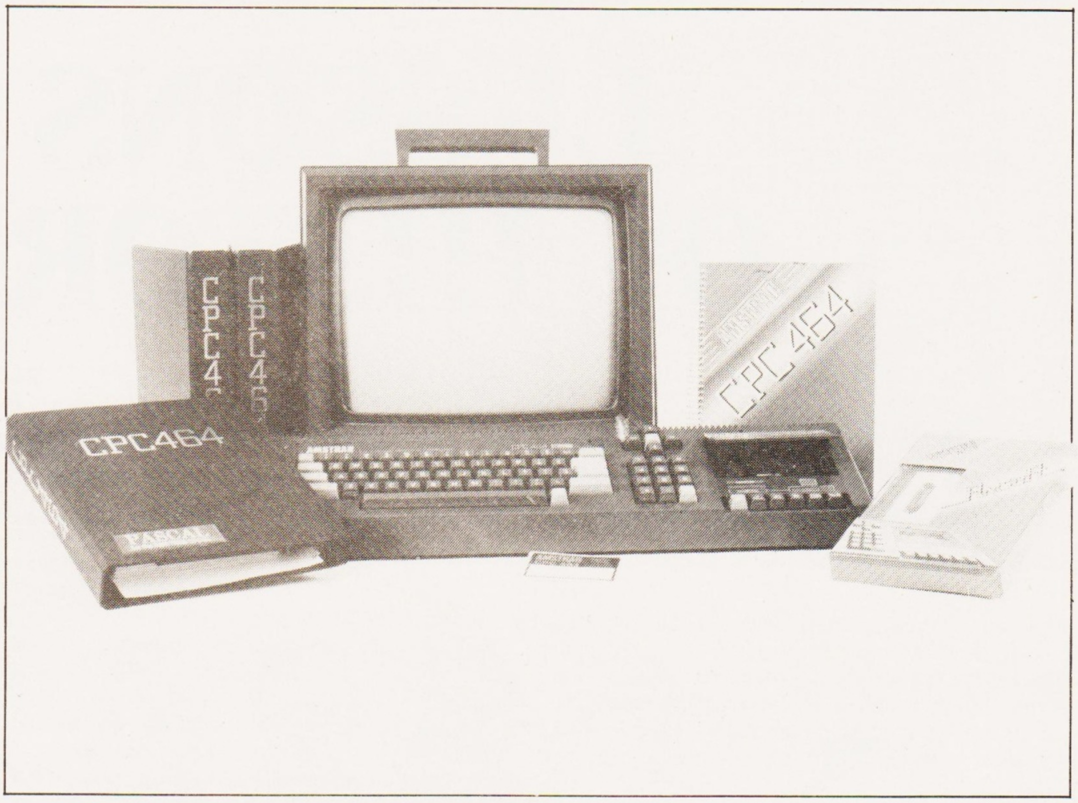
Dear Sir,
Computing today first came to my notice two or three weeks ago when I purchased the February issue. It would seem that Computing Today has gone through a bit of a traumatic experience, and that you have joined them as the Editor. I like the style and content, and although I have a quite expensive set-up based on the BBC, and subscribe to such mags as Micro User and Acorn User, I find your magazine quite refreshing to read, light and understandable.

I have a BBC B with a twin Mitsubishi 800k double sided drives, switchable 40/80 tracks, a second processor, a Mitrovitec monitor, a KAGA printer and a great deal of software. I use Hiview a great deal, typing up the work my wife prepares for her University degree.

I recently needed to sell my original printer for one that would support NLQ, and with the ability to microspace, and as such I contacted Watford Electronics, who recommended that I purchase the Kaga 810 plus the Acorn driver. I did explain that I was using Hiview, and I was told that the Kaga was fully compatible with the Epson FX80 and should therefore have the best of both worlds: compatibility and NLQ.

Up to a point that is correct, but now I have the problem that I do not seem to be able to Microspace nor prevent excess line feeds at the start and end of my pages. I telephoned Watford, (hve you ever tried? It took over half an hour of repeated dialling before I got through) to seek their help. It transpired that although I bought the printer and driver from them, they could not help beyond saying that their FX80 driver would be a better bet. I bought one but it didn't work, another telephone call but this time I was told that it wouldn't support Hiview and that no plans were afoot for another version which would.

I have also spoken to Acorn who state that their driver will support microspacing if I enter the correct codes when I form my own drive. I can't do that either as I cannot find a code that the driver program will accept, even though I have used several variants of the codes given in the manual. I even bought the book Watford produced called 'FX80 Commands Revealed' but all to no avail. So I turn to you, and as the several articles in the issues of your magazine that I have read indicate the knowledge is available can you help please, and if you cannot to whom *CAN* I turn?



Yours sincerely,
Charles Harvey.

P.S. You might be interested to know that Spellcheck II will not work with the Watford DFS 1.40. Mine is up for sale very cheap, as I have reverted back to the Acorn version.

Thankyou for your letter Charles.
Answering your questions in turn: yes, magazines should cater for the readers who support it, although the extent of this support varies considerably, and this is not just because of company 'policy'. Computing Today, cannot promise to answer letters personally. This is not through lack of desire, but by publishing the letter in PRINTOUT, we can kill two birds with one stone and give other readers the benefit of advice meted out in the course of our reply.

But to answer your main question: We can only use your experience as an example of what other readers should avoid. Before purchasing anything that exhibits dubious compatibility with existing equipment, ensure that the product IS compatible with your system. If possible, try to see the product demonstrated on an identical set-up before buying.

DRAGON ORANGE TEXT

Dear Sir,
I was really moved by the problem of Ms M. S. Grees (letter in CT 2/85)...to my good, old, tamed DRAGON. Here is the result: The Orange Text Screen.

Motorola's video generator chip used in the DRAGON allows the selection between two alphanumeric displays. This selection is made by setting bit 3 of the data input of 6847 to zero or one. As this bit can be set through the PIA 1 of the machine, a software selection is possible.

Because of reasons only known to the creator of the DRAGON, this bit is set to 'zero' by some BASIC commands (like PRINT, INPUT...) and therefore the only screen in text mode seems to be black on green or (thanks to Mr. Seymour's excellent program) green on dark green. It is therefore necessary to set this control bit to 'ONE' as often as possible. The following, short machine code routine does this at every software interrupt - every 1/50sec. The routine puts a jump to a short reset program into the place looked-up by the interrupt routine, which is at $010D.

```
ORANG LEAX XCHG,PCR
STR   STX $010D
      RTS
RESET LDX $9D3D
      BRA STR
XCHG  LDA $FF22
      ORA $08
      STA $FF22
      JMP $9D3D
```

For the immediate use this comes to the following dataline to be inserted as the last DATA into the inversion program of Mr. Seymour ('Negative Thoughts'):

DATA 308D0009, BF010D, 39, 8E9D3D,20F7, B6FF22, 8A08, B7FF22, 7E9D3D

The checksum control value must be adapted - no problem.

EXEC ST - 24 switches the orange screen on

EXEC ST - 16 will reset to black (green on black)

The advantage of the orange colourset is a much brigher display even on a B/W TV set. Some BASIC routines cause a flicker during execution but beside of this the screen is stable.

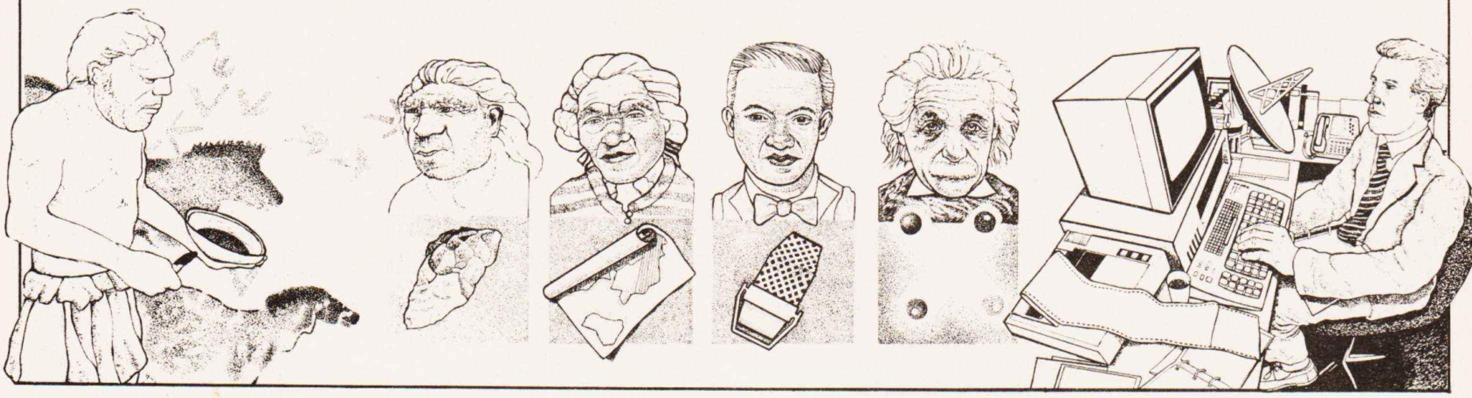
I hope this makes the sunny days of Ms Greer even brigter and her screen more sunlike.

Your sincerely,
Erich Moser
Vienna, Austria

LESSONS OF HISTORY

Bill Horne

By the end of 1950, the three main pioneer machines in Britain were in operation. They had proved that computers were viable in principle, and it was now necessary to decide how they could best be put to use in practice...



Ferranti, having built the fully-engineered Manchester Automatic Digital Machine (MADM), produced a copy for Toronto University, but in other quarters the idea of manufacturing machines for sale seems to have been given relatively low priority. English Electric, having been concerned with the N.P.L. Pilot ACE, set about the design of an extended version, the DEUCE, seeing it as a useful tool in the creation of other engineering products. J. Lyons set up a team to build a version of the EDSAC, calling it the Lyons Electronic Office, or LEO. Neither of these latter companies visualised themselves as making computers for other concerns.

This proved to be important in the long run. Before the computer could be produced in quantity, some key problems had to be solved. Each machine sold would require engineering support, and perhaps software support as well, though the word 'software' was not yet in common use. Even more immediate was the problem of economic construction, and the improvement of reliability.

These difficulties were not solved overnight. First to last, the building of the first three machines had occupied the best part of five years. Their successors took almost as long to mature.

Elliott Brothers, who had not been as directly involved in the first machines as the other companies named above, were ahead of this schedule. With the backing of the National Research Development Corporation, they produced the 401, demonstrated at the Physical Society Exhibition in January 1953, less than nine months after its conception. This was described as a 'large number-cruncher'.

The speed of development of this machine was in part traceable to the use of standard circuit modules, which were also used by Ferranti. These had served for the 153, a computer for the Admiralty, and for Nicholas, which had an interesting origin.

The contract for which Nicholas was built was not originally for a computer at all. It was for a series of computations. The team concerned decided that a computer would make the work easier and more interesting, so they adopted the modules designed for the 153, assembling them in their leisure time at home. The store, in the form of nickel delay lines, had a capacity of only 1024 32bit words, but the machine proved able to fulfil the requirements of the original contract, and was still in use some years later.

Dr J M M Pinkerton was educated at Trinity College, Cambridge, and joined J Lyons and Co. Ltd. in 1949. Here Pinkerton took charge of the development of the LEO series of computers, moving on to become Director of Leo Computers Limited in 1959.

This introduced an important new concept. Instead of selling a machine, why not use it to carry out calculations for other people? In the long run, that might prove more profitable. The idea of the computer bureau had taken shape.

Nicholas, to be frank, was a mess, by present-day standards. It occupied several six-foot racks, at least some were that high, others being slightly stunted. The face of the racks was a mass of tangled wiring, and odd wire trailed away to an ocsilloscope used for diagnostic purposes. That was not important. As has been remarked of some of its predecessors - It worked.

FORMAL STANDARDS

The 401 was built to much more formal standards. There were seven equipment bays, each equivalent to a freestanding rack in capacity, and the interconnecting wires were respectably hidden at the back. There were over 150 plug-in modules, contained in three of the racks, and the remaining bays were occupied by power supplies and control circuitry. The modules, on average, contained about two valves each, so the power supplies had to be fairly capable.

The main store of the 401 was a magnetic disc, which initially had seven tracks and a capacity of 1024 words. Shift registers were based on nickel delay lines, which worked like mercury delay lines but on a shorter circulation time.

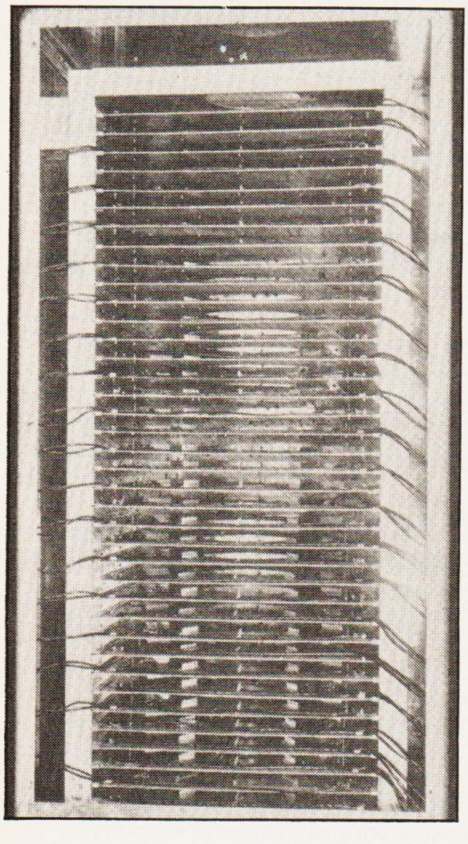
On exhibition, the 401 distinguished itself by working through a whole week, though it became temperamental when it was taken to the Mathematical Laboratory at Cambridge. It eventually went to the Agricultural Experimental Station at Rothamsted, and is now in the Science Museum, London.

The 401 was succeeded by the 402, of which nine were made, and most of them were still in use in 1969, fourteen years after they had first gone into service.

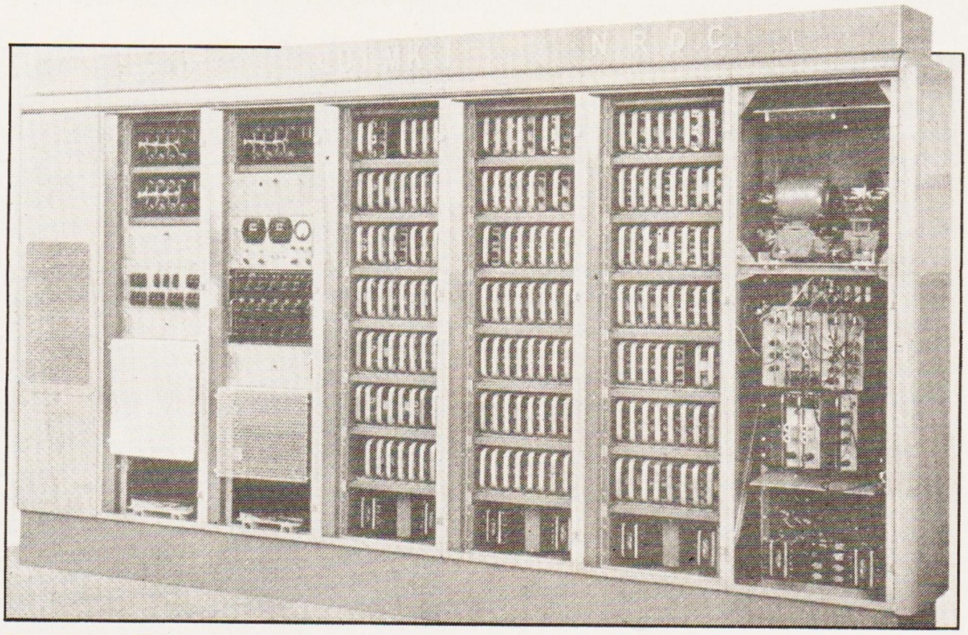
The 403 and the 405 followed, no less than thirty of the latter being sold, but these verge on the end of the era under consideration.

LEO were perhaps next in chronological order, though Ferranti were still making versions of MADM. Because it was primarily intended for payroll work and similar activities, the input/output arrangements for the original LEO I were of particular importance. When he joined the company in early 1949, Dr J.M.M. Pinkerton found that the requirements was none the easier for that.

The target was to handle a payroll for 18,000 staff at a rate of twenty persons per minute. It seemed that teleprinters, which then worked at seven characters per second, offered the only reasonable means of output, though Hollerith tabulators and many other devices were considered. It was estimated



The Nicholas 1024 x 32-bit nickel delay line memory.



The Elliott 401.

that three of four input tapes would be needed to match the required data input, but the bottleneck would arise with data output.

With so large a proportion of external data flow, it seemed that the actual processing time would be relatively small, perhaps a third of the input/output time. This was unusual, and led to the development, in conjunction with STC, of a rather specialised system. Data generated on teleprinters was recorded on magnetic tape, which was read into three concurrent input channels, each embodying a converter which accepted inputs in pounds, shillings and pence, and generated the binary equivalent. The result was passed into duplicated buffers, allowing the computer to read data from one buffer while its companion was being filled.

The output system reversed the process. The data was passed to holding buffers, which fed the system for re-converting from binary to sterling, this process including the suppression of leading zeroes and insertion of spaces. The converter output was recorded on magnetic tape to serve as a basis for the next run, and to drive a teleprinter output system.

The overall picture which emerges is of a highly specialised computer, quite different from the general purpose computers which were being made elsewhere. This, however, was entirely a matter of the input/output system. The main computer, which used about 7000 valves, was much the same as its contemporaries, not least in that it was somewhat unreliable. It was nominally ready to run in 1952, but nearly two years were to pass before it was fit to enter service.

The main problems were not with the computer itself, though that suffered some failures. The main sources of error were the input and output converters, which used gas discharge tubes rather than thermionic valves. The whole input/output system had to be reconceived on the basis of using card readers and punches and printing tabulators.

Work on the payroll began in earnest in January 1954, and went on with good reliability. In the summer of the same year, the Ford Motor Company payroll was taken over. Development continued, however, and it was found possible to put the sterling/binary conversions where they belonged, under the control of the main program. The throughput rate was raised beyond the originally stipulated 20 per minute to 50 per minute.

Ultimately, LEO I met the vital criterion - it worked - but it might have worked sooner if the original input/output system design had been less adventurous. There is something to be said for accepting the devil you know.

URGENT PATCH

In 1955, the DEUCE was not quite finished, the magnetic drum being the main outstanding item, but it was brought into service to support the radio coverage of the General Election. Two machines were used, the operators communicating by telephone to compare and check the results they were getting. That failed to detect a rather fundamental error, which predicted with increasing emphasis that the party which was obviously losing was going to win. An urgent patch to the program was made around midnight to correct the sign of the algorithm...

Basil de Ferranti joined the family business, Ferranti Limited, in 1954. Appointed a Director in 1957, he became a Director of International Computers and Tabulators Ltd. (now ICL) in 1963.

English Electric had already been approached by several companies and establishments that wanted to acquire computing equipment, and what had begun as an inhouse affair ended with a production run of fifty. This has a certain relevance to the very existence of this article.

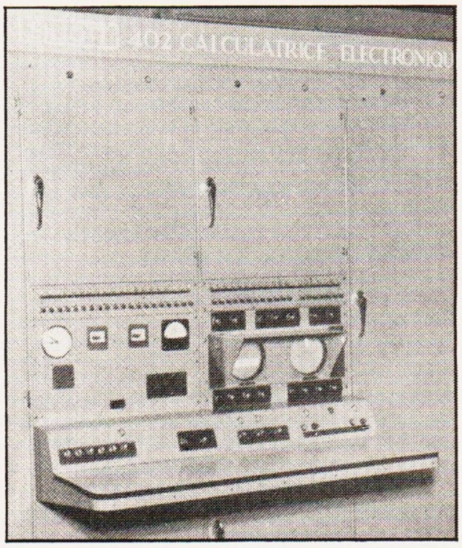
In the hot summer of 1959, an engineer specialising in electronic servo systems was called into his department manager's office, and asked what he knew about computers. He replied that he knew nothing at all. The manager grunted. "Well, go down and see why the one they've been installing for us doesn't work yet."

Now, the servo engineer was forty, and it was said that only the young could cope with computers. He therefore went to investigate with little hope of success, but was pleasantly surprised.

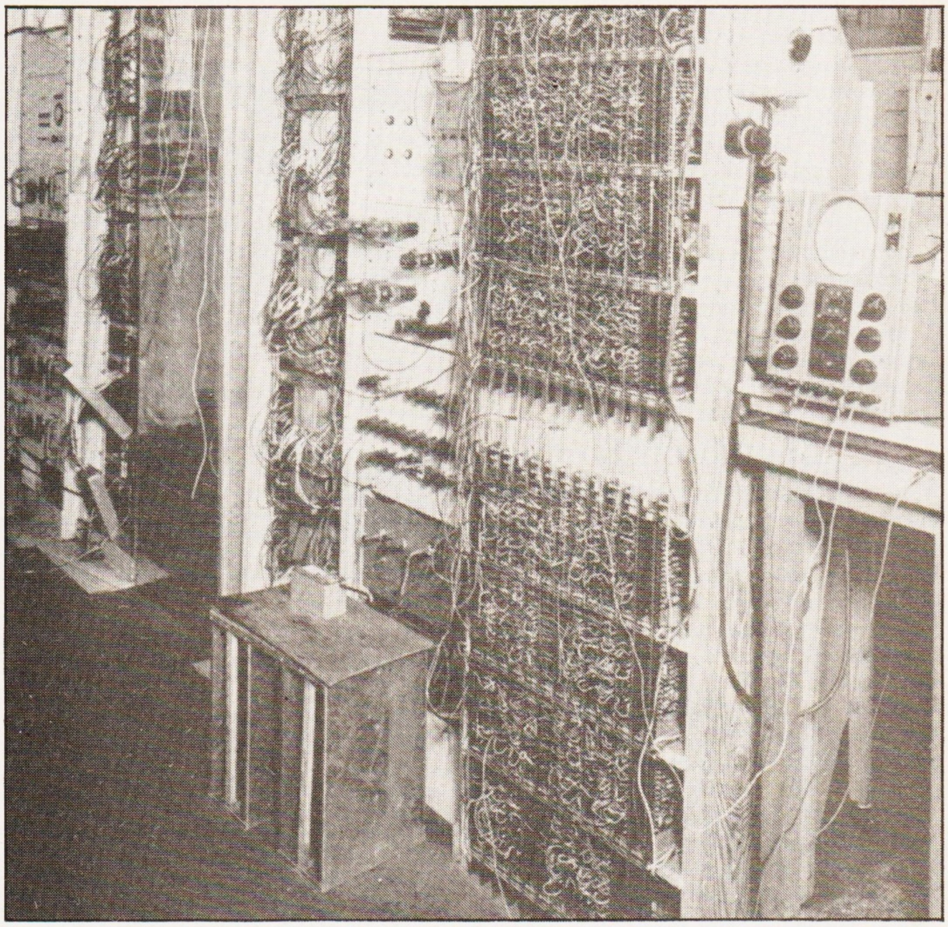
The thermal cut-outs protecting the main power supplies were dropping out prematurely because the computer room was too hot. An extension to the cooling system soon put that right. Then there were the mercury delay lines. They went out of adjustment every afternoon, and that was because they were adjusted in the morning, when the sun was shining on them directly. In the afternoon, they were shadowed, and could cool off - and fail. The most satisfying diagnosis, however, concerned the magnetic drum. The movement of the read/write head was controlled by a servo which used a small wire-wound resistor to sense position. this resistor kept falling apart, because the servo was unstable, and the contact arm dithered instead of coming to rest.

There were a number of other little bugs, but by that time the conversion from servo engineer to computer engineer was well under way. It all seemed so obvious. Not difficult at all.

It should be made clear that the problems were not design problems, but more a question of the way the equipment had been installed and set up. As mentioned earlier, the sale of computers had generated a demand for engineering support that was difficult to supply. The field engineers did their best, but there were too many things to think about.



The Elliott 402.


The Nicholas Computer, oscilloscope, and fire extinguisher.

That was in 1959. Less than ten years later, the servo engineer was working in a senior position in the company that had supplied the computer. And now he is writing about how it all happened...

FERRANTI

The activities of Ferranti remain to be considered. They looked at the possibilities of using a computer for wage calculations, but decided that it would only be viable with a very large payroll. This was an instance of the 'big/fast' concept, which will be examined at a later stage. Instead of taking a detour, like LEO, they went directly to the use of punched cards for input and output, and thereby saved themselves a great deal of trouble, but they failed to break into the commercial market, which was their original aim, and had to consider alternatives.

They concluded that a multi-pronged attack was needed, covering a large scientific machine, a large commercial machine, and a smaller general purpose machine.

The first to appear was the small machine, called Pegasus, and it was followed by Perseus, a large machine specialised to insurance work. These used the standard packaging system mentioned in connection with the Elliott 400 series computers. Then came Mercury, a very big machine conceived at Manchester University.

Pegasus was a success, but Perseus was not. Both were valve machines, and the days of such machines were numbered. Even by 1955 that was evident. Many new techniques were becoming available, not least among them the Core Store, which gave instant access to a large block of data, and could be used to hold constants or variables. It would even retain data when power was switched off, providing suitable precautions were taken. In other words, it did all that earlier forms of store did, and a lot of things they were unable to do. it was to hold pride of place for more than twenty years, yielding in the end because it was too expensive to make, rather than because anything equivalent or better in performance had been discovered. In terms of capability, it still remains supreme, though perhaps a trifle slow.

This, however, was not the only innovation of significance. Under the pressure of necessity, devices like teleprinters were being improved all the time, and fast readers and punches for card and tape were being enhanced. Magnetic tape systems also showed improved performance, and so did many other types of peripheral device.

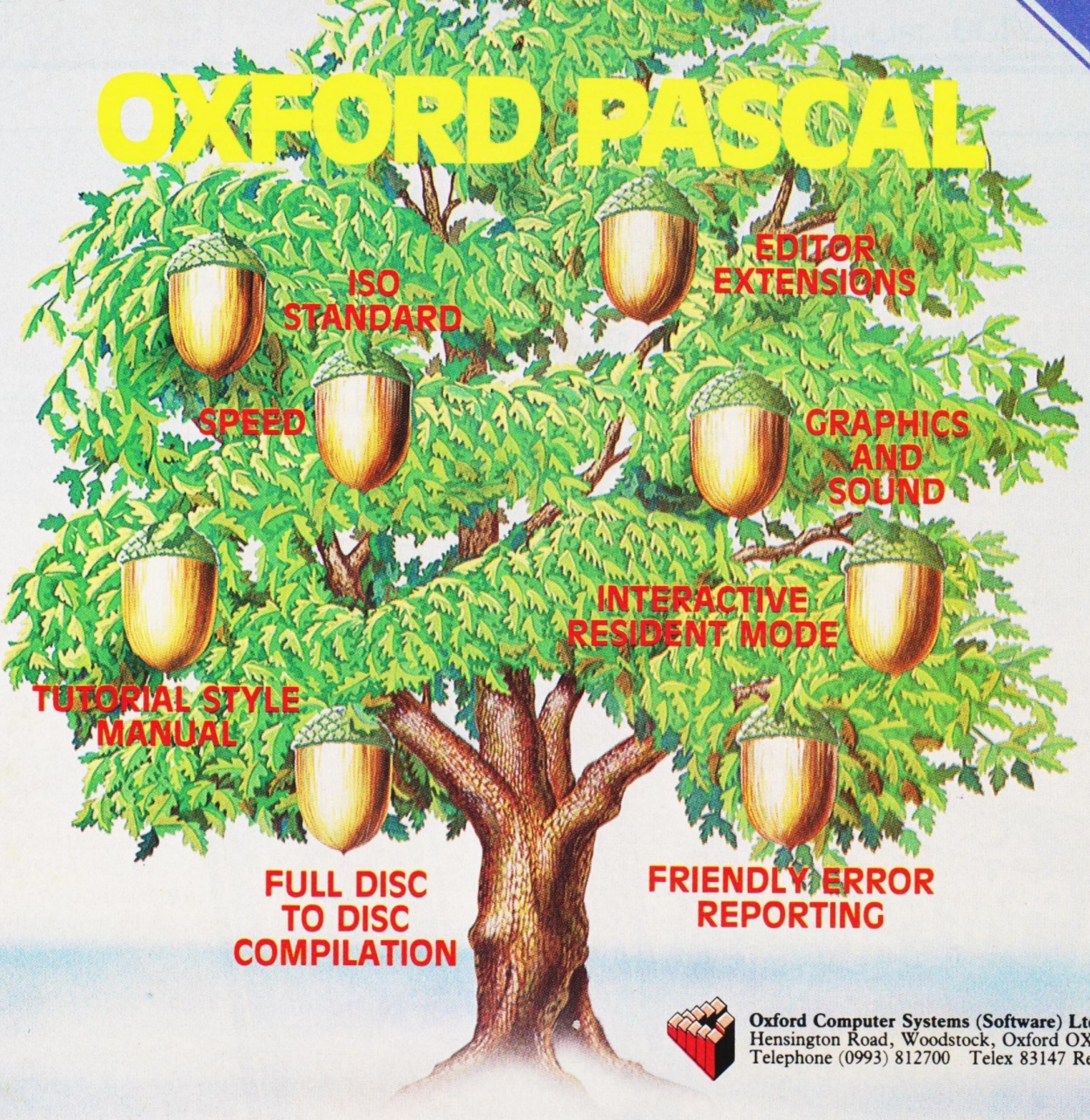
In the midst of all the peripheral activity, the computer makers were facing inevitable changes, to which they responded with some reluctance, but response was essential. This, perhaps, diverted their minds from equally important problems, such as the right dimensions for an economic computing device. Before examing that, however, we must look at the impact of the most important innovation of all, the Transistor.

The Lessons of History is continued in the next edition of Computing Today

Pictures Courtesy *The Radio and Electronic Engineer.*

BBC and Electron

OXFORD PASCAL



Oxford Computer Systems (Software) Ltd.
Hensington Road, Woodstock, Oxford OX7 1JR, England
Telephone (0993) 812700 Telex 83147 Ref. OCSL

Compilers like these don't grow on trees

Oxford Pascal is Fast
Oxford Pascal compiles down to FAST COMPACT P-code, giving you the real speed and power of Pascal, together with the ability to compile very large programs.

Oxford Pascal is Standard
Oxford Pascal is a full extended implementation of Standard ISO Pascal. This means that you can compile any Pascal program (subject to size), written on any computer, anywhere.

Oxford Pascal is Compact
Because it compiles into P-code, Oxford Pascal reduces programs into the most compact form possible. In fact it allows you to pack more code into your BEEB than any other language, and should your programs become too large, you can still use the CHAIN command to overlay limitless additional programs without losing data.

Graphics & Sound Extensions
In addition to the entire Pascal language, Oxford Pascal features a whole range of Graphics (all modes) and sound extensions designed to make maximum use of the BBC Computer. Oxford Pascal also provides numerous extensions such as hexadecimal arithmetic and bit manipulation instructions.

Oxford Pascal in Education
In Education, Oxford Pascal is fast becoming a *de facto* standard. It is already the most popular Pascal on the Commodore 64, and will soon be released for the Spectrum and the Amstrad. In fact, Oxford Pascal will soon be available for 90% of the computers installed in the U.K., and is already available in German, French, Swedish, and American versions. **Students and teachers alike find that it makes sense to use a standard implementation of Pascal across the whole range of educational micros.** Call us for details of our generous educational discounts.

Manual
Both these compilers come with a manual which has been carefully designed, not only as a quick reference guide, but also as a full tutorial for those new to Pascal.

Resident and Disc Compiler
Oxford Pascal comes in two forms:

For Tape Users...Oxford Resident Pascal. A compiler located largely in ROM which is available at any time. Programs can be written and compiled on the spot without disc or tape access, and compilation is fast enough to make using the compiler much like using the BASIC interpreter. Thus, learning Pascal is a simple interactive process. Some 15K of memory is available for user programs, the remainder being reserved for compiled object code.

For Disc Users...Oxford Disc Pascal offers all the above PLUS...a full disc compiler which is capable of using the WHOLE memory for Pascal object code, it is supplied with a powerful LINKER, allowing you to break large programming tasks down into separately compilable, easily-manageable files.

Friendly Error Messages
Many compilers produce little more than an error and line number to help correct mistakes in Pascal programs. Oxford Pascal however, gives you one of 49 friendly and informative error messages. Messages which not only indicate the reason for an error, but also print out the line in question with a pointer to the exact position where the error was detected.
Run-time errors are reported using line-numbers from the original source-program, with a full explanation of how the error occurred.

Powerful Editor
With Oxford Pascal there is no need for you to learn how to use a new Editor. Pascal programs can be entered in exactly the same way as BASIC programs, without the need to learn any new commands. When you are used to using Pascal, you will find our extensions to the Standard Editor even more useful. What is more, Oxford Pascal allows you to mix BASIC and Pascal together, in much the same way that you can mix BASIC and assembler. In fact you can, if required, mix all three together...BASIC, Pascal and assembler...in one program.

Stand Alone Code
Unlike other compilers, Oxford Disc Pascal allows you to compile on the BBC and then relocate your program so that it will run on the BBC **and** on the Electron. The relocated program will run without a Pascal **ROM** and can be loaded and run from tape or disc just like any other program.

This means that you can distribute or sell your software freely and without the need for **ROM**s, to run either of the above machines.

Price/availability matrix

	BBC 'B'	ELECTRON	C64	SPECTRUM
DISC	£49.95	Not yet!	£49.95	Available April 1st 1985
CASSETTE	£39.95	£59.95 inc. Cartridge	£22.95	

All prices are inclusive of VAT
Please add £2.00 for postage and packing.

Oxford Compilers — The Future
During the next year, we at Oxford will be releasing a series of language implementations such as C, and Modular 2, for the BBC, and other popular micros.

These compilers are being built, using the most modern techniques in **automated** compiler construction, and will bring to the micro-user, a level of robustness and efficiency, only now becoming available to mini and mainframe users.

Oxford...
the **Compiler Compilers.**

Oxford Pascal Order form. Please make cheques payable to OCSS Ltd.
Please rush me my copy of Oxford Pascal. I enclose £ ....
including £2 postage and packing
My computer is BBC 'B'☐ ELECTRON☐ C64☐ (Please tick as appropriate)
I would like my compiler supplied on DISC☐ CASSETTE☐
Name ....
Address ....
....
Postcode ....
Telephone ....

LANGUAGES

Garry Marshall

...and quite a few of them; Lisp, BCPL and a 68008 assembler. All for the QL, and all from Metacomco.

With the QL's own language boasting the name SuperBASIC, why would it need any other language? The answer, as ever, is that it depends on the sort of application that you want to program it for.

SuperBASIC is, indeed, superior to most dialects of BASIC, having taken on board most of what Pascal has to offer for writing well-structured programs. It possesses IF - THEN - ELSE, REPEAT - END REPEAT, and EXIT statements, and procedures. These features give the language considerable flexibility. This is illustrated by its REPEAT - END REPEAT statement pair, which can be used with EXIT to create the equivalents of both a REPEAT - UNTIL loop and a DO - WHILE loop, as well as any other kind of loop. The language also has a coherence that comes in part from an overall use of bracketing statement pairs such as IF - END IF, REPEAT - END REPEAT and DEFINE - END DEFINE.

SuperBASIC has certainly incorporated the features of Pascal with sufficient flexibility and coherence to overcome the generally voiced objection that BASIC does not encourage the writing of well-structured programs. It might even be argued that it has done this sufficiently well to remove the need for Pascal itself.

I have recently seen the view expressed, although lightheartedly, that eventually BASIC will be extended to such a degree that it will gobble up almost all other languages as a consequence. This view was expressed by Richard Forsyth, and he was talking about Prolog at the time!

Well, SuperBASIC has not been extended that far yet, and there are still certain activities to which even SuperBASIC does not readily lend itself. One is the sort of general symbol processing that is required in all artifical intelligence programming and, to give one concrete example, in writing expert systems.

Because, generally speaking, they deal with knowledge rather than data, these kinds of application require the manipulation of symbols that can represent anything: to be able to deal with numeric variables and string variables is not sufficient. In addition to this, the symbols may need to have various properties associated with them and not just numeric or character values. This is where Lisp comes in, because it lends itself to these kinds of things, and others. it is not the only language that is used for such purposes, although it is probably the most widely used and, in any case, many of the others are derived from it.

FACTSHEET:	Lisp, BCPL, 68008 assembler
Price:	£59.95 each
Available from	Metacomco 26 Portland Square Bristol BS2 8RZ. Tel. (0272) 428781

Another activity for which SuperBASIC is not entirely suitable is what is called systems programming. This means, more or less, the writing of compilers, interpreters, editors, assemblers and all the software tools that help to make computers easier to use. BCPL is one language that is used for systems programming, although it is probably obvious that if it can be used successfully to write systems programs it can be used to write any other sort of program.

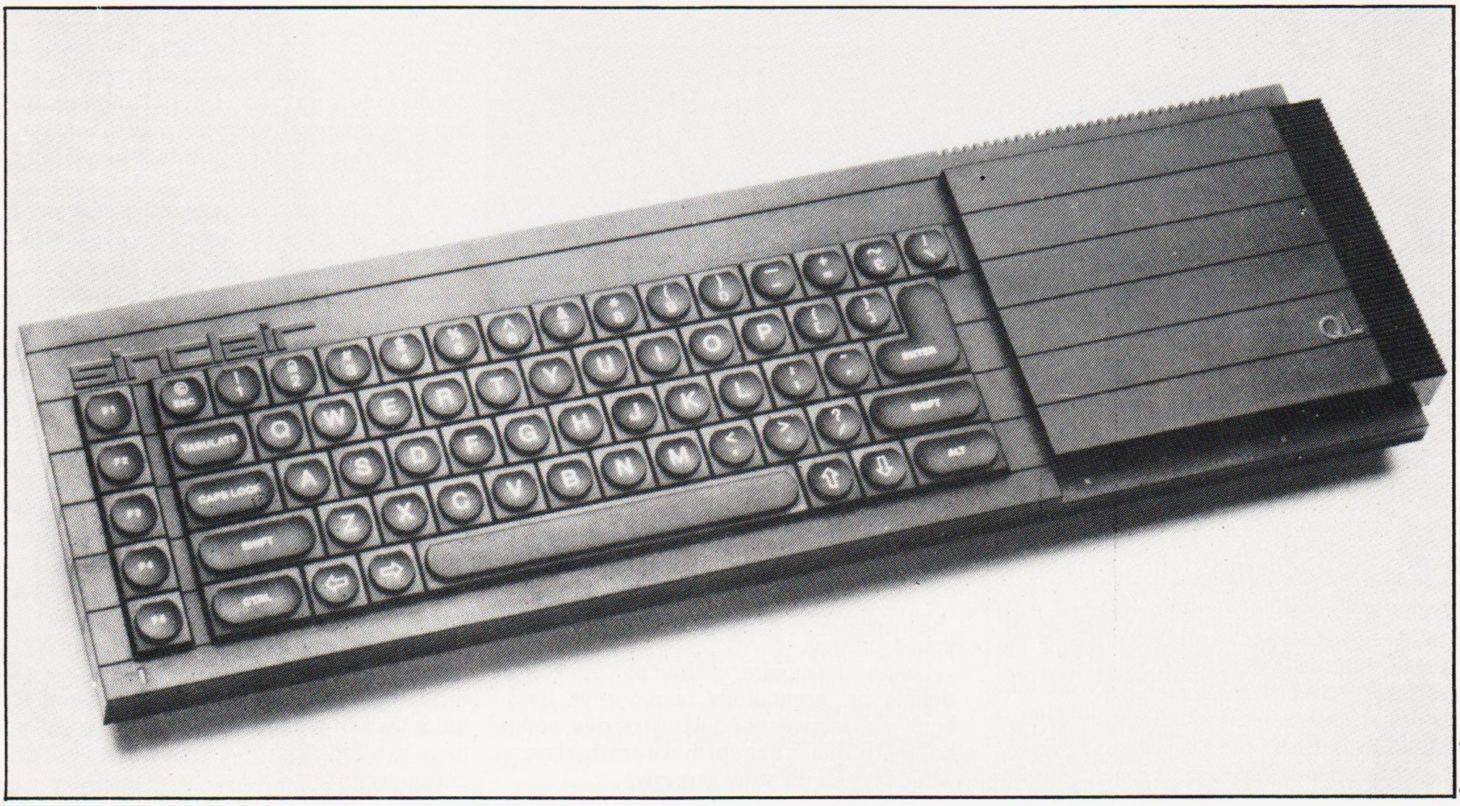
Versions of Lisp and BCPL are now available for the QL from **Metacomco,** as is an assembler that allows you to address yourself directly to the QL's (arguably) 32bit 68008 microprocessor. These languages all come as development kits complete with an editor which, rather neatly and very conveniently, is the same in all three cases.

LISP AND WHAT IT'S LIKE

Lisp is classified as a functional language. All this means is that it works by applying functions to arguments. A Lisp program is a list and is written as a sequence of items enclosed in brackets. The programs (PLUS 2 2), for example, is a list of three items. When any program is executed, Lisp expects its first item to be a function and the remainder to be arguments for the function. It then evaluates the arguments for the function. It then evaluates the arguments, applies the function to them, and delivers the result. The program just given evaluates each '2' to 2, applies PLUS to them to give 4 as the result.

This way of proceeding is not really as strange to the BASIC programmer as it may at first appear. After the assignment LET A=5, the command PRINT A works by evaluating its argument, A, and applying PRINT to give the result. In Lisp, the instruction for printing would be written as (PRINT A). We can take this a little further to show that Lisp is rather more consistent in its notation and approach than BASIC. The BASIC command PRINT A+B would be written in Lisp as (PRINT (PLUS A B)). The Lisp version is dealt with by evaluating A and B and applying PLUS to them to give a result to which PRINT is applied. Much the same thing happens in BASIC, but he uniformity is far from apparent from BASICts notation.

In Lisp, then, everything is done with functions. There is a function for assignment, and Lisp's equivalent of LET A=5 is (SETQ A 5).



An assignment can also involve a list, and a list of colours can be assigned to C by (SETQ C '(Red Orange Yellow Green)). Notice the quotes sign in front of the list: it is there to *prevent* Lisp from evaluating the list, which is the second argument for SETQ. The list is just a list of items; it is not a program and its first item is not a function. If Lisp is allowed to try to evaluate it, an error will result.

Lisp provides a certain range of functions, but if you need others you can always provide them yourself. The structure of a Lisp program comes from the way that functions are combined to give other functions. The functions that are provided include some for dealing with lists. A list can always be split up with CAR, which gives the first item in a list, and CDR, which gives a list with its first item removed. When used with our list of colours, (CAR C) gives Red, and (CDR C) gives the list (Orange Yellow Green).

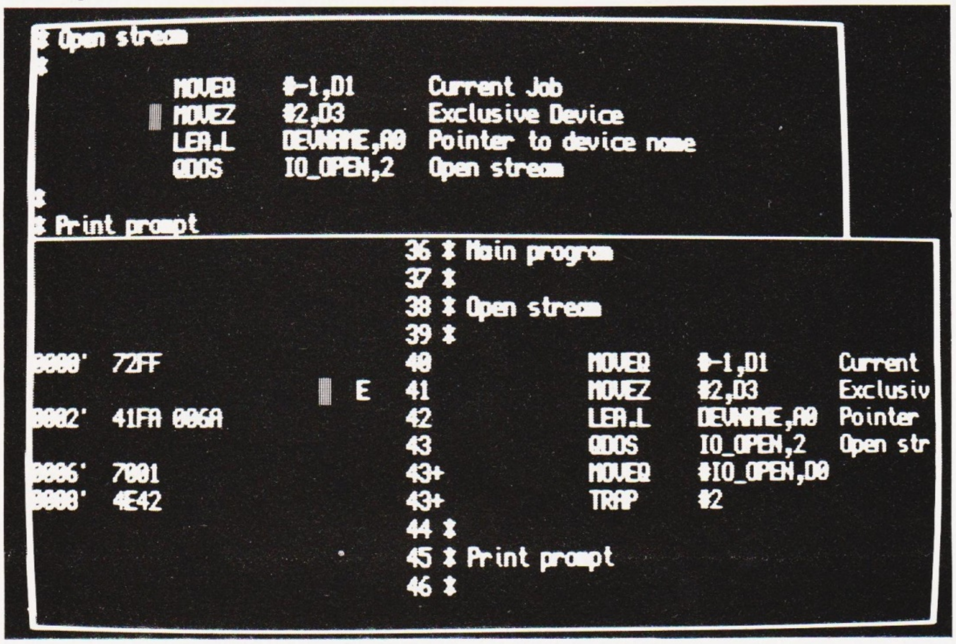
New functions can be defined, and naturally, this is done with a function. The function is named DEFUN, and if we wanted to define a function to count the number of items in a list, we could do it like this:

```
(DEFUN COUNT (L)
(COND
((NULL L) 0)
( T (PLUS 1 (COUNT (CDR L)))) ))
```

This involves the function COND which is the conditional function broadly equivalent to BASIC's IF-THEN. Its argument is a list of pairs. Each pair consists of a condition and a value. When COND is evaluated, the conditions are tested in turn, and as soon as one is found that is true, the value that is paired with it is given as the value of the function.

The definition is in the spirit of Lisp in that it is recursive. The definition of the function can be written in English as: "The number of elements in a list is zero if the list is empty, otherwise the length of the list = one plus the length of the list (except that this list is the previous one without its first element)."

Having shown that we can write Lisp programs to handle lists, we have also shown that Lisp programs can be written to handle programs, for a Lisp program is a list. This capability is needed in systems programming where a compiler, for example, is a program that deals with another program by translating it, but we will leave systems programming to BCPL.



An expert system is also a program that handles other programs, for it must deal with rules that are expressed in an IF - THEN form. A typical rule, from an expert system for computer fault diagnosis, is:

IF the storage device is a 3380

AND a state save or dynamic trace has been performed

THEN the packchange byte = bits 8 to 15 of location INBR of the trace.

The prospect of writing a BASIC program to examine and manipulate rules written in this form is forbidding. In Lisp, since the rules will be written as lists, they can be dealt with in the same way as anything else.

GETTING STARTED WITH LISP

To load Metacomco's Lisp from its Microdrive cartridge all that is necessary is to place the cartridge in Microdrive 1 (the lefthand one) and type the command

EXEC_W mdv1_lisp

The rather clumsy file-handling command is the QL's, and nothing to do with Metacomco. When loaded in this way, Lisp is placed in RAM, and SuperBASIC waits until the Lisp session is completely ready to take over again. Lisp provides a window that sits on top of SuperBASIC's but is slightly wider and a little less high. You can change its size if you want to, but it is perfectly acceptable and pressing ENTER is all that is needed to accept it. You are then confronted with "Evaluate:" and Lisp is ready to deal with your programs.

I entered a number of short programs and definitions, particularly of recursive functions, and Lisp responded just as one would hope at all times except that it crashed once when I typed an entry wrongly.

You finish a session by entering (STOP) for Lisp to evaluate. It does so, promptly passing you back to SuperBASIC and all trace of the session is lost. If you want to keep a record of a session, perhaps with some definitions that you will want to use again, you can save the entire workspace created during a session on cartridge with a command such as:

(SAVE 'mdv1_file)

This saves the workspace on the cartridge in the lefthand Microdrive in a file name 'file'. it can be loaded again with a corresponding LOAD, although when it is, it will completely replace everything in the workspace at the time.

This shows that a certain amount of care is needed to retain what has been accomplished in a Lisp session, and that the editor will be valuable, even essential, once you get well into a project. The editor isn't particularly friendly to my way of thinking, but it is adequate and I dare say that you could come to like it given enough time and use. If you use more than one of the Metacomco development systems, it may well be worth trying to like it as it works with all of them.

But it does provide a way of preparing programs away from Lisp so that you can save just what you want and not an entire workspace. When a file has been created from the editor, it can be loaded and run from Lisp by, for example:

(rdf 'mdv1_edfile)

To begin with, though, it is much easier to prepare material from within Lisp itself. The reason for this can be explained with reference to the definition of COUNT given earlier. Even the most rapid glance at it will show that the number of brackets is slightly alarming. When you type anything into Lisp, it keeps track of the bracket situation and if you press ENTER without having balanced up the brackets properly, Lisp will give you a suitable prompt and assume that you have not finished your entry. This not only lets the bracket counting be passed over to Lisp but also allows you to lay out entries in any way that you find readable.

Besides the usual functions this version of Lisp has functions to mirror the QL's SuperBASIC commands, including cls, repeat, draw and the turtle graphics commands. This makes the Lisp decidedly QLish (to Lisp purists it would probably make it just L-ish). However, it does give a direct line to the QL's existing capabilities and provides the Lisp user with a direct means of tapping them.

The cartridge with Lisp on it also contains a number of example programs and some of them are very impressive. One good way to advance with Lisp is to examine thse programs, particularly as the manual is, by intention, rather thin on anything but the vital information. The cartridge also contains a file called 'image' which is used by Lisp when it is loaded to provide, as it were, the core of the language. if you feel that you want to tailor the language to make it fit you better then it is possible to amend this file accordingly.

As I have already hinted, the manual is not for beginners. To be fair, it says that it is not and sets its sights firmly on experienced Lisp users. For them it is adequate, telling them how to get started, providing a reference guide, and leaving them to get on with it. For the beginner, there is a simple introductory session, a few example programs and a list of books on Lisp. I feel it is a pity to ignore the beginners like this, for they would swell the potential sales of the language considerably, and there is no doubt that armed with this Lisp and some suitable guidance they could go quite far and quite fast. I also think that the examples that are given are a little misleading to the beginner, for example, in carrying out repetitive tasks by iteration rather than recursion. Incidentally, my favourite book on Lisp, **The little Lisper** by Daniel Friedman (SRA), is not in the book list.

BCPL AND WHAT IT'S LIKE

BCPL is a procedural language and, as such, is more familiar in style to the BASIC programmer than Lisp. Its appearance does tend to conceal the similarity, but a brief familiarity is enough to confirm it.

The language 'C' is derived from BCPL, so if you have been 'Learning C' with Mark Woodley in Computing Today you will find that BCPL is not unfamiliar. Like 'C', BCPL is neither a high-level nor a low-level language, but comes somewhere in between. It allows you to 'get at' the hardware easily while providing the features necessary to write well-structured programs. it is a stark, sparse language which imposes few restrictions on the programmer while providing very little help. The inventive programmer will, of course, enjoy the freedom.

It is a strictly procedural language, and all programs must begin with a procedure named START which can call as many procedures as may be needed. All data is held as 32-bit words with no type explicitly associated with it. This allows a word to represent a number, a character, a pointer or anything else. In the same spirit, it is possible to perform any operation on a word, but BCPL is not going to be in a position to stop you doing something like multiplying two characters together. Sequences of words can be established for vectors, arrays and tables, and the indirection operator, familiar from BBC BASIC, is available for giving access to their individual elements. The data must initially be declared as GLOBAL or LOCAL to establish its scope.

A program can be dealt with in sections, with a section containing a number of procedures. This allows each section to be developed separately. When a program is run, all its sections must be linked together, and this provides the additional bonus that sections written in assembly code can be linked in with ease, perhaps to speed up the operation of a critical part of the program.

The following few lines should give an idea of what a BCPL program looks like:

```
LET START ( ) BE
$( LET v = VEC length
   LET no.of.characters = ((length + 1) *
BYTESPERWORD) - 1
   $( LET res = readstring(v, no.of.chars)
      WRITEF( ((res=-1) -> "*NString too
                              long",
                             "*NString =
                              %S*N"), v)
   $)  REPEATUNTIL  COMPSTRING(v,
                              "stop") = 0
                             $)
```

GETTING STARTED WITH BCPL

The previous description hints that BCPL does not operate in the same way as Lisp and BASIC and, indeed, it does not. Rather than providing an interpreter, it provides a compiler. The BCPL compiler can be run with the command

EXEC_W mdv1_bcpl

The cartridge containing BCPL must be left in place during compilation because the compiler uses overlays. The compiler first asks for the name of the file containing the BCPL program and then for a file in which to place the compiled version of the program. If you are developing a program you can simply press ENTER, not giving a name for the output file, and the program will be compiled and checked for syntax errors. After successful compilation, the compiled program must be linked with the BCPL runtime system before it can actually be run. This is done by running the linking progam with:

EXEC_W mdv1_blink

The linker requests the name of the file containing the compiled program and the name of the output file, and it then proceeds to place the compiled program and the runtime system in this file. The program can finally be run from this file using the EXEC_W command.

If a program consists of several segments then the segments can all be linked to form the complete program by giving the names of the files they are in to the linker when it asks for the file containing the compiled program. The linker contines to accept file names until you press ENTER on its query for another file name.

This procedure is not quite as arduous as it may sound, and it soon comes naturally. The effort involved is hardly worthwhile for small programs, but it provides considerable flexibility in the handling of substantial applications.

The manual with BCPL is a good deal more forthcoming than that with Lisp. This is mainly because there is hardly anywhere else to turn for information about BCPL, apart from a couple of publications by its originator, Maurice Richards. The manual provides all the information needed to get the system up and running, and also contains a complete reference guide. it also provides a tutorial on the language which would be a great help to the beginner, although I think that it would have to be supplemented, particularly in terms of what sort of things you might do in the language.

CONCLUSIONS

For anyone interested in Lisp and BCPL, these well-made implementations can be recommended with confidence. Even when run on an unexpanded QL, there is plenty of memory for developing substantial applications. After loading the Lisp interpreter, for instance, 50K of RAM remains available, and an expanded QL will obviously provide more.

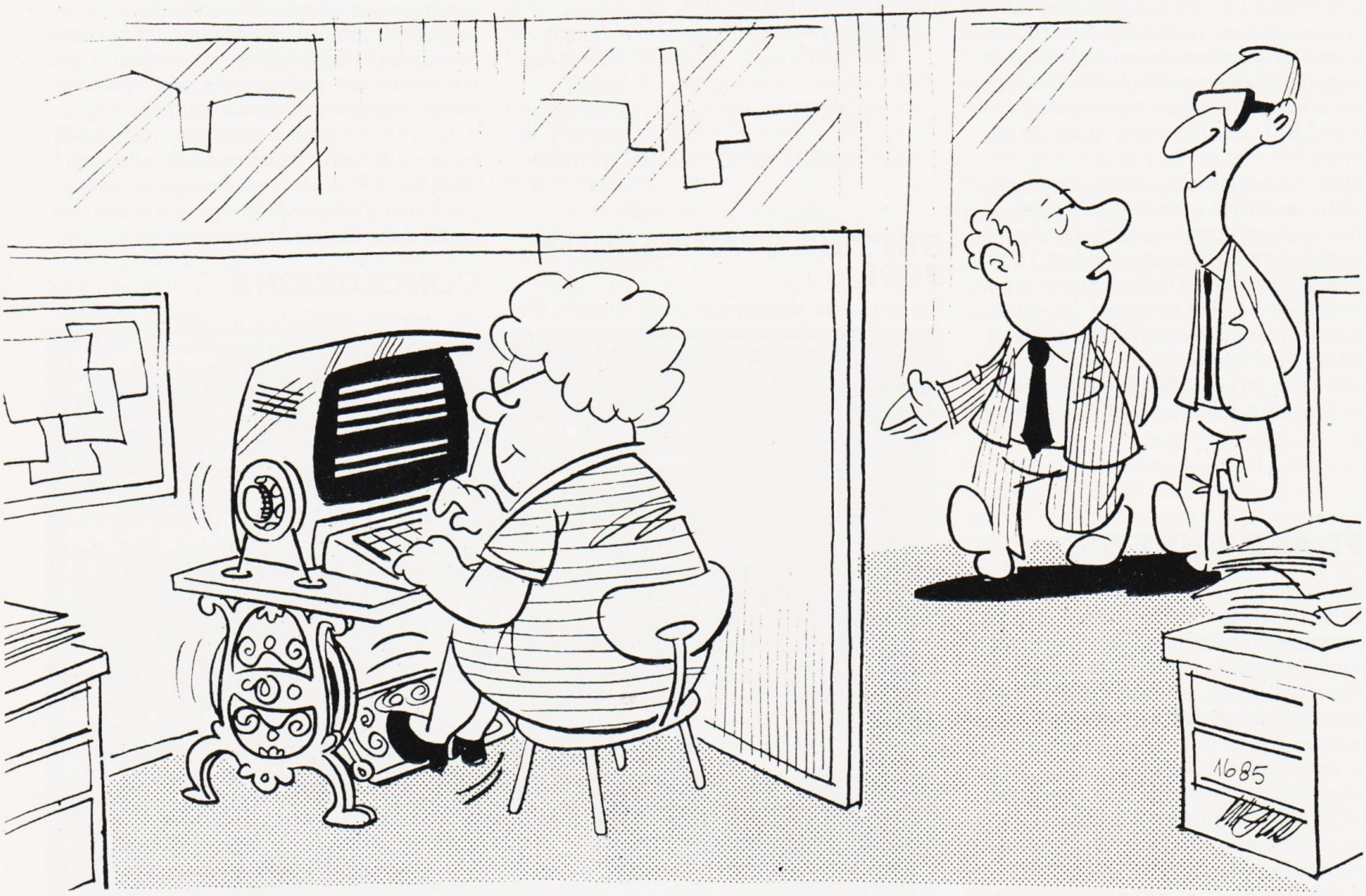
The diagnostics are helpful on the whole. Lisp's error tracing is very clear and helpful. BCPL's reporting of errors during compilation could be less ambiguous on occasion. Lisp could be more robust in operation, although it is the equal of any other microcomputer implementation that I have come across in this respect.

Both kits are suitable for the development of large projects. Lisp could also be used for learning the language, particularly in conjunction with suitable reading material such as Winston's **Artificial Intelligence** (Addison Wesley) and Friedman's little book. Anyone coming to BCPL would probably need to have experience with large software projects. We all have to start somewhere, but I do not think that there is enough supplementary material available to allow this kit to fill the bill.

This brings me back to the subject of manuals. Those supplied with the kits are adequate within their own terms of reference. I do feel that the provision of further manuals aimed at a lower level would enhance considerably the potential sales of the kits. The manuals for the versions of Lisp and BCPL that are available for the BBC Micro are cited as being suitable for this purpose. The BBC implementations are compatible with those for the QL, but the latter have more to offer. Besides, the BBC manuals are not as good as they might be. So, as a final thought, I would make a plea for the provision of manuals at a lower level, in addition to those that are already provided.

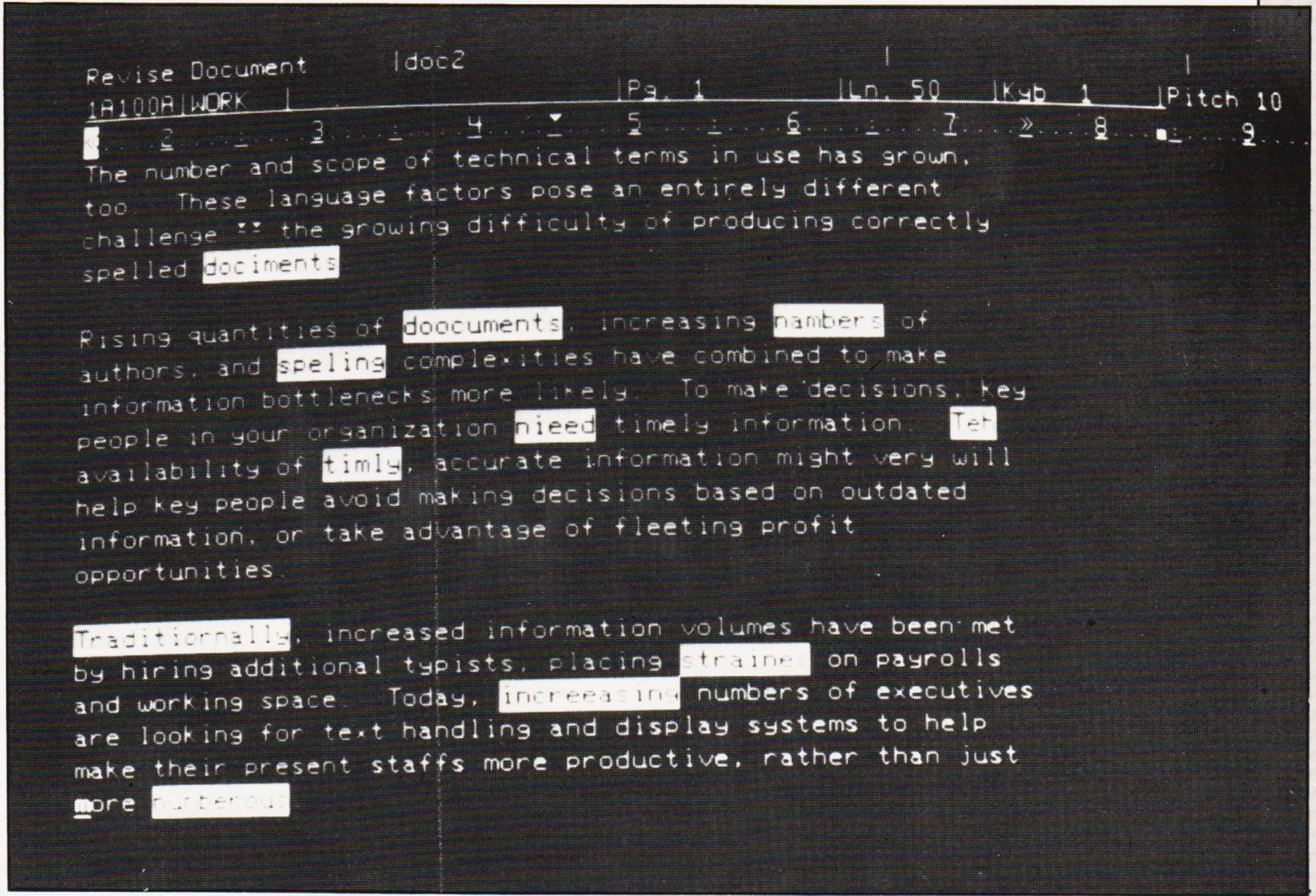
LIGHT RELIEF

Timo Mikama



INSIDE A WORDPROCESSOR

Bill Horne **takes a look inside a typical wordprocessing package.**



Word processors come in a variety of shapes and sizes, with facilities varying from minimal (to put it kindly) to overkill. The program examined here might be classed as 'low mid-range'. It satisfies most of the common requirements that arise in word processing, and can be extended by specialised user code, but leaves a useful area of RAM available for text storage.

Competition for store space between a word processor program and its data is almost inevitable, and judging the right balance can be difficult. Too small a text store means that its contents have to be saved to tape or disc at inconveniently frequent intervals, and there may be difficulty in transferring text from one record to another.

The program to be examined uses 8K of ROM for its code, and allows all but about 2½K of available RAM to be used for text storage. Since available RAM in the parent machine runs to nearly 48K, the text area is vast, and in practice is saved only as a precaution against catastrophes, or at the end of a session. In more recent machines, with colour, sound and other irrelevancies, the text area would be much smaller. That is called progress.

CONCEPT

The word processcr program divides into two main parts, one serving Edit mode, the other for Command mode. Edit mode allows text and control characters to be input, while Command mode sets up parameters and calls the main functional modules, such as print, save and load.

A word processor needs many of the normal control characters provided by an average microcomputer, but it also needs a number of additional controls, and if these cannot be covered by function keys there can be serious practical operating difficulties, some keys having to be given a number of different interpretations according to context. It is much better to use a basic machine which has an adequate number of function keys.

Another performance requirement for the basic machine is an adequate screen width. A 32-column screen is tolerable for some purposes, but a word processor may be asked to work with a printer which can cope with 128 columns, and that makes life difficult. One solution is to display only a limited number of columns at a time. In the program we are considering, it is possible to view columns 0-63, 32-95, or 64-127 at a given moment. That is not entirely convenient, but may be necessary where the full 128 column width is required, and layout is important. On the other hand, it is possible to reduce the line width so that the whole text is visible at one time.

This raises a long-standing argument regarding ideal screen widths. On A4 paper, the typing area is about 6 inches wide. At 10 ch/inch there are thus 60 columns, with 72 columns at 12 ch/inch. Even at the closer spacing, it is possible to work comfortably with 64 columns, but 80 columns are usually demanded. Unfortunately, an 80-column display is only readable with comfort on a good monitor, whereas a 64 column display can be much clearer. (And 64-column copy leaves a wider right-hand margin in which the Editor can make rude remarks!)

All this, however, applies only to the creation of text, as the final printout can be changed to any desired width. Where the text is simple, this creates no problem, but if the format is complex it is useful to be able to view it before printing.

Before leaving the subject of compatibility between performance requirements and hardware, the question of printer characteristics must be raised. Despite claims to the contrary, many printers have special facilities that call for additional control codes, and this means that extra program may be needed to generate such codes. It is sometimes necessary to cater for operation with different printers, and this leads to multiple print modes.

One conclusion which can be reached from points made above is that a word processor system based on a standard microcomputer can never be as satisfactory as a purpose-built device. That, to some extent, is true, but a standard machine can give an acceptable level of performance if some compromises are accepted.

EDIT MODE

To simplify processing, Edit Mode acts on a text line extracted from main store and held in the Line Buffer, which is 138 bytes long, to allow for a 128-column line plus some control characters. The text, and some control characters, are displayed on the screen.

When the Line Buffer is set up, it is filled with &0F entries. These are overwritten by incoming text from keyboard or main store, so any remaining &0F entries mark unused column positions. The current entry position is marked by a cursor, and may well contain &0F when it is at the end of the line, but if the previous entry position also contains a code less than &10, the 'off end' condition is reported on the top screen line, because there is a danger that an entry at the cursor position will become detached from the preceding text.

When the buffer is full, the system scans back to locate the last space character in the line. This is replaced with a line terminator code and the whole line is copied into main store. The next line is then brought out into the Line Buffer, starting at the terminator code. The characters following the last space therefore appear at the beginning of the new line.

The last line of a paragraph will not fill the buffer, and in this case it must be terminated by an Enter code, which becomes the line terminator. As before, the contents of the Line Buffer go to main store, and the next line is brought out, but this time there are no spillover characters at the start of the line.

Edit action inludes the provision of a Tab facility. At start-up, tabs are set at every tenth column, but they can be altered in Command Mode. Pressing the Tab key moves the cursor to the next tab position, filling in the intervening columns with spaces.

The start of a line can be inset by pressing Indent (Usually a function key). This sets up a special entry at the start of the line, with the format &1F, X, &1FR. X is the number of the column at which text is to start. Indent can only be used in the first fifteen columns, and X is determined by the next tab position.

Some Editors provide for insertion as a variant of overwrite, but this can lead to a confused display. Instead, Expand opens up a gap in the display, and extra data can be inserted within the gap. Pressing Expand again restores normality. (Another function key).

Cursor Left and Right can be used to re-position the cursor on the current line for the purpose of making alterations, but Cursor Up and Down store the current line and bring out the previous or subsequent line.

Scan changes the section of the line displayed, and Rub is the usual erase function. Less familiar is 'Soft Hyphen'. This puts a code in the text to indicate that a hyphen is permissible at that point if it ends a line, but should be omitted in mid-line.

CODE VARIATIONS

Tracing out a Word Processor program is not made easier by the fact that given codes have to have different meanings in different contexts. For example, &01 as an input code means Tab, but &01 in the text marks the start of text. In input, &02 is expand, but in text it means end of text.

Each individual line, created on the basis of the displayed line length, is terminated by a code less than &10, so a given line can be found by starting at the beginning of text and counting off the terminating codes. To help in locating particular text, the line and column number for the present cursor position are shown on the top screen line.

In addition to line terminators, the text may contain a number of control codes in the &10-&1F range. Some of these are for the control of print action, such as Form Feed, Upshift and Downshift. (For superscripts and subscripts.) In all, the text stream holds complete printing instructions.

However, the original text stream may have been created in a manner suited to the display, rather than the final printout, and it can be re-arranged in a different way by Command Mode. The terminator &0C or &0E is used as a 'soft return', meaning a return which will be ignored in re-formatting. This turns a complete paragraph into a single text stream, ter-

INSIDE A WORDPROCESSOR

minated by the final Enter code, and the re-formatting process can work out the appropriate lines for any given line length, displaying the result.

Before leaving the subject of code variations, it should be said that in the printout process codes &01 to &0C specify a number of spaces to be output.

COMMAND MODE

There are twenty-two Command functions, each selected by an upper case letter input when the program has returned to the Command mode loop point. The other four letters give a jump into RAM, where a further jump normally reports 'Invalid Command' on the top screen line, but the RAM jump can be altered to access user code.

The command **Macro,** without qualification, moves data from between the current cursor position and the end of text into the Macro Buffer. Up to 512 bytes can be copied. If a numeric qualifier is appended to the command, the number specifies how many times the macro is to be executed. This allows a command string to be set up and called whenever required, since the Macro is read as if it was the Command input buffer.

Back with a numerical qualifier, moves the cursor back the specified number of lines. Without a qualifier, the command moves the cursor back to the start of text, or to a &18 code, called Mark, which can be set up in Edit.

Delete erases either all text from the cursor position on, if unqualified, or n lines from that position if qualified by n.

End takes the cursor to the end of text.

Forward with a qualifier, moves the cursor forward n lines.

Erase clears all text. Since its effect is so drastic, the user is asked to confirm his intention before anything is done.

Length, unqualified, displays the current line length on the top screen line. With a qualifier, it sets line length. This is the length used by Edit, and by Command mode in dividing text up into line segments.

Memory reports remaining free memory on the top screen line.

Search looks for a segment of text which matches the data qualifying the command.

Top takes the cursor to the start of text.

Hold n puts n lines of text into the Hold buffer, beginning at the current cursor position. Hold 0 clears the buffer, which must be done before calling Hold n again.

Unhold copies data from the Hold buffer into the text stream.

Exit escapes from the program to the main operating system.

These are the minor functions, though some are extremely useful. The major functions are more complex.

PRINT

The main memory is always organised in accordance with the current length setting, and is reformatted when length is changed. Printed out in this form, the text would be 'rag edge', i.e. with the right-hand edge in varying positions. This can be altered in printout by the contents of the 'Y' table.

First, there is 'Print Option', which specifies the type of printer. Option 2 is standard, but options 0 and 1 require special user code. This may allow serial output instead of the normal parallel form.

The second Y Table entry allows print action to be stopped at the end of each page, the third entry specifying the number of lines in a page. This is useful when non-continuous stationery is in use. The fourth entry allows a number of line feeds to be specified, to be inserted when page end is reached. This is for continuous stationery.

Next, insertion of a repeated page title can be called up. The title line is placed at the start of text, and can include a page number on the right. This is automatically incremented.

Extra line feeds can be specified to give wider line spacing.

Then comes the most important of the Y table conditions: Right Justify. If this mode is specified, each line is padded out with spaces to be of uniform length, giving a regular right-hand page.

The method of doing this is interesting. Each single space is initially represented by code &20, as usual, but this is changed before a line is output to the printer, the space codes being overwritten by &01 codes. The length of the line is checked, and the result deducted from Length. The different indicates the number of spaces which need to be inserted. The line is scanned, and the &01 entries are changed to &02 until either the end of the line is reached or the number of changes matches the difference between actual length and Length. If the end of the line comes first, the process begins to turn &02 entries into &03, and so on.

The print driver recognises codes &01 to &0C as specifying a number of spaces to be output. So simple, really...

The remainder of the Y Table is specialised for proportional spacing printers, and is unused unless special user routines are provided.

The Y Table settings can be changed by Command, and so can the Tab Table settings, giving wide-ranging control over printout style. The Y Table also specifies a standing indent, moving the left-hand edge of text to the right.

BACKUP STORAGE

The sytem provides for backup storage on tape, but does not cover discs in its original form, though it has been adapted to run in conjunction with CP/M, at some sacrifice of main store space.

One or two tape recorders can be used, and part or all of the stored text can be saved. Each record has a filename, and can thus be seleted for reading without the possibility of ambiguity. The record read is added to the existing record, if that is not deleted first. Since sections of text can be moved around by use of Hold, it is possible to assemble a combined version by combining parts of two source versions.

It is even possible to execute 'mail-shot' processes, using the Hold facility to contain the standard text and the main program to hold the variations, but Hold is a little limited in size for this purpose.

CONCLUSION

The objective has been to outline the principal functions of a small word processor in terms which would illustrate the type of coding needed. Once the general principles are understood, writing the necessary code is not unduly difficult. Much of the action is executed by short routines, backed up by some common-use subroutines. The difficulties lie in determination of the overall structure, the choice of coding, and other details.

The routine described is partly written in 8080 code, which rather dates it, though some sections have been revised in Z80 code to save space. It has subsequently passed through a number of upgrades, one of which is well known as 'Spellbinder'. The relationship is rather remote, but can be seen on close examination. Spellbinder goes much further, using disc instead of tape, the latter being an optional extension, and providing much-extended facilities.

If the older routine has a fault, it lies in a tendency to miss characters now and then, especially at the end of a line, when quite a lot of processing is needed in the 50 mS or so between key presses, a failing which could be overcome by availability of a keyboard buffer.

If you already have a microcomputer, the foregoing description should tell you whether it is likely to make a good word processor. If you are considering purchase, you should be able to see which models will serve your prupose best. Naming names would be neither popular nor really necessary. The limitations of unsuitable machines are only too obvious, despite publicity to the contrary.

THE ALVEY PROGRAMME

Jamie Clary

Most readers will have heard of the Alvey Programme — but who and what are involved in it?

Much is made of the race towards the 'fifth generation' of computers and computing technology, with Japan's highly publicised fifth generation computer programme apparently leading the field, followed closely by the Americans.

But what of the UK's efforts? We are generally acknowledged as being hosts to considerable expertise in information technology, but very few of our own achievements ever 'hit the headlines', usually because of the low public profile adopted by British industrialists and academics (with one or two notable exceptions).

Early in 1982, a special committee was set up by government to "...advise on scope for collaborative research in Information Technology". Overseas competition in IT was getting progressively fiercer, and the Japanese programme was already in progress.

Headed by John Alvey (then British Telecom's Chief Engineer) the committee submitted their report in September, 1982. Entitled "A Program for Advanced Information Technology", the report urged government to begin a five-year programme, costing £350m, which would eventually involve three government departments: The Department of Trade and Industry, the Ministry of Defence, and the

Department of Engineering and Science, the latter acting through the Science and Engineering Research Council.

The government accepted the committee's recommendations, offering £200m with a request that industry should provide the balance. This was to be the beginning of the Alvey programme, announced in April 1983, and designed to make the most of Britain's expertise in IT research.

The Alvey Directorate was subsequently established, drawing 45 people from the DTI, the SERC, the MoD, and from industry. The directorate, lead by Brian Oakley and chaired by Sir Robert Telford, were to collectively steer the programme, and define the strategy within which projects funded by Alvey were to fit.

YEAR 1

The first year was spent clarifying the guidelines and strategies of the programme, and establishing communication channels between them and the academic/industrial community, with Alvey directors accepting numerous invitations to speak about their work and thus making contact with the community. During this time, the Alvey team received over 300 research proposals, of which 100 - consisting of 60 industrial and 40 academic projects - have been given technical approval. Much of the academic work is done in a few carefully selected 'centres of excellence' where teams drawn from different departments are specifically united to work on Alvey projects.

Work on most projects is done by various consortia, typically consistin gof two or three firms and one or two universities/polytechnics. However, some projects are of such a speculative nature that it is often too early for industry to be directly involved. However, such projects **do** need people from industry, to ensure that the work is in an area of industrial interest, and to advise the directorate when industrial participation is appropriate. Projects of this nature are termed 'uncle' projects.

CATEGORIES

Alvey operations have now fallen into five main categories: **Infrastructure and Communications**, providing information and support for persons and institutions involved in the programme; **Man-Machine Interface** (MMI), concerned with improving the methods of exchange between humans and computers; **Intelligent Knowledge Based Systems** (IKBS), encompassing a variety of research in AI-related areas; **Software Engineering**, formalising programming method and improving the reliability of software; and **Very Large Scale Integration** (VLSI) **and Computer Aided Design** (CAD), set up to ensure British access to highly developed VLSI components in order to guarantee a measure of international competitiveness for the future.

WORK CATEGORIES IN DETAIL

Within the five operational areas further 'research themes' are pursued, particularly within IKBS (Intelligent Knowledge Based Systems) and MMI (Man Machine interface).

IKBS has nine research themes, as follows:

Parallel Architecture;
Declarative Languages;
Intelligent Knowledge Based Systems;
Expert Systems;
Intelligent Front Ends;
Inference;
Natural Language;
Image Interpretation;
and Intelligent Computer Aided Instruction

Seven GEC Series 63 computers and four Systime B750 machines (based on VAX 11/750 processor boards) have been installed in research centres for work on the IKBS program, plus a further four machines-for-software engineering work. The systems all run UNIX as a common operating system, and the machines are networked using ALVEY-NET (an advanced communications network designed

LARGE SCALE DEMONSTRATOR PROJECT

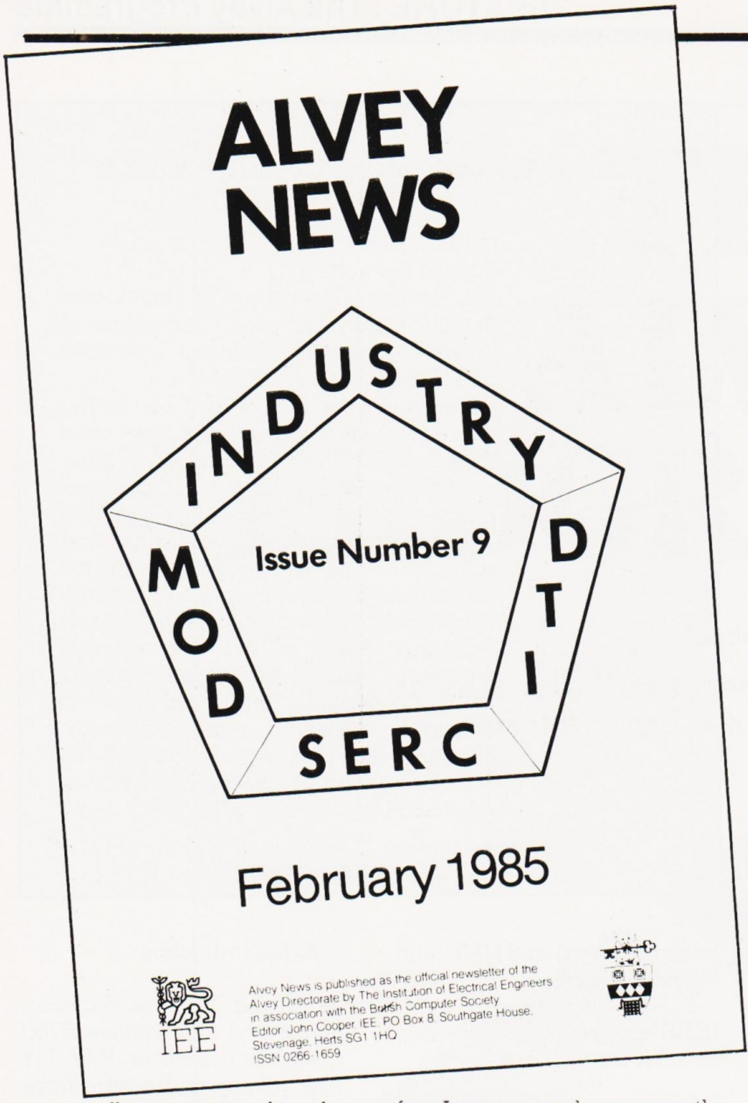
Project title:	DHSS Demonstrator
Project leader:	Charlie Portman, ICL
Consortium:	ICL, Logica, Department of Health and Social Security (as user), Universities of London (Imperial College) Lancaster and Surrey
Aims:	To evaluate the role of new computing technologies in assisting legislation-based organisations in the formulation, interpretation, and application of complex policies and rules. Detailed investigations and demonstrations of IKBS, MMI, and software engineering techniques will be undertaken as part of the project. Software engineering tasks will include the evaluation and choice of the most suitable languages, techniques for speedy access to large databases, and formal means of preparing software more efficiently. Further work in IKBS includes methods for knowledge acquisition, representation fo rules and the means of using this.

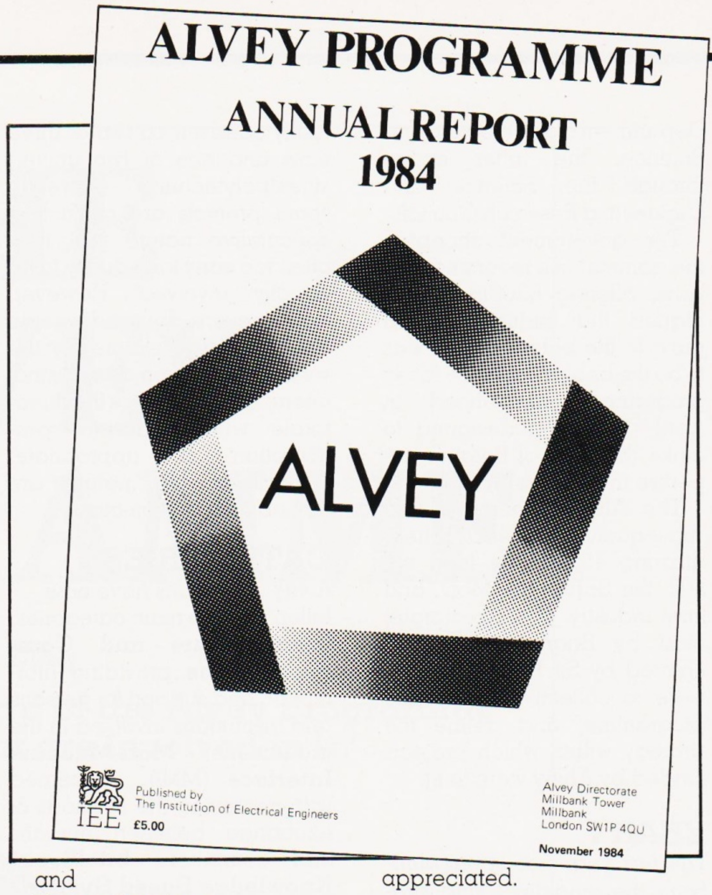
LARGE SCALE DEMONSTRATOR PROJECT

Project title:	Design to Product
Project leader:	Rex Tomlinson, GEC Elecrical Projects Ltd., Rugby
Consortium:	GEC, Edinburgh University (AI Dept.), Loughborough and Leeds Universities, Lucas CAV (as user), National Engineering Laboratory
Aims:	To demonstrate the automation of a total production process, from design through manufacture to maintainance in the field. It represents the next generation of computer integrated manufacturing, and will use IKBS techniques throughout to capture and apply the skills of the human operators.

LARGE SCALE DEMONSTRATOR PROJECT

Project title:	Speech Input Wordprocessor and Workstation
Project leader:	Peter Schwarz, Plessey
Consortium:	Plessey, Shell (as user), Universities of Edinburgh, London (Imperial) and Loughborough
Aims:	Practical speech recognition; speaker-adaptive recognition of connected speech, with a vocabulary of 5000 words at minimal error-rates. Bulk of speech research to be carried out at Edinburgh. Three departments will work together on the project: Electrical Engineering, Linguistics and Artifical Intelligence, managed by the new Centre For Speech Technology. their task will be to generate the entire transcription system, using a large amount of research in natural language understanding and phoenetic speech extraction aimed at the understanding of unconstrained speech.





especially to meet the demands of the Alvey programme).

One of the aims of Alvey in IKBS is to achieve standardisation of languages commonly used in AI. PROLOG, LISP, and POPLOG are all supported by the programme, but efforts are being made to limit the number of 'dialects' in order to maintain compatibility.

'Community Clubs' have been formed, within which member companies contribute £5000-£10000 to a common pool, the Alvey directorate matching contributions on a pound-for-pound basis. The money is used to sponsor work into areas which will be of mutual benefit to the member companies and to the AI industry as a whole. One such club, launched in collaboration with the Royal Institute of Chartered Surveyors, is looking into the use of IKBS techniques in cost-estimation for major office building work.

An IKBS 'Journeyman' scheme has been set up to enable industry to send a limited number of staff for six-month hands-on training in IKBS techniques at established centres of excellence in the UK. The first four Journeymen have recently joined their selected centres of excellence. Two candidates, from ICL Defence Systems and Burroughs at Cumbernauld, are now attending the Turing Institute in Edinburgh, while two others - from British Rail and Hewlett Packard - are at Imperial College, London.

WORKSHOPS

Workshops are held in conjunction with the IKBS programme, with titles ranging from the well-known topics ('Expert Systems') to the not so well-known ('Inference Under Uncertainty').

The principle challenge for the programme is to produce a larger pool of manpower skilled in IKBS techniques. Specialist MSc courses have been created, expected to produce between 25-50 graduates per year, although this is still below the expected demand for expertise in this area, especially in the future as interest in IKBS begins to develop.

MMI has three main research themes:

Pattern Analysis;
Displays;
and
Human Interface

So-called 'large scale demonstrators' - Alvey projects which set practical targets (see boxes) - are important to the MMI project. Technological advances per se are inadequate, and there is a need to demonstrate that such advances are useful in the industrial sector. Also, the demonstrator projects are of particular use, as they provide early examples of MMI where human interaction problems are not yet fully understood or appreciated.

CONCLUSION

To summarise, the Alvey programme represents significant progress towards a coherent national approach in solving the problems of the fifth generation of computing. The large scale demonstrator projects, although only four in number at the last count, show that some quite 'heady' goals have been set. We hope that at least one of them is achieved by the end of the programme - April 22nd, 1988.

LARGE SCALE DEMONSTRATOR PROJECT

Project title:	Mobile Information Systems
Project leader:	Dr John Walker, Racal
Consortium:	Racal, Acorn Computers, Universities of Cambridge, Loughborough (HUSAT), Surrey, Sussex, and the Open University. Users are BL Technology, The Transport and Road Research Laboratory, The RAC, and the Electricity Council.
Aims:	IT techniques to be employed to provide a number of services to the motorist, including: **road guidance** to plan and guide optimum routes for motorists; **traffic information** for up-to-date information on traffic incidents and road conditions; **a mobile electronic office** with portable terminals to provide a remote mobile terminal for computing, access to databases, wordprocessing, electronic mail, expert systems etc.; **mobile IKBS** with access to a wide range of IKBS. Acorn's experience of volume markets for computer products considered 'useful for the project'.

PHOTOCOPIES

A COMPUTING Today Reader Service

To take advantage of this simple service, just fill in the required information and send it (or a photocopy) to:

COMPUTING TODAY Photocopies,
No. 1, Golden Square,
London W1R 3AB.

together with your money and we'll do the rest!

CT PHOTOCOPIES

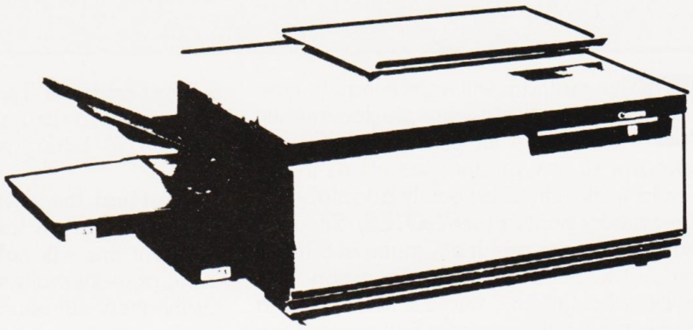
NAME ....
ADDRESS ....
....
.... POSTCODE ....
Please send me Photocopies of the following items

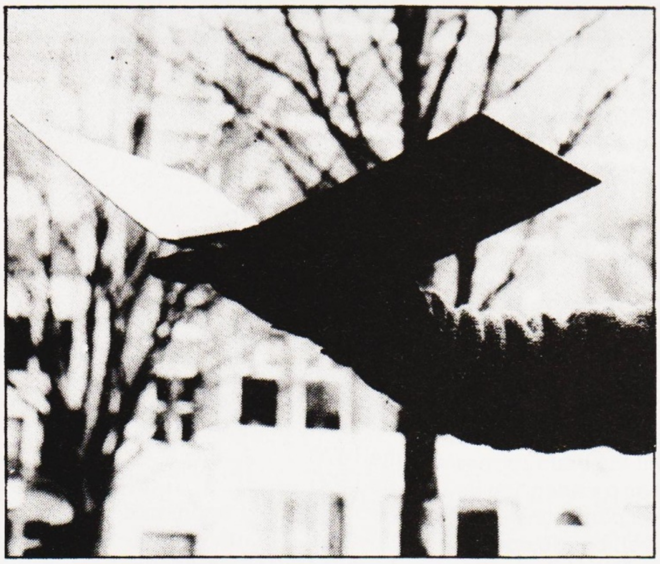
ISSUE	ARTICLE	PAGES

At £1.50 each, I enclose £ ....
Cheques and Postal Orders should be made payable to ASP Ltd.

Lost and can't be replaced? Well, if you've lost one of the early issues that cannot be replaced from our stocks of backnumbers, all is not quite lost.

If you know the article name and the issue it appeared in, we can supply you with a photocopy for the miserly sum of £1.50 including postage and packing. If you're stuck for the month, try checking in our annual index which is included in the December issue each year.





In days gone by, falconry was the sport of gentlemen and kings — this noble and time-honoured tradition is not so prevalent in these technological times, and it is quite a pity, too. Just imagine the pride you'd feel standing in your own back yard while your very own hunting falcon swooped down upon unsuspecting dogs, cats and Ford Sierras.

For a limited time only, Computing Today is offering you the chance to experience the thrill of commanding your own bird of prey, with the new CT Hunting Falcon/Magazine Binder. Swift of wing, sure of eye and made of genuine vinyl and cardboard, the Computing Today Hunting Falcon/Magazine Binder is the spitting image of the hunting birds of old to anybody suffering from cataracts. Release it from your arm, and it dives just like a traditional hawk. If it lands on a small animal, it will probably stun it. Also, when you tire of the sport, and would rather hunt hedgehogs with your Ford Sierra, your CT Hunting Falcon converts into a useful magazine binder that holds a full year's supply of Computing Today. The new CT Hunting Falcon/Magazine Binder will cost you not a farthing more than the old binder alone used to: just £5.00. This includes postage and packing, so your falcon won't have to tire itself out flying to your abode.

Cut out and send to:

COMPUTING TODAY HUNTING FALCON/MAGAZINE BINDER, INFONET LTD, TIMES HOUSE, 179 THE MARLOWES, HEMEL HEMPSTEAD, HERTS HP1 1BB, ENGLAND

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)

Insert card no. [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss) ....
delete accordingly
Address ....
....
....

Signature ....

Date ....

BBC TEXT EDITOR

C R Dickens

A modest text-editing program for the BBC Micro

This program will enable you to type letters or to make simple documents. It is written for an Epson FX80 but there are so few printer instructions that it should work with or be easily adaptable to other similar printers (see NOTES). The program is run from one main menu of 8 items some of which ask a further question. The keyboard of the BBC micro is used just as a normal typewriter except in Edit mode when two keys allow insertion or deletion of a string of characters. Following is a description of each menu item and these also serve as instructions in using the program.

1. **Type new text**. This enables you to start. Type normally — the cursor keys should not be used. User <DELETE> to delete back to a mistake. If the mistake is a long way back don't bother; you will be able to make corrections at any time by going to Edit mode, item 6. <ESCAPE> at any time returns to the menu.

2. **Add from tape**. A previously written document, (item 8) can be read back. It will be added on to anything that has been typed or has been read before. If you wish to be clear before reading in text then move to 1. on the menu first.

Text is written to tape compactly by using BPUT and BGET, 256 characters being in each block. The normal file handling messages and rules apply.

3. **Set print style**. This item will enable you to use some of the features of your printer. See the program notes below if you wish to change some of the printing features. e.g. you may wish to change the margin size of 8 characters set in this program. You can have proportional spacing for those 'types' which are allowed. Four 'types' are offered for you to choose from.

If the printer was not switched on when the program was first <RUN> then it will be necessary to use menu item 3 to automatically set the margins, particularly useful for neat letter writing.

4. **Print address**. The date is asked for. Any characters may be typed in here. The printer must be ON LINE now.

5. **Print text**. the text is printed in the chosen style. You can ask for right margin justification, but this will not work correctly when using proportionally spaced type. This right justification, although lining up correctly on the margin, is a little pseudo. Text should have been finished with a <RETURN> otherwise, with right justification, the last line will be spaced across the page.

6. **Edit text**. It would not be strictly true to say that this mode provides full screen editing but the operation of the program is very close to much cleverer packages. Here the text is wordspaced to 40 columns for easy screen reading.

There are 3 functions:

a) Replace: The cursor keys can be moved to any character which can then be replaced by simply overtyping with the replacement character. Note that a newline character, <RETURN>, occupies a position on the screen at which it is typed and also in the text. But the extra spaces following the last word on a screen line do not. A newline character can be inserted, see b), or deleted, see c), but do not use <RETURN> unless Replacing the character at the cursor position with a newline character.

b) Insert: <CTRLI>, (holding <CTRL> and pressing <I>), will enable you to type in additional characters which will be inserted BEFORE the cursor. A newline character cannot be inserted. For this, first insert any character then Replace it (see 6a) with <RETURN>. Do not insert characters such that the total is more than 9000.

c) Delete <DELETE> will ask for the number of characters to be deleted from the cursor position onwards. Remember that newline characters should be included in the count if desired.

The text is displayed 20 lines at a time on the screen. The <ESCAPE> key will move on to the next 20-line page, finally returning you to the menu.

Storage allows for 240 of these Edit-mode lines, or 9000 characters. The number of characters is always displayed at the bottom of the screen.

Use "Edit Text" to read through your document.

7. **Continue text**. This item displays the whole of the text, again wordspaced for easy reading. The cursor is placed at the end of the text and typing may continue as under item 1. <ESCAPE> at any time returns to the menu.

8. **Write to tape**. Here the text can be saved to cassette using the normal file handling rules.

PROGRAM NOTES

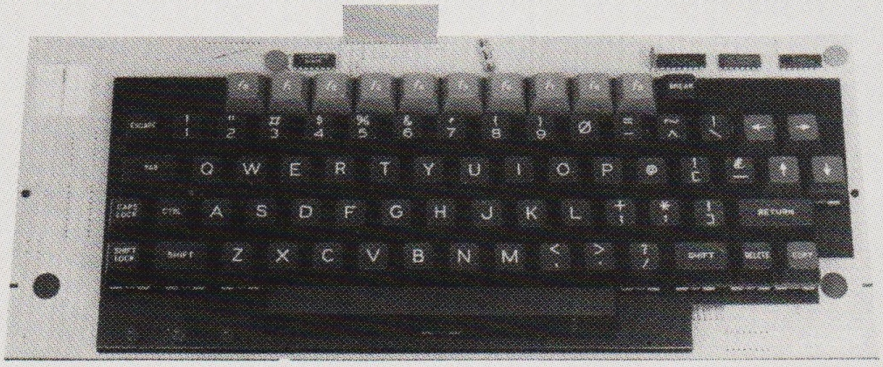
These notes are mainly concerned with the printer codes. if necessary, they can be changed to suit you own requirements.

Line 40	Send all characters, including LINE FEED, to the printer.
Line 50	Allow ESCAPE key to be read as a value.
Line 70	"Line" is number of normal characters per line; "on target" is the printer code for setting or cancelling enlarged mode which must be sent to the printer for each line when set.
Linr 80	Select printer (2); initialize (1,27,1,64); left margin (1, 27,1,108); at 8 characters (1,8); de-select printer (3).
Lines 90-160	Your own address.
Line 800	Form-feed (1,12) at end of printing and between multiple copies.
Line 1090	PROCcalc (line 1220) and PROCfill are Right Justification procedures.
Line 1170	Formfeed (1,12) after 56 lines are printed.
Line 1470	As line 80.
Line 1500	(1,27,1,112,1,49) selects proportional spacing. Not valid for Elite mode. "props" is approximately the extra number of characters per line you will get whilst using proportional spacing.
Line 1520	(1,27,1,77) Elite size characters - 12 per inch.
Line 1540	(1,15,1,14) gives condensed enlarged characters.

This program adequately performs the services of a letter-writer. it will enable the purchaser of a printer to become familiar with a few of the functions of a word processor before finally assessing more sophisticated software.

Listing: BBC Text Editor. Note, for '£' type '#'

```
L.
   10 DIM A 9000,AX 10,P%(241),AD$(7)
   20*OPT1,1
   30*OPT2,1
   40VDU15:*FX6
   50*FX220,0
   60TYPE=FALSE:JUST=FALSE:FEED=FALSE
   70line=65:enlarge=20:J=0:R%=0:JA%=0
   80VDU 2,1,27,1,64,1,27,1,108,1,8,3
   90AD$(0)="'Chez moi',"
  100AD$(1)="  Apple Tree Road,"
  110AD$(2)="    Nomansland,"
  120AD$(3)="      Nr.Stonehenge,"
  130AD$(4)="        Cidershire."
  140AD$(5)="          AB1 2CD"
  150AD$(6)="           "
  160AD$(7)="     tel: Bbcmicro 390295"
  170REPEAT:PROCMENU:UNTIL FALSE
  180DEFPROCMENU:CLS:VDU26
  190PRINTTAB(5,2)"Select from the following"
  200PRINTTAB(0,5)CHR$133"1      Type new text"'
  210PRINTCHR$133 "2      Add from tape"'
  220PRINT " 3      Set print style"'
  230PRINT " 4      Print address"'
  240PRINT " 5      Print text"'
  250PRINT " 6      Edit text"'
  260PRINT " 7      Continue text"'
  270PRINT " 8      Write to tape"
  280PRINTTAB(5,22)"Type 1,2,3,4,5,6,7 or 8"
  290REPEAT:R$=GET$:UNTIL R$>"0" AND R$<"9"
  300PRINTTAB(0,3+VALR$*2)CHR$131
  310VDU7:G=INKEY(50)
  320IF R$="1"CLS:J=0:PROCtype
  330IF R$="2"PROCinput
  340IF R$="3"PROCstyle
  350IF R$="4"PROCaddress
  360IF R$="5"PROCjust:FORW%=1TOV%:PROCprint:NEXT:JUST=FALSE
  370IF R$="6"R%=0:S%=20:K=0:PG%=0:REPEAT PROCedit:PG%=PG%+1:
       UNTIL NOT (K<J)
  380IF R$="7"K=0:R%=0:S%=241:CLS:PROCDISP(39):PROCtype
  390IF R$="8"PROCsave
  400ENDPROC
  410DEFPROCtype
  420REPEAT JA%=J:PROCscore:A?J=GET
  430IFA?J=127 VDU127:J=J-1 ELSEPROCgoto
  440UNTIL A?(J-1)=27 OR J>9000
  450IFJ>9000SOUND1,-10,200,40
```



```
460J=J-1:TYPE=TRUE:ENDPROC
470DEFPROCgoto:PRINT CHR$(A?J);
480IF A?J=13 VDU10
490J=J+1:ENDPROC
500DEFPROCedit:CLS
510PROCDISP(39):VDU 30
520*FX4,1
530REPEAT
540*FX21,0
550Y=GET
560*FX21,0
570IF Y=138 IF VPOS<20 VDU 10
580IF Y=139 IF VPOS>0 VDU 11
590IF Y=136 VDU 8
600IF Y=137 VDU 9
610IF Y=127 PROCPOS:PROCDEL
620IF Y=9 PROCPOS:T=T-1:PROCIN
630IF Y>12ANDY<127ANDY<>27 PROCPOS:VDU9,127:A?T=Y:PRINTCHR$(Y);
   :IFY=13VDU10
640UNTIL Y=27
650*FX4,0
660ENDPROC
670DEFPROCaddress:CLS
680PRINT "TYPE DATE  ";
690INPUT LINE D$
700VDU 2:FOR I=0 TO 7
710VDU1,enlarge
720PRINT SPC(line*.6);AD$(I)
730NEXT:PRINT
740VDU1,enlarge
750PRINT SPC(line*.6);D$
760PRINT:VDU3:ENDPROC
770DEFPROCprint
780IF TYPE=FALSE VDU 7:ENDPROC
790CLS:VDU 2:S%=1000:K=0:R%=0
800PROCDISP(line):IF FEED VDU1,12
810VDU3:ENDPROC
820DEFPROCIN:VDU7
830PRINTTAB(0,22)CHR$134;
840INPUT LINE"Type characters to be inserted"insert$
850len=LEN(insert$)
860FOR C%=J-1 TO T+1 STEP -1
870A?(C%+len)=A?C%
880NEXT
890FOR C%=1 TO len
900A?(T+C%)=ASC(MID$(insert$,C%,1))
910NEXT:K=KA:R%=RA%
920J=J+len:CLS:PROCDISP(39):VDU 30
930ENDPROC
```

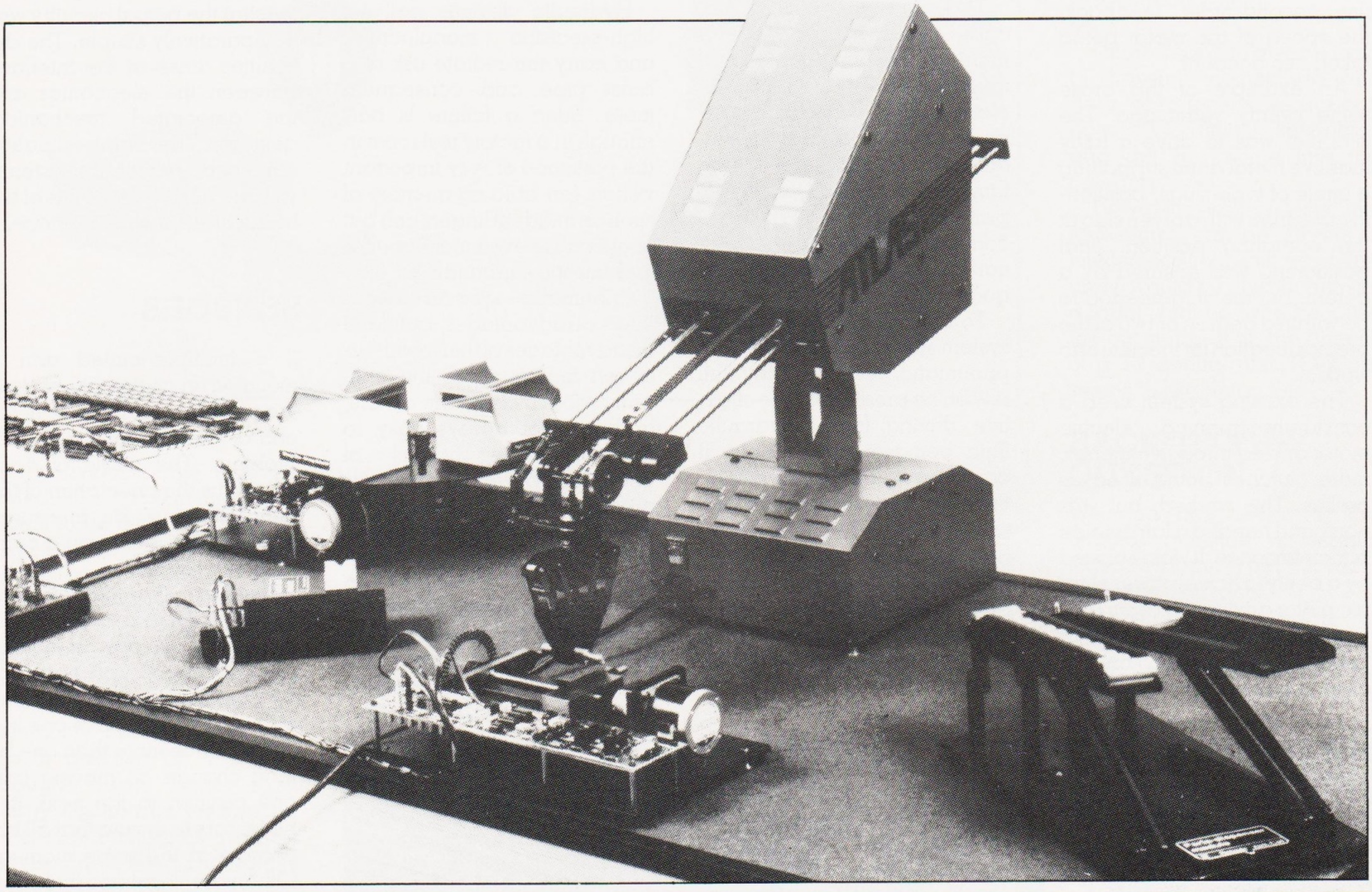
```
940DEFPROCDEL:VDU7
950PRINTTAB(0,22)CHR$134;
960INPUT" How many characters to delete"delno:
   IFdelno+T>J delno=J-T
970FOR C%=T TO J-(delno+1)
980A?C%=A?(C%+delno)
990NEXT:K=KA:R%=RA%
1000J=J-delno:CLS:PROCDISP(39):VDU 30
1010ENDPROC
1020DEFPROCDISP(L):RA%=R%:KA=K
1030IFJ=0ENDPROC
1040REPEAT
1050I=(K+L):IF I>J-1 I=J-1 ELSEREPEAT I=I-1:UNTILA?I=32 OR A?I=13
1060VDU1,enlarge
1070IF K>I I=(K+L)
1080H%=L-I+K-1
1090IF JUST IFH%>0PROCcalc
1100FOR M=K TO I
1110PRINTCHR$(A?M);
1120IF JUST IFA?M=32 AND H%>0 PROCfill
1130IF A?M=13 THENH=I:I=M:M=H
1140NEXT
1150IF(R$="7"ANDI>J-2ANDNOT(A?(J-1)=13))PRINT;ELSEPRINT
1160P%(R%)=I:R%=R%+1:IFR%>240SOUND1,-10,200,40:I=J
1170IF FEED ANDR$="5"ANDR%MOD56=0 VDU1,12
1180IFNOT(R$="5")JA%=J:PROCscore
1190K=I+1
1200UNTIL K>=J OR (R% MOD S%=0)
1210ENDPROC
1220DEFPROCcalc:C%=0
1230FOR M=K TO I:IFA?M=13 H%=0
1240IFA?M=32 C%=C%+1
1250NEXT:IFC%=0 ENDPROC
1260D%=H%DIVC%:E%=H%MODC%
1270ENDPROC
1280DEFPROCfill
1290IFD%>0 FORI%=1TOD%:PRINTSPC(1);:NEXT
1300IFE%>0 PRINTSPC(1);:E%=E%-1
1310ENDPROC
1320DEFPROCPOS
1330X%=POS:Y%=PG%*20+VPOS
1340IF Y%=0 THEN T=X%:ENDPROC
1350T=P%(Y%-1)+X%+1
1360ENDPROC
1370DEFPROCstyle:CLS
1380PRINTTAB(5,2)"Select type"
1390PRINTTAB(1,5)"1      Pica ie.normal"
1400PRINT " 2      Elite"
1410PRINT " 3      Italic"
```

```
1420PRINT " 4      Condensed enlarged"
1430PRINTTAB(5,10)"Type 1,2,3 or 4"
1440REPEAT:B$=GET$:UNTIL B$>"0" AND B$<"5"
1450PRINTTAB(0,4+VALB$)CHR$131
1460enlarge=20:props=0
1470VDU 2,1,27,1,64,1,27,1,108,1,8,1,27,1,65,1,20,3
1480PRINT TAB(2,15)"Do you want proportional spacing?"
1490PRINT TAB(13,17)"Type Y or N":PROCANS
1500IF G$="Y" OR G$="y" THEN IFB$="1"ORB$="3"
    VDU2,1,27,1,112,1,49,3:props=6
1510IF B$="1"line=65+props
1520IF B$="2"VDU 2,1,27,1,77,3:line=80
1530IF B$="3"VDU 2,1,27,1,52,3:line=65+props
1540IF B$="4"VDU 2,1,15,1,14,3:line=55+props:enlarge=14
1550ENDPROC
1560DEFPROCjust:CLS
1570INPUTTAB(10,7)"How many copies "V%
1580PRINT TAB(3,10)"Do you want right justification?"
1590PRINT TAB(12,12)"Type Y or N":PROCANS
1600IF G$="Y" OR G$="y" JUST=TRUE
1610PRINT TAB(9,15)"Do you want FORM FEED?"
1620PRINT TAB(12,17)"Type Y or N":PROCANS
1630IF G$="Y" OR G$="y" FEED=TRUE
1640ENDPROC
1650DEFPROCinput
1660PRINTTAB(0,22)SPC40
1670INPUTTAB(0,20)"Type file name "$AX
1680X=OPENIN($AX)
1690REPEAT:A?J=BGET£X:J=J+1:UNTILEOF£X OR J>9000
1700CLOSE£X
1710TYPE=TRUE:J=J-1:ENDPROC
1720DEFPROCsave
1730IF TYPE=FALSE VDU 7:ENDPROC
1740PRINTTAB(0,22)SPC40
1750INPUTTAB(0,20)"Type file name  "$AX
1760X=OPENOUT($AX)
1770FOR I=0 TO J:BPUT£X,A?I:NEXT
1780CLOSE£X:ENDPROC
1790DEFPROCscore:X%=POS:Y%=VPOS
1800VDU28,0,24,39,24:PRINTSPC(40)
1810PRINTTAB(27)CHR$131"CHRS ";JA%;
1820IF R$="6" PRINTTAB(0,0)CHR$130"DEL; CTRL-I";CHR$131"
    LINES ";R%;
1830VDU28,0,22,39,0
1840VDU31,X%,Y%
1850ENDPROC
1860DEFPROCANS:REPEAT G$=GET$
1870UNTIL G$="Y" OR G$="y" OR G$="N" OR G$="n"
1880ENDPROC
>
```

ROBOTICS

Bill Horne

We explore some primary considerations for a practical working, robot.



What is a robot? The word first appeared in the play 'R.U.R' by Capek, the letters being an abbreviation for 'Rossum's Universal Robots'. To make the play practicable, the robots were represented as mechanical simulations of human beings, and Isaac Azimov has continued this concept, but in real life robots need not be humanoid in concept or form.

Nevertheless, it would be foolish to ignore the human model completely. If we want a robot to deputise for a human being, we may need to give it a number of human characteristics, while avoiding any human features that are not essential.

Consider, for instance, the paint-spraying robot, which will learn a movement pattern in one lesson, and then repeat the pattern again and again. All the necessary actions can be performed by a flexibly-jointed arm and a 'finger' on the trigger of the spray gun. It would be a mistake to add mobility, since the system needs to have a fixed point of reference, and it would be easier to move the articles to be painted on a conveyor of some sort.

Apart from the flexible arm, the system will need means for sensing the position of the joints. During the learning phase, this data will be stored for reference. During the working phase it will provide information to be compared with the stored data to determine what movements are required.

This illustrates a key point about robot systems, the use of feedback. It is possible to build simple systems which need no feedback, as such. A printer can position the paper and typehead with extreme precision by driving stepper motors through a given number of quarter-revolutions. However, it is common to set the base reference position when the print head is at the end of its travel to the left, so cancelling out any errors which may have crept in.

The printer needs computing facilities to keep track of position and generate the necessary drive pulses, but some early robot systems managed without such assistance. The brake servo system of a car uses positional feedback to control the force generated by pneumatic pressure differential, and this is done on an entirely mechanical basis. Such a system might be seen as an elementary robot, but one which is rather outside our range of interest here.

It may appear that mobile robots, of the kind that scamper across the living-room floor or through the intricacies of a maze, are also remote from serious robotics, but we must not forget systems in which fork-lift trucks find their way about a store room, stacking and collecting specified blocks of goods.

MOTIVE FORCE

The essential feature of a robot is that it can generate controlled mechanical force. In the broadest view, both the necessary power and the means by which the power is controlled are likely to be electrical, but it may not be convenient to use the power directly in that form.

An electric motor is inherently a high-speed, low-torque device. As it speeds up, its armature stores kinetic energy, and that has to be dissipated when the motor is stopped. Disconnect power, and the motor will run on until the energy has been used up. If a short-circuit is applied, the run-down will be faster, because the wastage is augmented by electrical effects, but the delay may still be inconvenient. To make the system stop in the right place, it may be necessary to use second-order feedback, the speed of the motor being taken into account.

An example of this arose some twenty years ago. The problem was to drive a fairly massive metal drum supporting a circle of type-slugs, positioning the drum with a given slug at the operating position. final positioning was ensured by a detent, but the slug needed to be within a degree or two of the precise location for the detent to work.

The existing system used a constantly running electric motor driving through clutches, more clutches being used as brakes. This worked, but was noisy and needed a fair amount of maintainance. It was replaced by a beefy ⅓ horse-power electric motor driving the drum by a toothed belt. A digital sensor provided information on the drum position, and its output was compared on a subtractive basis with the required position. While the error exceeded 16, the motor was powered to drive the drum in the required direction, but for smaller errors the drive was reduced, or - if the speed was too high - reversed so that the zero error position was approached as speed fell to zero. The system was very quiet, and wear and tear were minimal.

This, however, is a special case, and in other cases it is more convenient to use hydraulic power generated by an electrically-driven pump. Hydraulic power is especially suitable for linear movement, using rams and electrically-controlled piston valves. If the valves are correctly designed, the electrical driving force needed can be quite small.

To be strictly correct, such a system should be called hydro-pneumatic, because it may well rely on air pressure as the effective driving force. Hydraulic fluid being incompressible, it cannot store energy. The solution is to use a 'reservoir', which is filled with air. As the hydraulic pressure builds up, fluid is forced intor the reservoir, compressing the air. Any reduction of pressure allows the air to expand, forcing the liquid out again.

There are also 'live' hydraulic systems, in which the fluid can bypass the ram when no force is required.

For rotary systems, hydraulic power is less convenient, though we must not forget the swash-plate motor pioneered by DOWTY. This used a diagonal plate on the driven shaft, with a ring of hydraulic rams pressing on the plate in turn and forcing it to rotate.

Hydraulic devices call for high-precision manufacture, and carry the remote risk of a burst pipe and consequent mess. Such a failure is bad enough in a factory test room in the presence of very important visitors (an abiding memory of such an incident lingers on!) but could cause even more trouble in domestic surroundings.

Pneumatic systems avoid these disadvantages, but have disadvantages of their own. Air-driven motors share the over-run problem of electric motors, and are not always easy to reverse, while control of pneumatic power is not always precise. Anyone interested in this kind of power would do well to investigate a pianola or pipe-organ, both of which use pneumatic power extensively.

Returning to electrical power, reference has been made to the use of stepper motors in printers. These can generate relatively high torque at low speed, and can be controlled very precisely by trains of pulses. However, for best results it is necessary to generate pulses of a special shape, with an initial high transient to get the motor moving and then a lower level to hold the motor in the new position.

There was a time in electronics when everybody involved felt that the actual circuitry was comparatively simple. The difficulties arose at the interface between the electronics and the associated mechanical devices. The same situation remains true for robotic systems, with the added problems of the hardware/software interface...

SENSORS

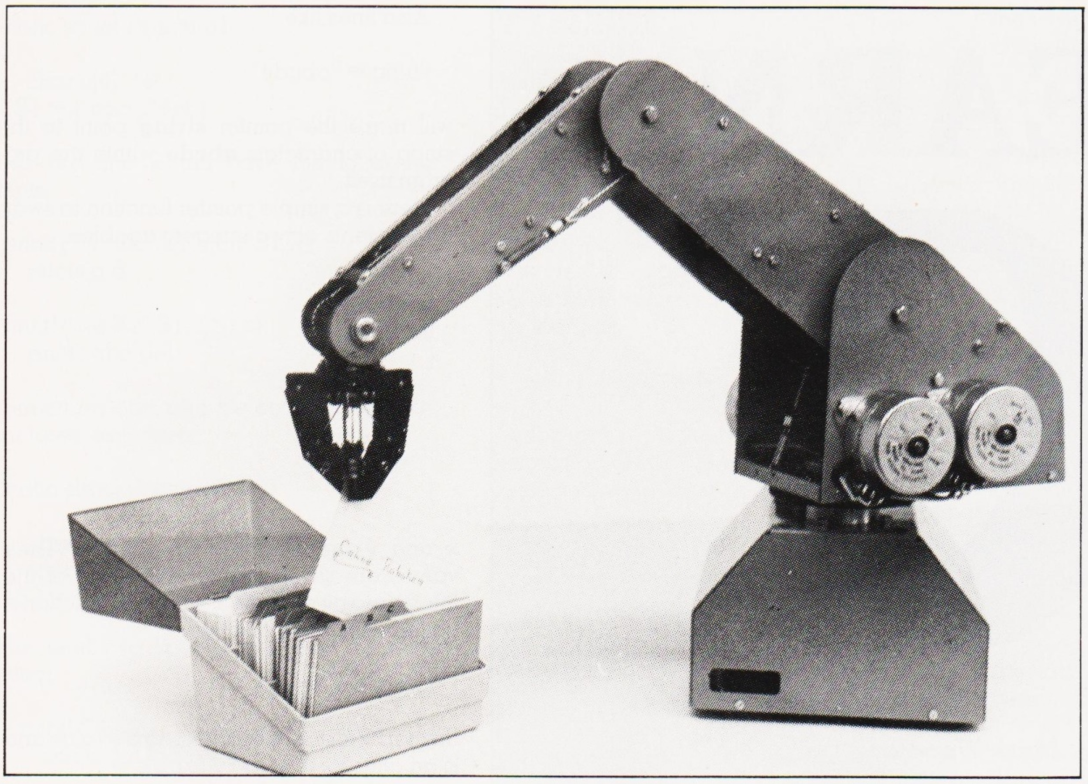
If a multiple-jointed arm is mounted on a turntable, sensing the position and attitude of the end of the arm can be quite a problem. The angle of each joint, plus the orientation of the turntable, must be measured and the results fed into a complex calculation.

Sensing angular position digitally is often favoured on the grounds that the measurement is precise and free from drift, but digital sensors working on a binary basis have one key defect: since more than one bit may change on moving from one position to the next, it is necessary to ensure that all bits change at the same moment. Otherwise, a change from, say, 0111 to 1000 might momentarily give 1111,0000, or a number of values between.

In some cases, it is possible to generate a clock pulse when the binary output is valid, and this used in punched paper tape readers. Another solution is the use of Gray Code.

Gray Code is a modified form of binary in which only one bit changes as you move from one number to the next. For example;

BINARY	GRAY CODE
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0110



ULTRASONICS

In some situations, ultrasonics provide a substitute for vision. An academic project a few years back used ultrasonics to trace the movement of a referee round a football field. *(If anyone who was involved cares to write about that, we'd be interested.)* This rather odd application illustrates the possibilities, though not directly in the robotics field. field.

Ultrasonics can also be used to transmit commands and data, or to receive sensor outputs. "Cordless" keyboards and the like come into this category.

However, such systems, like radio links, can be subject to interference from similar systems, and it would be disconcerting if a robot went haywire because the cat stepped on the buttons of a TV control unit. (Our cat seems to like doing this!)

The conversion rule may be stated thus: if a bit in the binary form is 0, the bit to its right is the same in binary and Gray Code. If the bit is 1, the bit to the right is complemented in Gray Code.

This presents no serious problems in converting from binary to Gray Code, but it is the reverse conversion that is more commonly needed, and that has to proceed on a 'ripple-through' basis, since each binary digit from the left must be determined before the next bit can be calculated. During this process, invalid numbers a can appear. For example, translating 0100 in Grey Code involves the steps;

0100
0110
0111

The uncertainty during a numeric change is still present, but it is now limited to a short period of time, rather than a small range of movement. The effect can be overcome by treating the number as valid only if it remains unaltered for a time that is longer than the maximum ripple-through time.

Angular digitisers are usually more convenient than linear types, which can be awkward to mount and which may take up an inconvenient amount of space. A common arrangement converts the linear action of a ram into rotational movement, so angular movement is commonly measurable.

In some cases, it may be preferable to work on a coordinate system, but here again the linear movement may well be generated by rotation, or can be easily converted into rotation.

An entirely different form of sensing is needed where the force applied needs to be controlled. A pair of jaws used to pick things up must close just far enough to grasp the objects, but not firmly enough to crush them. Pads of conductive plastic foam, which show reduced resistance under pressure, can be used to sense the force exerted by the jaws. However, even this may not be delicate enough to ensure equal pressure by both jaws, and without that there is a risk that the item being picked up may topple over...

VISION SIMULATION

The simulation of sight is an extraordinarily difficult subject. One reason is that we use memory a great deal in interpreting what we see, and this allows us to recognise objects on a basis of minimal data.

Put a small box on the table, and consider what you might see: A pattern of adjacent rhomboids, the visible sides of the box, and perhaps some extra rhomboids representing shadows. You can see the direction in which the box lies, and binocular vision gives you a fairly accurate idea of its distance. Armed with this data, you can reach out and pick up the box with confidence and precision. Note that you aim first for a point above the box, so that you don't brush it aside before your fingers straddle it, then lower your hand and close your fingers.

To simulate this in a robot would be quite a problem. A single photo sensor would not be enough. Your simulation needs to have a field of view, so that it can locate the box and detect the outlines of it. These will appear, after processing, as a set of co-ordinates which must be interpreted as representing a box-like shape. That may call for reference to stored data.

Next, you want binocular vision to detect distance. But be wary. If two similar boxes were in the field of view, the system might go cross-eyed.

You now have the data needed to send out the gripping arm. As it nears its objective, the arm may very well interfere with the view of the box, and the system must allow for this. Reaching the right position, close the jaws gently and lift...

Incredibly, vision-controlled robots do exist, doing serious work like putting windows into car bodies...

BACK TO THE QUESTION

We can now return to the original question in slightly extended form; What, in practical terms, is a robot?

It would be possible to include any servomechanism, but that would rather stretch the definition. Essentially, a robot must respond to its environment, not just to arbitrary commands.

In other words, the robot must display a measure of independence, extrapolating from a simple control input on the basis of what data it can pick up for itself. The parallel with artificial intelligence is clear, but I've had my say on that already.

Automatic housework and the like seem to me to be a pipe-dream. Cylindrical 'pigs' scuttling through underground pipes looking for side branches are practical, but only because they work in a fairly standardised environment, whereas no two houses are alike, and their contents move round from place to place at frequent intervals. Most household equipment is designed for human use, so unless the design was changed, it would be necesary to make the robots humanoid.

Personally, I hate housework of any kind, but I doubt whether I would ever come to terms with a robot housekeeper scuttling round the place. It might decide that I was part of the domestic trash...

LEARN 'C'

Mark Woodley

Advanced features of the language.

This article will look at the advanced features of the language C and show how and why they make the language so powerful.

There have been two other articles in this series; the first dealt with the format of C programs, variable declarations, simple types and the unique concept of storage classes; the second dealt with the commands that make up a complete C program, and the expressions used in commands and assignments to variables, and introduced arrays and strings.

It would obviously be useful to have read the other articles before this article; however, we hope that this article can, on its own, give the reader a good idea why C is rapidly becoming so popular.

POINTERS

A pointer in C is a variable whose value is the address of another variable. We can declare a pointer **p** which points to a character ***p** in memory, with the declaration

```
char *p;
```

It is necessary to specify the type that a variable points to so that pointer expressions are evaluated correctly. As an example of this a pointer might be used to reach characters from a buffer. A character may be taken from the buffer with

```
c = *(p++);
```

which makes **c** equal to the character (as previously declared) 'pointed at' by (p++). Which gives the character at **p** and increments **p** to the next position, in this case one character ahead.

The asterisk (pointed at) operator is the exact opposite of the ampersand (address of) operator. So **p=&*p**!

So to summarise,

```
*p = n
```

means, load **p**'s variable with **n** (not p's contents), and,

```
p = n
```

means, load pointer **p** with the value in **n**.

As you might expect, pointers and arrays (which are similar interpretations of the same concept) can be freely interchanged. For example:

```
{char string[10], *p;
p = string; /* p points to start of string */
char c0, c9;
c0 = *p; /* c0 = c[0] */
c9 = *(p+9) /* c9 = c[9] */
}
```

With pointers however, the user has to program the array transformations himself. This only becomes feasible when handling one dimensional arrays, strings and buffers.

Be careful of assignments like:

```
string = p;
```

which would change the base address of the string and may have spurious consequences.

Also lines like

```
string = "abcde"
```

will make the pointer **string** point to the string of characters **abcde** within the program itself.

Here is a simple pointer function to swap the contents of two integer variables.

```
swap (x, y)
int *x,*y;
{
   int temp;
   temp = *x;
   *x = *y;
   *y = temp;
}
```

x and **y** are declared as pointers. Their values are given as the addresses of the integers to be swapped, when the function is called with a line such as:

```
swap (&a, &b);
```

So that ***x** is an integer holding **a**'s value and is an alias for **a**, and ***y** likewise.

STRUCTURES

Structures in C are equivalent to records in other languages. Here is a declaration which creates a structured variable **someone**, to hold **name**, **age**, and **weight**.

```
struct /* a structure with identifier someone */
  {
    char name[30];
    int age;
    float weight;
  } someone;
```

Each field can be accessed with the field selection operator, which is a period (.). For example:

```
someone.age = 21;
```

We can create a structure as a type. The user defined type's name is specified after the reserved word **struct**.

```
struct person /* a structure of a type person*/
  {
    char name[30];
    int age;
    float weight;
  };
```

Now we can declare variables of that type, by:

```
struct person someone;
```

which has the same effect as the first declaration of someone.

When a structure is declared, it can be given initial values by listing them between braces. For example:

```
static struct structure1
{
   char c[4], *s;
} S1 = {"abc","def"}
```

these two line would have the following effects,

```
printf ("%c %c",S1.c[0],*S1.S);
   prints a d

printf ("%s %s",S1.c,S1.s);
   prints abc def
```

Structures can also be contained within structures, as follows:

```
static struct structure2
{
   char *cp;
   struct structure1 SS1
} S2 = {"ghi", {"jkl", "mno"}};
```

The next two lines would then have this effect,

```
printf ("%s %s", S2.cp, S2.SS1.s);
   prints ghi mno

printf ("%s %s", ++S2.cp, ++S2.SS1.s);
   prints hi mn
```

Structures are passed to functions via their base address, in much the same fashion as with arrays. Corresponding declarations must however be provided for, in the receiving function. The ampersand and the asterisk can be used for this purpose.

For accessing particular fields in a structure which has a pointer indicating its base address, C provides a special operator; the characters −>. If **ptr** is a pointer (declared as pointing to a particular structure) and there is an **age** field in that structure, then

```
ptr −> age
```

represents the contents of the age field or ***(ptr.age)**.

UNIONS

In Pascal, when the programmer creates a record, it is possible to declare its fields as 'variant'. The compiler would then reserve an amount of space large enough to hold the largest of the optional fields.

In C, this is also possible. A float variable cannot be assigned to (stored in) a variable declared as an int (and anything might happen if you try!), casting may lose valuable accuracy. So the union type is provided;

```
union number
{
   int i;
   float f;
} x;
```

This declares a new type **number** which is large enough to hold either an *int*eger or a *float* variable (but not both). The programmer must then keep track of the actual value using either **x.i** or **x.f**

TYPEDEF

To declare any other user defined type, the C programmer used the reserved word **typedef** in declarations. For example

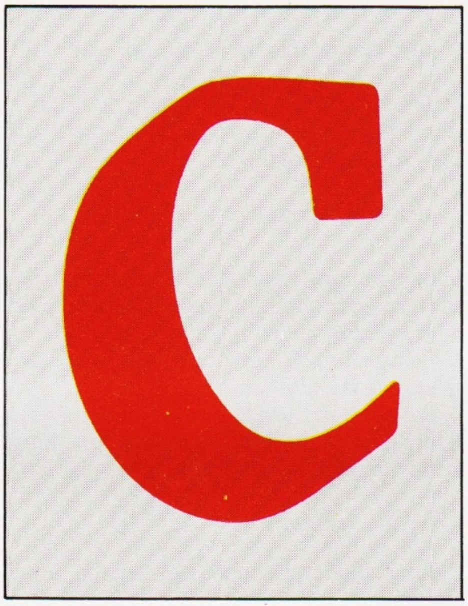
```
typedef int integer;
```

makes the word **integer** an alias for the reserved word **int** in variable declarations. More complicated types can be formed, for example:

```
typedef float *ptf;
```

which makes **ptf** the type identifier for pointers to float variables. **a**, **b**, & **c** could then be declared as such pointers with

```
ptf a, b, c;
```



STORAGE ALLOCATION

In C, storage allocation is the responsibility of the programmer. To determine how much space is occupied by a variable of a particular type, the **sizeof** function is used, in one of the forms,

```
sizeof (object)
```

or

```
sizeof (type)
```

To allocate **n** contiguous bytes of store and return the address of the first three, the **malloc** (make allocation for) function is used:

```
malloc (n)
```

To return storage used up by an area of store **free(n)** can be used where **n** is an address found with **malloc**.

THE C PREPROCESSOR

The C compiler automatically invokes a preprocessor when a program is compiled. The preprocessor interprets lines beginning with the hash (#, sometimes printed as £) character.

```
# define identifier token
```

Throughout the remainder of the file, each 'identifier' is replaced by a 'token', the token can be anything: identifier, keyword, constant, string, operator or even punctuation! it can also have abstract forms to produce macros. For example

```
#define max(a,b) ((a)>(b)?(a):(b))
```

Another of the preprocessor commands is #include,

```
#include filename
```

At the point of each #include in the program the contents of 'filename' will be included in the compilation. This has tremendous advantages, particularly in the selection of common macros. For example, as we will see later, the standard I/O macros in the header file "stdio.h".

Standard functions are not provided by the C compiler, instead libraries of functions have to be made available to your computer system at compile time. This simplifies the design of the compiler and allows more flexibility in the use of standard functions. For example, users can write their own libraries.

One library that must always be present, because it contains C's I/O functions, is the Standard Input/Output Library. This is used in conjunction with a file of macros called the Standard Input/Output Header file "stdio.h", which was mentioned in the first article.

C treats peripherals and files alike, as **streams**. When a C program is invoked, there will usually be three streams available. These are **stdin**, the standard input device (usually a keyboard), **stdout**, the standard output device (for example a screen) and **stderr**, the file or device where error messages are sent. Stdin, stdout and stderr are constants that point to locations in memory where there are tables of data for each stream. These tables are declared with type name **FILE**.

For example, "stdio.h" might begin,

```
# define _BUFSIZE 150
# define _NFILE 20
typedef
struct _iobuf
{char *  _ptr; /* pointer to current
                 character in buffer */
 int _cnt; /* number of bytes left in buffer
                 */
 char* _base; /* pointer to start of buffer
                 */
 int _flag; /* bits set for read, write, error,
                 eof etc. */
 int _fd; /* file descriptor - the link with
                 the Op Sys */
   char _buf [BUFSIZE]; /* io buffer */
} FILE;

extern FILE _iob[_NFILE];
define stdin (&(_iob[0]))
define stdout (&(_iob[1]))
```

define stderr (&(__iob[2]))

The FILE structure, of which twenty are allocated in this implementation, produces a link with the operating system, a set of status flags, and an I/O buffer with associated variables. The structure is available for use by C programmers.

Underscores are used to prevent a clash between any user declared variables and those declared as part of the system.

Some other constants that will be defined as part of the header library are;

EOF The value returned by I/O functions at end-of-file.

NULL The value used to indicate a NULL pointer, it is returned by pointer functions to indicate an error.

I/O FUNCTIONS

Here is a list of some of C's standard input/output functions.

int fprintf (format string, arguments);
int fprintf (stream, format string, arguments);
int sprintf (stream, format string, arguments);

All three functions work in the same way as **printf**, except that **fprintf** sends its outputs to the specified stream and **sprintf** sends its outputs to the specified string.

int scanf (format string, arguments);
int fscanf (stream, format string, arguments);
int sscanf (string, format string, arguments);

All three functions work in the same way as **scanf**, except that **fscanf** reads from the specified stream and **sscanf** reads from the specified string.

int fputc (c, stream);
int fputs (s, stream);

fputc sends the single character in **c** to the specified stream, and **fputs** sends a string **s** to the specified stream.

int fgets (c, stream);
int fgets (s, stream);
int ungetc (c, stream);

fgetc fetches the character **c** from the specified stream, and **fgets** gets a string **s** from the specified stream. **ungetc** pushes the character **c** back onto the stream, so that a subsequent input will pick it up.

FILE *fopen (filename, mode);

Used to open a file for "w" writing, "r" reading and "a" for appending; returns a pointer to the FILE table for the stream.

int fclose (stream);

fclose empties any buffered output from the stream, closing the file making its associated FILE structure available for reuse.

Remember, all these functions return EOF at end-of-file or when an error occurs.

In "stdio.h", the following macros are defined in terms of C standard functions. They cannot be pointed to.

int getc (stream)

Returns the next character on the input stream.

int getchar (stream)

is actually **getc (stdin)**.

int putc (c, stream)

Writes the character **c** to the output stream returns **c** or EOF error.

int putchar (c)

Same as **putchar (c, stdout)**.

int ferror (stream)

Returns nonzero when the error flag is set. The error flag remains set until the stream has been closed.

Here is a list of C's reserved words:

auto	if
break	int
case	long
char	register
continue	return
default	short
do	sizeof
double	static
else	struct
entry	typedef
enum	switch
extern	union
float	unsigned
for	void
goto	while

PROBLEM

Here is the solution to the problem that I set last month. (note that # appears as £ in this printout).

```
£include  <stdio.h>
£include  <ctype.h>
£define   STOP       "STOP"
£define   EOT        "ZZZZ"
£define   BUFFSIZE   50
main()
{
      int TelNo;
      for (TelNo=1; Telegram(TelNo); TelNo++)
           ;
}
£define   STOPWORD            (strcmp (word, STOP) == 0)
£define   SINGLEUNITLIMIT     12
int Telegram(TelNo)
int            TelNo;
/* Prints a telegram and returns zero when it is null */
{
      char word[BUFFSIZE];
      int  len, Singles=0, Doubles=0;
      Capitalise();
      if (len = Getword (word, BUFFSIZE))
      {
           printf ("\n\n\n");
           printf ("Telegram number %d.\n\n", TelNo++);
           while (len != 0)
           {
```

```
                if (STOPWORD)
                {
                        putchar ('.');
                        Capitalise();
                }
                else
                {
                        PutWord (word, len);
                        (len > SingleUnitLimit)  ? Doubles++  :  Singles++;
                }
                len = GetWord (word, BUFFSIZE);
        }
        PutWord (word, 0);   /* implicitly resets pos */
        printf ("\n\n");
        printf ("Charge: %3d words at 1 unit%5d units\n",Singles,Singles);
        printf ("%11 words at 2 units%4d units\n",      Doubles,2*Doubles);
        printf ("                        TOTAL%7d units\n",Singles+2*Doubles);
        return (1);
    }
    return (0);
}
    PutWord (word, len)
    char      word[];
    int            len;
    /* Simple function to format text output. */
    {
        static    int col;
        if (len == 0)
                col = 0;
        else
        {               /* increment column position and wraparound */
                col += ++len;
                if (col > BUFFSIZE)
                {
                        col = len;
                        print ("\n");
                }
                printf (" %s", word);
        }
    }
    £define WHITESPACE(C)  (C == ' ' || C == '\t')
    int GetWord (word, size)
    char          word[];
    int                size;
    /* Simple function to read a single word string */
    {
        int c,i;
        SkipSpaces();
        for (i=0; c=getchar(); !WHITESPACE(c); )
                if (c != '\n' && i < size)
                        word[i++] = c;
        word[i] = '\0';
        return (strcmp(word, EOT) ? i : 0);
    }
    void Capitalise()
    /* Capitalises the first letter in the next word to be read */
    {
        int c;
        SkipSpaces();
        c = getchar();
        ungetc ('a' <= c && c <= 'z' ? toupper(c) : c, stdin);
    }
    void SkipSpaces()
    /* leaves file pointer at next non-space character */
    {
        int c;
        while (isspace (c = getchar())
                ;
        ungetc (c, stdin);
    }
```

POLAND

Dr George Blazyca, Lecturer in Economics, Thames Polytechnic, London.

Earlier in the year Sinclair Research announced plans to step up exports to eastern European countries. Here, we describe the struggle of Polish citizens to achieve computing parity with the west.

Not much is heard about personal computing in Eastern Europe for the simple reason that not much exists. But in Poland, at least, things may soon change. The ministry of education has been given until September to produce a plan for the development of computing in schools, and, in January the high-brow weekly, *Polityka* launched its own discussion on the future of Polish personal computing.

The timing is crucial: later this year the five year plan for 1986-90 will be settled and personal computing, if it is to be developed, will have to be in that plan.

Why is personal computing so weakly developed in Eastern Europe? Part of the problem is that centrally planned economies have difficulty in making room for entrepreneurs. The Clive Sinclairs of Eastern Europe stand no chance of running anything much larger than garden shed sized enterprises. Under central planning the red tape is redder than anywhere else and the incentives to innovate are much weaker.

Another problem is the poor state of the components industry. Intel produced its famous 8080 microprocessor in 1974 but a Polish 'lookalike' appeared only in 1982. Even now the production run is just 10,000 per annum, 'a symbolic quantity' according to one participant in *Polityka's* debate.

There *are* computer producers in Poland but they tend to be more concerned with the large orders regularly received from the Soviet Union and have shown little interest in micro computing. A Polish personal computer, the Meritum-1, did appear in 1982 but it was expensive, emerged in the usual 'symbolic quantities' and to compare it to a Sinclair machine was, according to one Polish writer, like comparing a Syrena (a 1950's car) to a Fiat Panda.

BEST KNOWN OF MICROS

The best-known micros in Poland are Sinclair products. They figure prominently in the small ads of the Warsaw daily newspaper, *Zycie Warszawy*. A recent estimate put the number of Sinclair computers in Poland at 20,000. Most are privately owned but recently the gap in the market for micros has been approached (filled would be too strong a word) by a number of small private firms which have taken up the business of packaging Sinclair systems for institutional customers like the ministry of education. Those firms are vitally interested in persuading the ministry to go for a Sinclair based solution to its computing problems. One businessman has gone to the extent of delivering 300 free Sinclairs to schools to give them a hands-on experience which, it is hoped, the ministry will take careful note of.

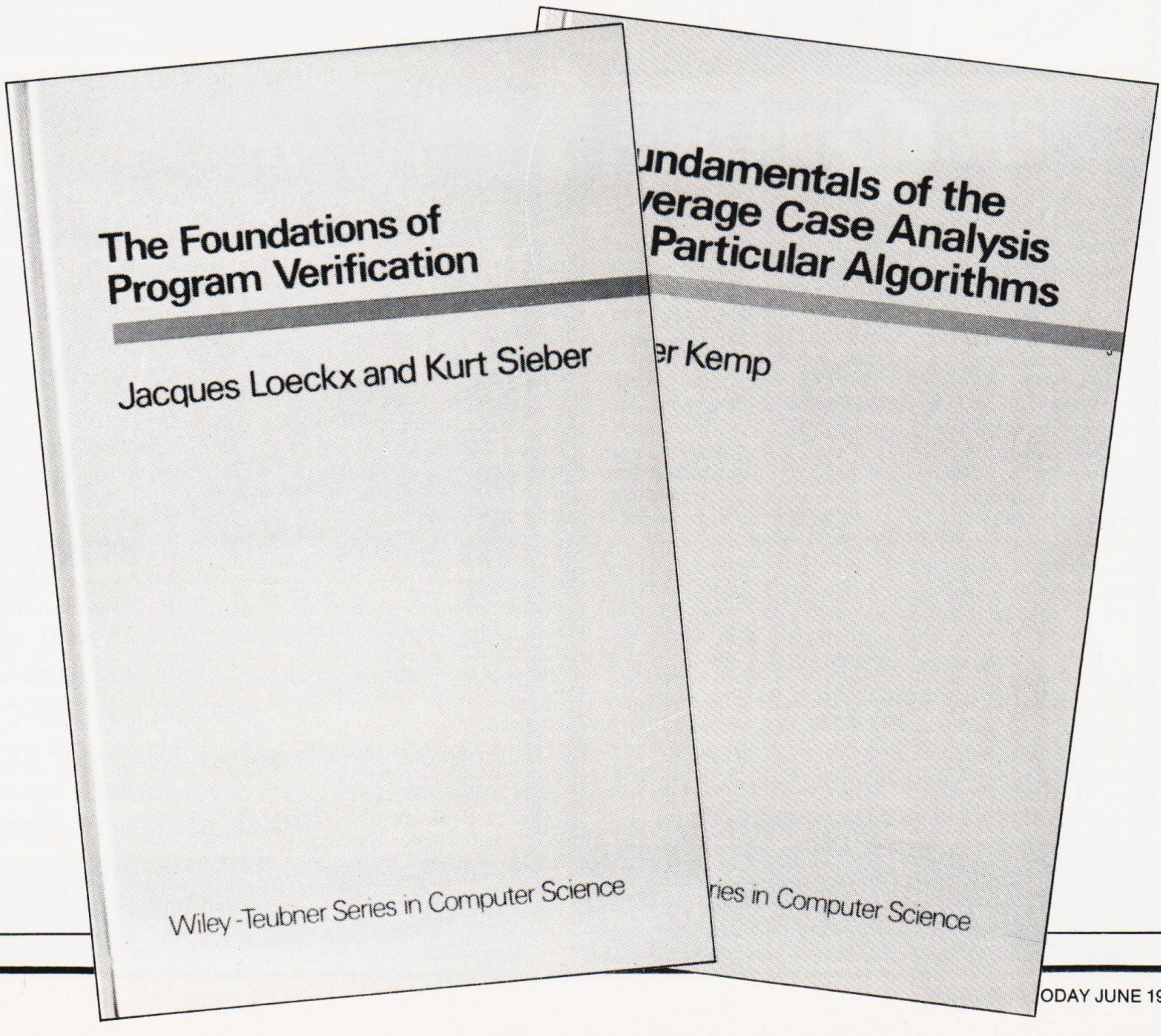
What is likely to happen? The Polish industry has promised to update its own product and produce a 'Meritum 1.1' with 32K of memory but the ministry of education is unlikely to base its plans on promises alone. Sinclair computers probably do impress the men from the ministry but they aware of the country's chronic shortage of foreign exchange which is likely to rule out an imported solution to the problem. In this context some Polish experts point to an ingenious half way solution: do-it-yourself.

They argue that money should be put into developing the domestic components industry and this should be supplemented by imports from the West. Microprocessors worth a few US dollars are after all much cheaper than computers. With a reliable supply of components more people organised in clubs, for example through the socialist youth organisation, could be persuaded to get involved in 'self-assembly'. The country could become computerised (and incidentally the young people kept off the street corner) both quickly and cheaply. Since this is just about the only way Poland can afford to get into mass microcomputing it's the solution worth watching.

AN ACADEMIC APPROACH

Bill Horne

An enormous gulf separates the theoretical and practical worlds of computing. Where academics demand lengthy proofs for system correctness, programmers and engineers would often use a more intuitive approach. Meanwhile, we straddle the great divide...



Among the books which deluge into the Computing Today office, two recent examples stood out as being totally different from the rest. One was called "The Foundations of Program Verification", the other "Fundamentals of the Average Case Analysis of Particular Algorithms." Both were packed with abstruse mathematical formulae, and the first book referred in the first few lines of the preface to "the inductive assertions method of Floyd", and "the axiomatic method of Hoare and Scott's fixpoint induction."

The average reaction to these books was to open them, scan a few lines, and put them away again with eyes raised heavenwards, but it occurred to me that this was dodging the issue. To me, the books illustrated the vast gulf which lies between their kind of world and the world of everyday software.

Consider the process of 'program verification' which is likely to be used in practice. The target is a program which performs precisely to specification, and is preferably idiot-proof. Some companies employ a tame idiot to help with the testing.

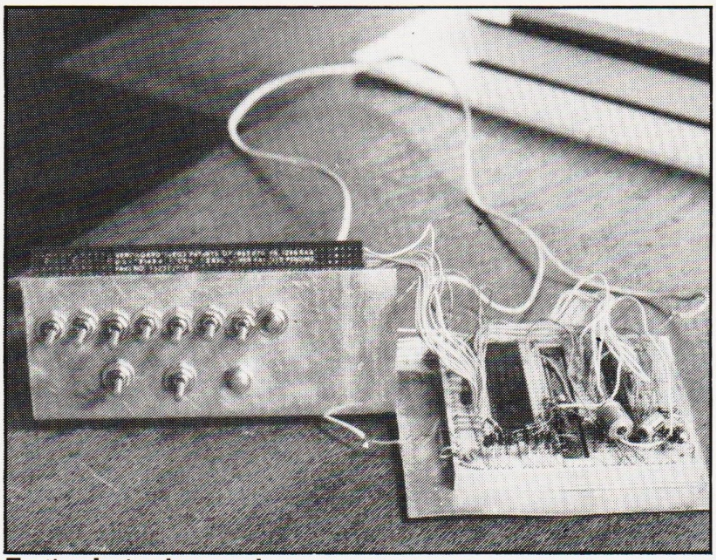
NOTIONS OF CORRECTNESS

The first book devoted five chapters to laying the mathematical groundwork, and in the sixth chapter dealt with 'the notion of the correctness of a program', as if it were a variable parameter. It is not. Either a program works correctly or it is bugged. Is that too simplistic a view?

All this took my mind back to a fairly simple central processor which I designed some years ago. When it was built and running, I was asked to devise test routines which would demonstrate that the thing would work correctly with all possible data values. Nothing less would be acceptable as proof of the validity of the design.

It was soon evident that there were problems. Generating all the possible values for a single variables would take over an hour, and adding a third variable would extend the time to a thousand days!

To provide a practical test, I postulated that the logic system, as designed, could be shown by careful analysis to work with



Testing 1: simple enough

all possible data combinations. This, in itself, was a somewhat arduous task, but in the end the analysis was accepted. It was then necessary to devise a test routine that would demonstrate that the actual hardware did all the things it was supposed to do, which was possible by using selected variable values, rather than all possible combinations.

That approach was eventually accepted, partly because anthing more comprehensive was demonstrably impractical. Fortunately, those who had to be convinced were unaware of the possible existence of the kind of timing error that is sometimes called a 'race hazard', and made no comment on the careful study of the system timings which gave assurance that such problems would not arise with in-spec components.

In the end, the system ran for some years without serious hardware trouble. The simplified test routine proved able to detect faulty components, and once these had been replaced the processor did its job perfectly.

The solution adopted was, I submit, a practical solution serving adequately in place of an impractical rigid academic approach, and the principle can equally well be applied to software, given that the possibility of errors is properly understood.

A particular case will illustrate the point. A multiplying process appeared entirely satisfactory until it was noticed that a graphics program was behaving rather oddly. It then emerged that where the product ended in a zero there was a failure to pass a carry, and a wrong result was generated.

The bug was found by chance, but there should have been no need for that. a rigorous analysis of the routine concerned should have shown the problem in time to allow correction. It is very likely that the most comprehensive practicable tests would have shown up the bug, because it appeared in a restricted area.

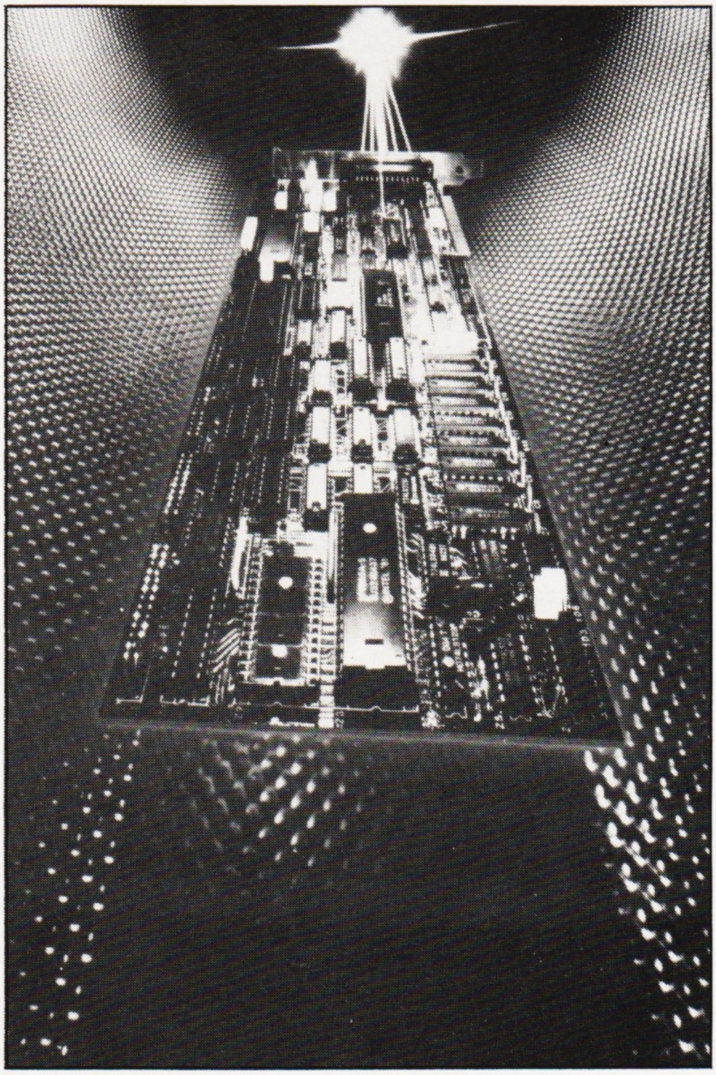
What really happens in software checking? Detection of the existence of a bug is usually the easiest part. Tracing it to its lair may be much more difficult. Perhaps an 'anomalous' two's complement number has arisen. (A'1' followed by all 0s, which remains negative when negated.) The best approach is usually to hunt through the printout looking for sillies. Once you suspect that they exist, they are much easier to find...

Paradoxically, a large program producing a lot of data may be easier to test than a comparatively simple routine, since there may be an obvious inconsistency in a set of output values, but the size of the program makes debugging that much more difficult.

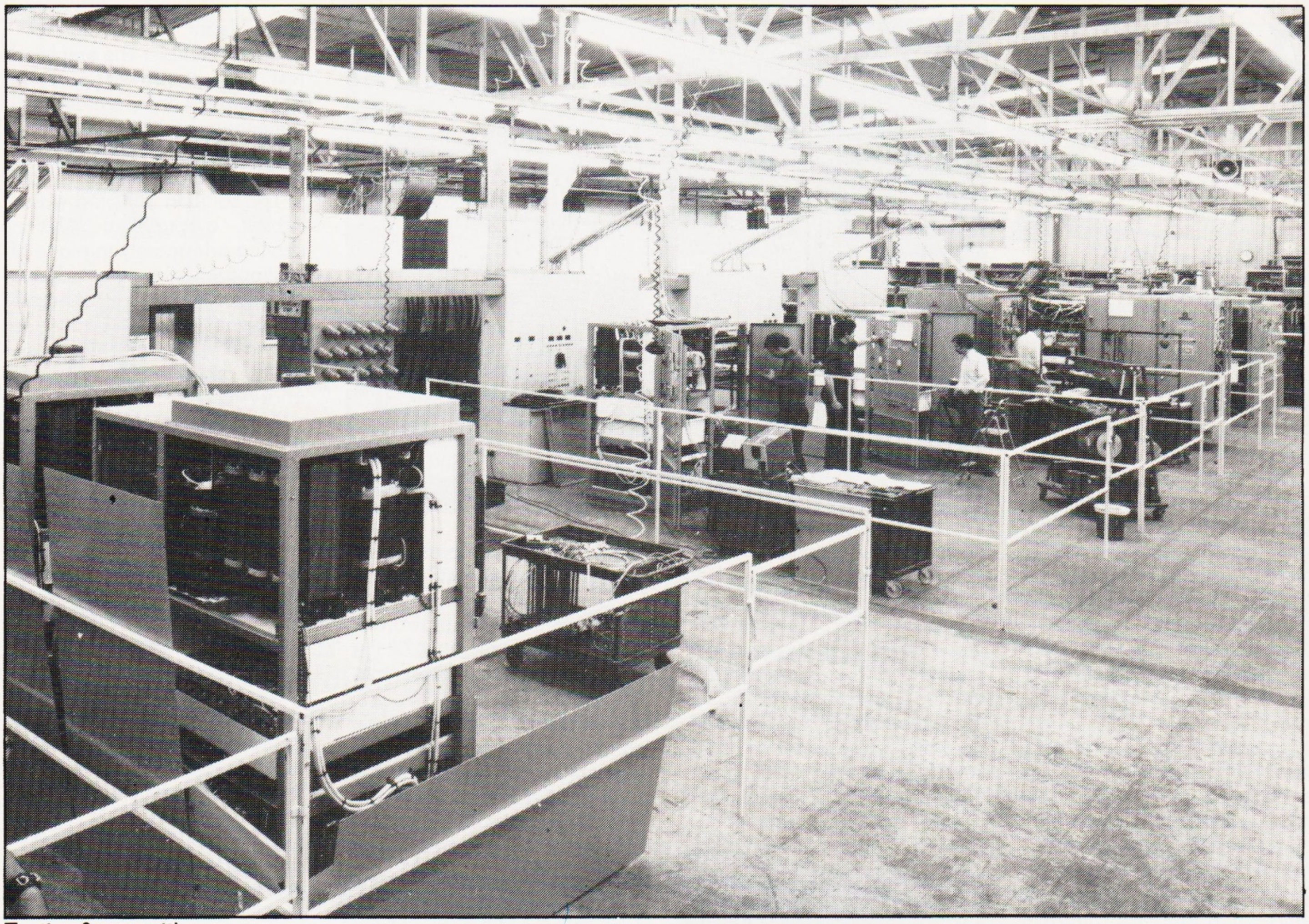
COAL FACE WORK

How could abstruce mathematical studies help in this process? It is coal-face work, requiring a keen eye and considerable mental stamina, and is often carried out under pressure. I can remember circumstances in which the intrusion of a theorist might well have led to murder.

So the first book may delight someone who lives in the clouds



Testing 2: getting tougher



Testing 3: prove it!

high above the hurly-burly of practical programming, but I doubt whether it would be in great demand among down-to-earth software people.

The second book begins, almost immediately, by introducing a 'Turing Machine', which is not clearly defined, as a basis for working out the time required to execute a particular algorithm. It would have been equally possible, and much more directly applicable to everyday work, to use;

```
L1    LD     HL,TIME
L2    INC    (HL)
      INC    HL
      JR     Z,L2
```

In the book, a binary number is incremented, bit by bit, a process relevant to the outmoded concept of a 'Turing Machine', but not at all relevant to modern addition processes, which use parallel action and look-ahead carry. The small routine given above increments a number held in several successive store locations, and is far more relevant to current practice,

The objective of the book is to provide a basis for selecting one algorithm rather than another, critical factors being running time and required storage space. In the practical world, algorithm selection is often based on intuition, experience, or perhaps on the results of comparative testing.

As with the first book, the subject and its treatment are set at an elevated level far removed from every-day programming. The assumption is made that the subject is amenable to exact treatment, which is not always the case. A particular algorithm, for example, can be shown to require $2*N*LOG_2N$ transfers to handle N items, but the calculation fails to reveal that a particular machine adds a lengthy garbage collection process which makes nonsense of the theoretical figures.

It should also be said - and here old friends of mine would be justified in having a quite chuckle - that in many systems a central executive or interrupt handler may erode the available machine time to such a degree that time-efficient applications programs are of little advantage. There was an ancient computer which brought this home with some force. It used to sing to itself, and its more experienced handlers could recognise from the tune when the executive was in action. It seemed to be in action for rather a large proportion of the total running time...

□ □ □

THE OBJECT here has not been to decry the two books mentioned at the start, but to highlight the differences between the world to which they belong and the world of practicality. There is a dilemma here. While intuitive methods can take us a long way in the improvement of algorithm efficiency and program testing, a point is reached where the matter becomes too complex to handle in this way, and confusion sets in. The books propose a way of proceeding further on a mathematical basis, but the methods they propose would only be accessible to a limited number of people. However, the books do provide a glimpse of a remote academic world far removed from ours. Of the two, that dealing with algorithm analysis is closer to familiar ground and may have a wider appeal, but I fear that I will have to go on relying on intuition.

The Foundations of Program Verification, Jaques Loeckx and Kurt Sieber (John Wiley & Sons, Chichester)

Fundamentals of the Average Case Analysis of Particular Algorithms, Rainer Kemp (John Wiley & Sons, Chichester).
(Both are of German origin, and are also published by B.G. Teubner, Stuttgart.

ALGORITHM ANGLES

Geoffrey Childs

Coming to grips with some common stats functions.

Most mathematical tables can be constructed with ease if you own a computer. SIN(x), for example, can usually be constructed directly, as the sine function is available on most computers. It is not very difficult to make tables for more advanced functions such as COSH(x) if you wish.

The construction of statistical tables is, however, another matter. In this article I want to consider how the two tables that are most often used, the normal and the chi-squared, can be stored for use in a program.

The normal table is not too difficult to construct, given some post 'O'-level mathematical knowledge, and it certainly has practical uses. For example, I have used it to program the evaluation of whether an investor should buy convertible or ordinary shares. Many random variables that occur in nature and everyday life follow the normal distribution, at least approximately, so it can be used in business or scientific situations to test whether practical results or trends confirm theories.

The chi-squared table is used to test observed frequencies of events against theoretical frequencies. An interesting example would be to test the random number generator on your computer by using the 'two-dice test'. You write a program to simulate two throws of a dice, then add the two scores together. The result of a large number of throws are accumulated in an array. If there were an infinite number of throws, then the frequency of each number in the range 2 to 12 would be exactly proportional to the number of ways of making the number: for example, there is only one way of arriving at 2 (both throws giving 1), but there are two ways of getting 3 (first throw 1, second 2; first throw 2, second 1) so there should be twice as many 3s as there are 2s. However, with a finite number of throws, the proportions will not be exact, and the chi-squared table can be used to find how much variation we could expect for the number of throws. You may well find that your random numbers are not as random as they should be!

It is not appropriate to go into mathematical detail here, but both the normal and the chi-squared functions are covered in enough detail by text books on statistics (and in probably greater depth than you'll ever need!) so look there for the theory and the applications if you are unfamiliar with these tables.

I would also like to add that while I think my normal algorithm is reasonably efficient, I am well aware that there are likely to be quicker and/or more accurate ways of constructing chi-squared. All I can say is that I haven't seen any attempts at this algorithm published, and if I goad readers into finding a better algorithm... isn't that the point of this page?

Segment one initialise the program which is written in detachable subroutines.

```
10 DIM W(40),Z(250),Y(250),X(250)
20 DEF FNI(R)=1-(Y(R)+Y(R-1))/2
```

Segments 2 sets up an array W giving the probabilities of a random variable from a given normal distribution being a given number of standard deviations (or more) below the mean. It is scaled in steps of .1 SDs, so that W(20) is the probability of a variable from the distribution being more than 2 SDs below the mean (or, indeed, above the mean by that amount). Line 510 gives the probability density function for a normal distribution and line 530 uses Simpson's rule to calculate the area under this curve.

The idea of degrees of freedom is the most difficult part of the chi-squared test to understand. If you know the theoretical frequencies and enough of the observed frequencies, say 5, to deduce the rest then the number of degrees of freedom is 5. The chi-squared variable with n degrees of freedom can be defined as the sum of the squares of n independent squares of standard normal

```
500 V=.5:W(0)=V:SQ=SQR(.125/ATN(1))
510 DEF FNN(X)=SQ*EXP(-X*X/2)
520 FOR K=.1 TO 4 STEP .1
530 V=V-(FNN(K-.1)+4*FNN(K-.05)+FNN(K))/60
540 W(K*10+.1)=V:NEXT:RETURN
```

variables. Sorry about that mouthful, but it explains Segment 3; if you're completely baffled, it's time to look for a friendly stats textbook.

```
1000 J=.1:FOR N=1 TO 100
1010 S=SQR(N*.1)*10:T=INT(S):U=S-T
1020 Z(N)=(W(T)+(W(T+1)-W(T))*U)*2
1030 NEXT:X(0)=1:Z(0)=1:Y(0)=1
1040 FOR N=1TO100:PRINT N/10,Z(N):NEXT:RETURN
```

The chi-squared variable with a single degree of freedom is obtained by the square of one normal variable. The array Z works in steps of .1. The probability that chi-squared (1) is greater than 3 will be given by Z(30).

```
2000 FOR N=1 TO 100:CZ=Z(N-1)-Z(N)
2010 CN=(N+.5)/10:CM=CM+CZ*CN:CV=CV+CZ*CN*CN
2020 NEXT:CV=CV-CM*CM:CS=SQR(CV)
2030 PRINT" MEAN: ";CM,"S.D.: ";CS:RETURN
```

Segment 4 is much simpler. Having the chi-squared (1) distribution in array Z, we calculate the mean and standard deviation of it.

```
4000 PRINT"▩":INPUT" ENTER VALUE OF CHI-SQUARED ";CH
4010 PRINT:INPUT" ENTER DEGREES OF FREEDOM ";DF
4020 NM=CM*DF:NS=CS*SQR(DF)
4030 R=10*(CH-NM)/NS
4040 IF R>40 THENPRINT" Significant at less than .00005"
4050 IF R<-40 THENPRINT" Very close to theoretical."
4060 IF ABS(R)>40 THEN RETURN
4070 IF R<0 THEN R=-R:KF=0
4080 Q=INT(R):T=R-Q:S=W(Q)-(W(Q)-W(Q+1))*T
4090 IF KF=1 THEN KF=0:S=1-S
4100 PRINT:PRINT" Significant at ";S*100;" % level."
4110 RETURN
```

If we have a large number of degrees of freedom, we are home and dry! If there are 20 degrees of freedom, chi-squared (20) will be approximately normal with the mean 20 times that of chi-squared (1) and the standard deviation increased by a factor of the square root of 20. This method wouldn't be too inaccurate for about 10 degrees of freedom. However, for a small number of degrees of freedom it is extremely rough and ready.

```
100 GOSUB500:GOSUB1000:GOSUB2000
110 PRINT" Press A for Normal approx.
120 PRINT" Press D for extra degree of freedom.
130 PRINT" Press E to end.
140 GET Z$:IF Z$="E" THEN END
150 IFZ$="A" THEN GOSUB 4000:GOTO110
160 IF Z$<>"D" THEN 140
200 IF ZF=0 THEN ZF=1:FOR N=1TO250:Y(N)=Z(N):NEXT:GOTO220
210 FOR N=1TO250:Y(N)=X(N):NEXT
220 FOR N=1TO100:X(N)=0:FOR P=1 TO N
230 J=FNI(N-P+1):X(N)=(Z(P-1)-Z(P))*J+X(N)
240 IF P>2 AND J<.001 THEN P=N
250 NEXT
260 IF N>10 AND X(N)>.999 THEN X(N)=0:N=250
270 X(N)=1-X(N):PRINT N/10,X(N)
280 NEXT:GOTO110
```

Referring to segment 6, lines 100-160 put these subroutines into a program, and lines 200-280 will construct the missing chi-squared arrays. I am not sure whether you would call the process inductive or recursive. The idea is that an extra degree of freedom is added each time the loop is done. If, for instance, you pass line 200 for the fifth time, Z contains chi-squared (1), Y contains chi-squared (4), and X is being made into chi-squared (5). Each new value that is put into X is in effect the result of a double integral.

The program is slow and only reasonably accurate. You could sacrifice speed for accuracy or vice versa. I haven't even tried to produce a tidy screen or done anything much about input checks; it isn't that sort of program. The whole is a series of subroutines; you do not need to use all of them. There is also a challenge to you to find an alternative algorithm that works quicker or more accurately.

On the credit side, I have tried to keep to the Editor's advice and not be machine specific. The program here was developed on the Einstein, printed on the Sharp MZ80K in a home made Basic, and for good measure, the text was word-processed on the MZ700!

HIGH TECH HOME BASE

Richard Porch

The time will come when the home-computer set-up plays a more prominent role in the home, becoming an integral part of 'the house' as a system. This is the opinion of Richard Porch in this, his second Viewpoint feature.

Take a good look at your own current home-computer set-up, colour TV monitor, computer printer, ram-packs, floppy disk assembly etc etc. Yards of wiring snaking about everywhere, lots of isolated little black-boxes, squatting all around you. Everything strung out and bitty. Wouldn't it make more sense if it was all 'stacked' together into one mobile and tidy 'resource tower'?

The resource tower is an idea, in my opinion, whose time will inevitably come. In the old days, if people wanted a record player, cassette player and radio, they had to buy them separately. Three individual pieces of equipment dotted around the room. Nowadays, of course, the music-centre packages them all into one compact (and relatively mobile) unit. The option to buy radios etc. separately still exists, of course, but it makes sense to buy a packaged 'centre', from the point of view of saving both money and space in the home. This space-saving economy, surely nowhere holds more good than in the home-computer field. And with the domestic robot idea just beginning to catch on, there will be a marvelous opportunity to combine the two into one exciting cybernetic utility device. Music centres are, after all, essentially passive devices, which are simply set up to operate and then ignored. But imagine a computerised resource tower that had an inbuilt TV monitor, a full range of clip-on accessories for the home-computer and a keyboard (in a foldaway compartment). Then imagine that this metal and plastic tower has an artificial computerised intelligence of its own; is responsive to commands and is also capable of plugging into other computer systems. Add some sort of battery powered power trolley in order to make it mobile and we have our 'high-tech home-base'.

The very basic and crude (not to say gimmicky) work being done in the field of domestic robots is at least a step in the right direction. However sensationalist and silly. The 'high-tech home-base' will not necessarily be anthropoid but instead might tend to look a little like a satellite. With a lot of mechanistic 'bumps and grilles' and ports for plugging things into and out of it, it will naturally have to be slim enough to get through doorways and may even be 'tracked' in order for it to deal with steps and the like. But it will be more, much more than a computerised cocktail cabinet. Having summoned your 'home-base' domestic robot to you by remote control, you will then have it at your service. It will be able to offer word processing (via dictation), medical diagnosis (using its medical knowledge-base), you will be able to shop (via its link to the 1990's cable-shopping network) and simply ask for advice (about everything from household economics to fixing the roof). This is of course in addition to a whole host of more mundane chores such as remembering to switch on the video for you, de-bugging your programs and paying bills via electronic mail etc. The 'home-base' will not be the malevolent or evil creation of science fiction - a sort of suburban R2D2, even down to the fake butler's voice - in fact it will more probably resemble C3PO in its compact and mechanistic appeal. Although, the more demanding you make its tasks, obviously the more humanoid it will tend to look. Some form of telescopic functioning hand would be desirable, from the point of view of opening books and fetching and carrying etc. But the 'home-base's' main use will be as the aforementioned resource tower, for all the bits and pieces of your advanced micro-computer set-up. No more haggling with parents, paranoid about 'burning-in' on the colour television's precious tube. The 'home-base domestic robot' will come complete with its own multi-channel television receiver, capable of tuning in not only to the normal TV channels but to all the satellite ones and the cable network too. And will naturally be capable of receiving and sending messages along the 'home-base' network - the ultimate in 'citizen band' communications networks. Such 'home-base domestic robots' needless to say, will need a memory capacity measured in gigabytes and will probably use (the still developmental) transputer technology. It will need some sort of camera vision system, for providing it with sight. And sonar 'sounding' equipment like radar, to direct it around the many obstacles to be found in the home. None of these 'seeing and hearing' systems need to be very sophisticated, as the 'home-bases' real virtues lie in its computing-communications capacity and relative compactness over all existing systems.

The 'home-base' computing robot as it may come to be called, would be a kind of 'oracle' to the family. Available to dispense advice and solve problems at the push of a remote control button. It would not lumber around like some dalek, dusting furniture and doing the washing-up. It's true, at the moment, that the domestic robot idea is gaining ground and we've seen prototype versions of them responding to various kinds of instructions of varying degrees of frivolity. But this is simply because we are at the same basic and infantile stage that the microcomputing industry was, say five years ago. Once the novelty factor wears off and people can see the logic of packaging our computer systems into responsive resource tower robots we'll see the idea really take off in a serious way. A period when ultra-sensitive systems will be at our elbow, and all the slog (and not to say mystique) of computing will disappear, as we summon our 'home-base' computers and issue all our instructions verbally. With the option of having the response in a verbal or printout form by return.

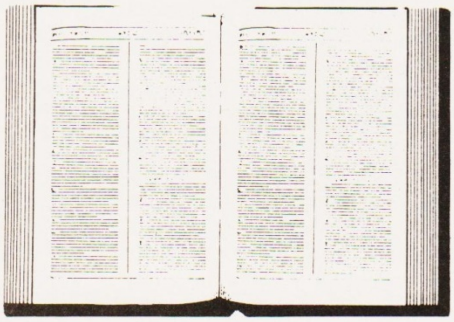
Our 'home-base' computer will be an expert system, in fields as wide as its owner wishes to make them, and will obviously be adept at 'multi-tasking'. While one person uses it to play video games, another might use it to compute household expenses, while a third might listen to an audio lecture given by the robot on how to check for faults in the plumbing. They even might all use it at the same time, running extension leads to still more equipment, all the while boosting the power of the basic 'home-base'. The 'home-base' itself might be capable of sprouting acoustic bubbles, in order to provide intimate quiet zones in order for the user to concentrate on the task in hand.

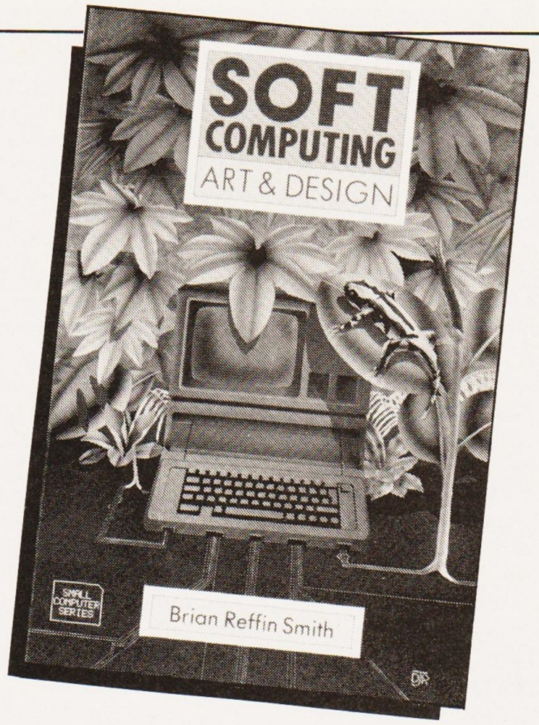
The outline I've tried to broadly sketch in, sees the machine as a repository of a vast and multi-disciplined variety of knowledge. Using its computerised power of super-human recall and enormous capacity to sort and sift through masses of data and information, to produce an answer. However, these machines will inevitably be superceded by still more powerful fors of artificially intelligent devices. We will eventually arrive at the independently motiviated domestic robot. A 'home base' robot that will simply require re-charging and nothing else. This will truly be an anticipatory, thinking, Cyborg, that will be capable of not only responding to tasks asked of it but will also, by observation and deduction, be capable of identifying its own tasks and of proceeding to carry them out. A genuinely interactive artificial intelligence, with its own synthetic voice, and independently targeted problem solving motivation impulse. This does sound a little disturbing, the idea of a domestic robot leading a 'life of its own'. But as long as that 'life' is spent relieving its owners of the tedium of low-level or repetitious chores, why worry? Such 'home-base' robots could take on all manner of mundane tasks, freeing the individual for more creative or rewarding activities. At the start or end of each day the 'home-base' robot might glide into view and supply you with either a verbal or written itinerary of its proposed activities for the day. You might amend or add to the list as you saw fit. With the 'home-base' going on to fulfil its schedule, while you are at work or even asleep. It will always be easier for humans to carry out the strictly physical tasks about the house, cleaning, cooking, washing or even just turning on the TV. To try and construct a robot (let alone program it!) to do something even as simple as slicing and buttering a round of bread, would cost more than it could ever be worth. Anyway the designing of high-technology robotics, with powerful artificial intelligences for such mundane manual work is to totally misuse the technology. The trick will be to correctly perceive the future 'home-base' robots as extensions of our minds not our bodies. Their task will really be to marshall data, analysing and advising us, after perhaps consulting even more powerful systems than itself. And then to put their specialist knowledge into advice which we can both understand and put to good use. For our part, we will have to acknowledge the superiority of their network of contacts and their intelligence. And above all trust their offered advice as being in our best interests. The way we now trust pocket calculators and other computers to provide correct solutions to strictly mathematical problems without endless re-checking. When home computers stop sitting self-consciously on table-tops, with their manifold accessories dotted around the keyboard, with cables snaking all over the room and they integrate with the steadily evolving domestic robot field, we will see a unique marriage of technology and utility that will transform our homes and our lives.

BOOK PAGE

Garry Marshall

'Rush out and buy it' sums up this month's book - 'Soft computing: art and design' by Brian Reffin-Smith.





This month's book is: 'Soft computing: art and design' by Brian Reffin-Smith - (Addison-Wesley), 196 pages, £10.95.

It isn't often that a really different computing book comes along, but one certainly has this month. It is **Soft computing: art and design'** by Brian Reffin-Smith. Although the book deals with many of the same themes as all the other books on computing, it is not at all easy to say what the book is about. In fact, if I tried to list all the matters that the author claims that the book is about, this review would be as long as the book itself! I hope that I do not over-simplify too much by saying that it deals with a new style of qualitative computing that is intended by Reffin-Smith to give the computer a fruitful role in art and design.

Brian Reffin-Smith has been forcefully putting the case for computers in art for some time as lecturer, artist, and writer. He was on the staff of the Royal College of Art, and the book contains many acounts of his teaching and interactions with students in attempts to explain his ideas on art, design and computing. One or two of his efforts as a 'computer artist' are described in the book. Perhaps closest to home as far as we are concerned is that he has written books and articles about computing. At one time, he contributed a regular column to 'Practical Computing' for which, by itself, the magazine was worth buying. (Now that he no longer writes for them...?)

POINT OF VIEW

I have to admit that my consideration of the book is rather one-eyed, for this is quite as much a book about art, and a book about the theory of design, as it is a computer book. I suppose that I read it mainly as a computer book. I can claim no particular expertise in art and design. I can only console myself, and anyone else who finds my remarks inadequate, with the thought that there can be very few people who can claim to be familiar and comfortable with all the strands and themes that are drawn into the fabric of this book.

Overall, the book is bubbling with ideas and practices drawn from computing, art and design. They do not always seem to me to be drawn together as fully as they could be, and sometimes their connections escape me altogether, but one can see the overall direction of the route by which a fusion could be achieved. The author's writing seems to follow his 'stream of consciousness', which gives us an entertaining read but, at the same time, means that he is writing of what is at his immediate command, is ever willing to side-track himself and can't really pause to investigate further a matter that might need it. I should say, though, that his references to other people's writing are usually meticulous, although he is occasionally liable to remark that an idea is someone elses, but that he can't remember whose.

If this is beginning to sound critical, I should say that the book is always readable, lively, and, perhaps most welcome of all, stimulating. Some carping is prompted by the feeling that there is an important book trying to find its way between these covers but that, partly because of some lack of organisation and a certain amount of self indulgence, it hasn't made it. Since Reffin-Smith has been worrying at these matters in print and in lectures for some time it would be good for him, and for us, if it did.

NO MAGIC

The book begins by 'de-mystifying computers'. This is apparently considered necessary in the fields of the art and design, where the perception of the computer is that it can only do 'hard' things such as number crunching, and that it cannot do 'soft' things such as producing designs for imprecise specifications like telling it to colour something 'quite blue' or to position one thing 'fairly near' to another

The breadth of the matters covered during the de-mystification is perhaps indicated by the following list, which also shows what seem to me to be some of the unexpected, but typical insights that are presented along the way:

- What a computer does can always be described by "stuff in things done to the stuff/stuff out"
- "We are even approaching the time when the computers themselves become throw-away objects. (...while small computers can now be had in Britain for less than half an average week's wages, this is of little use to people in Third World countries with less money and more pressing problems.)"
- "It is still unfortunately true, and quite ludicrous, that we often address the most powerful machines ever invented by means of 19th century typewriter technology, and receive the output on early 20th century television devices."
- We can view the development of computers through the categories of "origins in terms of 'need', nature of components, cost and size, their 'presence', and the 'sociology' of the machines and the people around them."

These extracts show quite well the sort of side-track that can unexpectedly be turned into, and the desire to put computers in as wide a context as possible. (As an irreverent side-track of my own, considering that sociological context and meaning are important, the author might have reflected on his title rather more, for the meaning that could be attached to the phrase 'Soft computing' in some sections of society is not likely to give an attractive impression of the contents of his book.)

SPLIT LEVEL

The de-mystification takes place on two levels, with a more-or-less technical explanation of how computers work preceding an explanation in terms of how it can be used as a tool by designers and artists. The relation between the two levels, or at least the intended relation, is captured in this paragraph:

> It seems obvious to me that writing a computer program, or thinking how a computer might be used to solve some problem, or using a computer to make art or design are all design activities. Then why have these things not been subjected to the same kinds of questions and studies as design?

The same thing can be said about designing computers, of course.

Now, the idea that comes to my mind is that a lot of thought has been given to the design of programs, although whether it has followed the same course as that about design in general I do not know. The results of it are manifest in program generators (programs that write programs from general descriptions of what it is that they are required to do) and, at a different extreme, in a theory of program correctness that goes at least some way towards showing whether a program is correct or not. So a lot of thought has been put into program design, and it has produced valuable results.

As far as designing computers is concerned, it is hard to see how modern computers would exist without recourse to computer-aided design. The computer can be used successfully as a design aid. And if something as complex as a computer can be designed, what cannot? Also, many of the designs that are produced, whether they are displayed on a screen or plotted on paper are aesthetically pleasing quite apart from any consideration of the primary reason for their existence. This may or may not make them works of art, but it does demonstrate that a computer running the appropriate software can produce results that meet many of the criteria that need to be satisfied by a work of art. More than this, the results are used as works of art: people hang them on their walls.

BY EXAMPLE

To illustrate the difference between what he means by 'soft' and 'hard' descriptions, the author contrasts a free-hand sketch of a bowl with the mathematical formula for its surface. Since the bowl is the normal rounded shape that we would expect, the equation he gives is $Z = X^2 + Y^*$. (He says that this is the equation of a porabola, but it is actually that of a paraboloid, and he only wants it for a limited range of values of Z, but that doesn't affect his main point.)

This example distinguishes quite clearly between the ways that 'soft' and 'hard' practitioners would like to present their inputs to the computer. But to go on to imply that the 'soft' description is much richer than the 'hard' one is something I can only disagree with. It may be a more appropriate description in given circumstances, but consider this. If the two descriptions are both completed designs for an artefact, then when it comes to making the artefact, the 'hard' description says exactly what it is to look like: it even contains enough information to control a machine to make the artefact. A numerically controlled cutting tool could cut a piece of wood to precisely this shape, for example. But the 'soft' description only says approximately what it should look like. When it is made, the finished product will have a definite shape of its own, but the designer will have abdicated some control over it, presumably to an item of software.

This difference in approach has shifted the areas of responsibility of the design. The 'hard' approach found the designer in complete control - the 'soft' approach caused some of the design to be left to the software and therefore to be done, implicitly, by the writers of the software. The division of responsibilities will matter more or less depending on whether the result is supposed to be a commercial product or a work of art or, indeed, anything in between.

Elsewhere, it is claimed that programs to be used for art and design need not (the claim is almost *should* not) be well structured. This is justified by saying that most design does not have well-understood goals: that often designers do not have a clear idea of the problem they face, have only a vague idea of their goals and cannot really tell when they have satisfied their goals. This is all encompassed in the quote that life is an ill-structured problem. Again, I have to take another tack. Designers do talk about problems and, for example, architects refer to their designs as solutions. If we don't know what we want the computer to do, how can we get it to do it? It isn't simple to deal with the point : articles are now being written, for instance, on how the Macintosh's MacPaint program can be used to draw out and concretise ideas.

But the main thing that I take issue with is that programs to deal with problems in which no structure is apparent must themselves be unstructured. This seems to me to be a confusion of ideas. We deal with a complex problem by modelling it. To do this we extract the underlying parts and their relations as they are relevant to our studies and explore this simplified model in order to understand the original. If the program for examining the model is not structured, it is difficult, and becomes increasingly difficult, to amend it to include further items which, as it may subsequently become clear, are important, or to modify the model in any other way in the light of investigations with it. Unstructured problems, yes: unstructured programs, no!

NO DIFFERENCE

The book makes much of computing with representations, as opposed to computing with numbers, as if this was one essential difference between 'soft' and 'hard' computing. I would suggest that it is not and, further, that all computing is computing with representations. The patterns of binary digits inside a computer can represent numbers, but they can equally represent anything else. Just because we are usually provided with functions for transforming numbers does not mean that a pattern of binary numbers cannot represent something else, a colour, say, and that we cannot devise a function such as MIX, which when given the inputs 'blue' and 'yellow' transforms them into the output 'green'.

Again, in Lisp a variable can be assigned any number of attributes. One of them can be a numerical value to be sure, but it can also have a colour, a shape, a position, a probability of occurrence and so on.

Also, to suggest that numerical computing is perfectly 'hard' and quantitative is to ignore the fact that as soon as you start to deal with real numbers (that is, numbers that are real in the mathematical sense that they can represent the position of a point anywhere on the real line) they cannot be represented exactly. Numbers like one-third need an infinite number of decimal places (and binary places) to be represented exactly, but they have to be stored in a finite amount of memory. An apparently quantitative numerical computation at once assigns the quality of approximation to each number that it deals with.

In this way, quantitative computing is qualitative, and we have already seen that qualitative computing must ultimately be quantitative. So 'hard' and 'soft' computing are not distinguishable to the computer. They are only different to us, its users. Perhaps the distinction is necessary because the designers among us receive much the same computer education as the 'hard' liners; if this is so then their education is clearly inappropriate. Reffin-Smith's book contains an appendix giving outlines for courses to introduce computers to designers. If such courses are given widely, perhaps the distinction between 'hard' and 'soft' will not be needed, or exist, to the same extent. I don't think that all the softies will become hard men after their courses, but courses appropriate to the needs of students of all kinds will help to bring us all nearer to making real use of computers.

So that's my reaction to 'Soft computing: art and design'. I found it stimulating and provoking. I didn't always agree with it. I found it highly readable.

I would recommend that you go out and buy it. It contains a good deal more than I have managed to indicate here. And when you have read it once, be prepared to read it again.

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)

BARCLAYCARD VISA

Please use BLOCK CAPITALS and include postcodes.

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

ADDRESS ....

....

.... POSTCODE ....

Signature ....

Date ....

SAVING SPACE

J.G. Gibbs

Remove unwanted line feeds from your listings with this short routine for the Amstrad CPC464

In the CP 464, a stream of ASCII characters may be directed freely to either the VDU or to a printer. In order to make the format acceptable to the VDU, each line is terminated with both the CR code (value 13) to return the print position to the head of the current line, and the LF code (value 10) to advance to the next line. Most printers will interpret either of these codes as a combined CR/LF operation; with the consequence that where both are sent to the printer a double line spacing will result.

Whilst the best solution must be an adjustment (e.g. a dip switch) at the printer which will remove the response to either one of the codes, the AMSTRAD published firmware contains a mechanism which may be employed to achieve the same effect. The disadvantage is that the software mechanism is reset by a general machine reset, and therefore may not be use able with commercial software.

The essential routine in the printer driver is MC WAIT PRINTER, whose entry is to be found in the indirections block at BDF1. The 3-byte block at this address will be found to contain the jump instruction JP 07F8. It is therefore possible to replace this jump address by the address of a suitable machine code routine whose function is to replace the CR code by an innocuous value (e.g. zero) and so eliminate the redundant line.

The 16-byte routine is poked by the program into locations from AB00. Lines 20 - 50 plant a jump to AB00 into the indirections block. The routine itself first preserves register A (which holds the character to be printed) by pushing the register pair A, F onto the stack. The exclusive OR of the character with value 13 can only be zero if A itself contains the value 13, ie the CR control code.

If the result is non-zero (any other character) the jump to AB0C results in the registers A, F being restored from the stack. Otherwise, A and F are restored and A is replaced by zero. In either case control reaches AB0D, where the original jump from the indirection block is obeyed to print the character.

The final line (220) sets the top of BASIC memory to the address immediately below the head of the routine. (AAFF)

In use, the program is loaded and run before any use is made of the system, and will remain available unless a general machine reset is performed. It will not be destroyed by a NEW command. In order to make the routine generally relocatable, it is only necessary to replace the jumps in lines 90 - 110 and 150 - 170 by the equivalent relative jumps, and modify the indirection replacement.

```
10   REM AMSTRAD CP 464 - REMOVE DOUBLE LINE SPACING IN PRINT
20   REM           (J.G. GIBBS, 11 FEBRUARY 1985)
30   POKE &BDF1,&C3: ' JP AB00
40   POKE &BDF2,&0
50   POKE &BDF3,&AB: ' REPLACE PRINTER INDIRECTION
60   POKE &AB00,&F5: ' PUSH A,F
70   POKE &AB01,&EE: ' XOR A, 13 (CR VALUE)
80   POKE &AB02,&D
90   POKE &AB03,&C2: ' JPNZ AB0C
100  POKE &AB04,&C
110  POKE &AB05,&AB
120  POKE &AB06,&F1: ' POP A,F   (CLEAR OUT OF STACK)
130  POKE &AB07,&3E: ' LDA 0
140  POKE &AB08,&0 : ' REPLACE CR CODE BY ZERO
150  POKE &AB09,&C3: ' JP AB0D
160  POKE &AB0A,&D
170  POKE &AB0B,&AB
180  POKE &AB0C,&F1: ' POP A,F  (REPLACE CHARACTER IN ACC.)
190  POKE &AB0D,&C3: ' JP 07FA
200  POKE &AB0E,&FA
210  POKE &AB0F,&7: ' REINSTATE PRINTER INDIRECTION
220  MEMORY &AAFF   : ' RESERVE SPACE
230  END
```

PHOTOCOPIES

A COMPUTING Today Reader Service

To take advantage of this simple service, just fill in the required information and send it (or a photocopy) to:

COMPUTING TODAY Photocopies,
No. 1, Golden Square,
London W1R 3AB.

together with your money and we'll do the rest!

CT PHOTOCOPIES

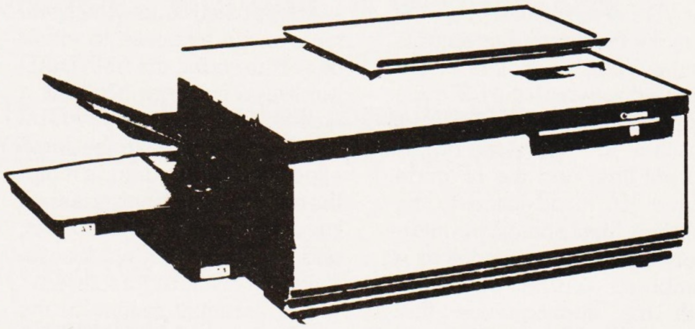
NAME ....
ADDRESS ....
....
.... POSTCODE ....
Please send me Photocopies of the following items

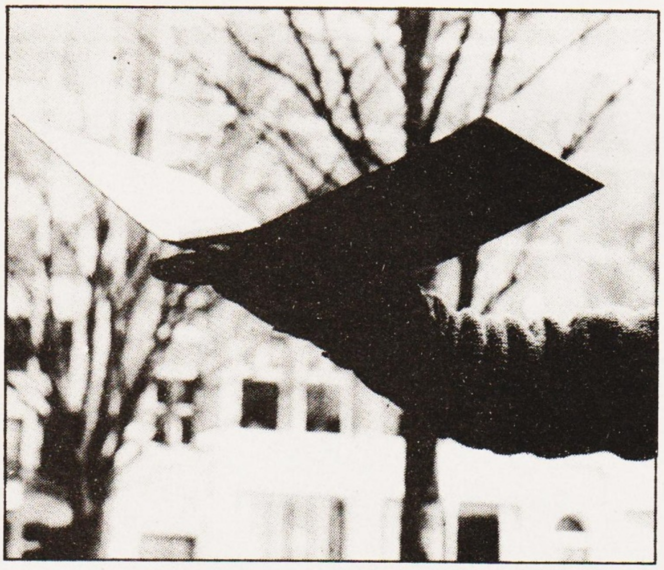
ISSUE	ARTICLE	PAGES

At £1.50 each, I enclose £ ....
Cheques and Postal Orders should be made payable to ASP Ltd.

Lost and can't be replaced? Well, if you've lost one of the early issues that cannot be replaced from our stocks of backnumbers, all is not quite lost.

If you know the article name and the issue it appeared in, we can supply you with a photocopy for the miserly sum of £1.50 including postage and packing. If you're stuck for the month, try checking in our annual index which is included in the December issue each year.





In days gone by, falconry was the sport of gentlemen and kings — this noble and time-honoured tradition is not so prevalent in these technological times, and it is quite a pity, too. Just imagine the pride you'd feel standing in your own back yard while your very own hunting falcon swooped down upon unsuspecting dogs, cats and Ford Sierras.

For a limited time only, Computing Today is offering you the chance to experience the thrill of commanding your own bird of prey, with the new CT Hunting Falcon/Magazine Binder. Swift of wing, sure of eye and made of genuine vinyl and cardboard, the Computing Today Hunting Falcon/Magazine Binder is the spitting image of the hunting birds of old to anybody suffering from cataracts. Release it from your arm, and it dives just like a traditional hawk. If it lands on a small animal, it will probably stun it. Also, when you tire of the sport, and would rather hunt hedgehogs with your Ford Sierra, your CT Hunting Falcon converts into a useful magazine binder that holds a full year's supply of Computing Today. The new CT Hunting Falcon/Magazine Binder will cost you not a farthing more than the old binder alone used to: just £5.00. This includes postage and packing, so your falcon won't have to tire itself out flying to your abode.

Cut out and send to:

COMPUTING TODAY HUNTING FALCON/MAGAZINE BINDER,
INFONET LTD,
TIMES HOUSE,
179 THE MARLOWES,
HEMEL HEMPSTEAD,
HERTS HP1 1BB,
ENGLAND

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)

Insert card no. | | | | | | | | | | | | | | | | |

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss) ....
delete accordingly
Address ....
....
....
Signature ....
Date ....

FIVE YEARS BACK

(from Computing Today June 1980)

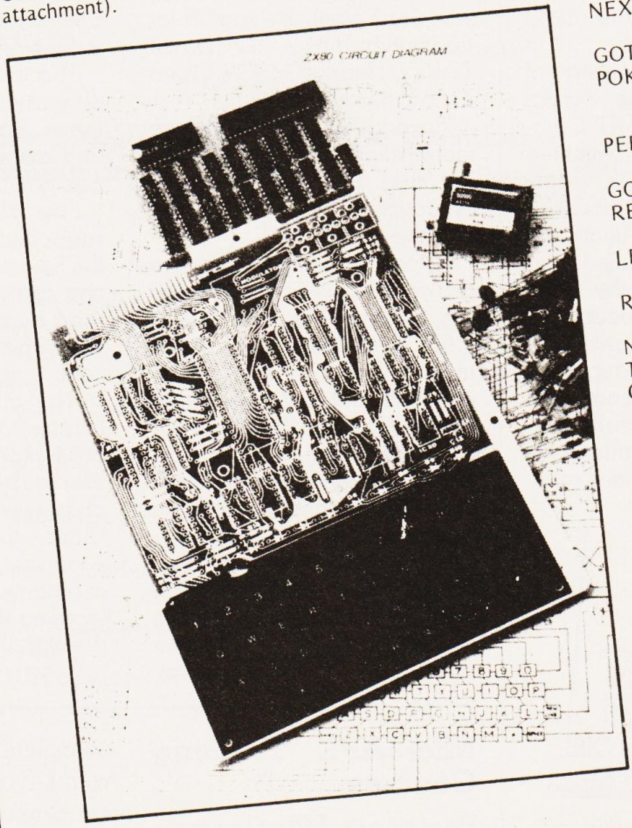
ZX80 REVIEWED

Appraisal
The ZX 80 is arguably the first complete system. The novice or first time programmer who has never had access to any kind of computer will be perfectly satisfied with his (or her) investment. It is unlikely that it will make anyone into a good BASIC programmer overnight, it will, however, teach the fundamentals well and for this it is highly commended.

Pros And Cons
To give a brief view of the facilities we have produced a table which may give a quick overall impression.

Pros:
Z80 CPU, with powerful instruction set for machine coders.
BASIC language, which is easy to learn for the rest of us.
Large quantities of software "free" in magazines.
Cheap and easy to build, with good instructions.
Excellent screen editor and syntax checker.
Good first system.

Cons:
Lack of machine code explanation.
Poor BASIC manual.
Incompatible BASIC with only integer arithmetic.
Cassette format incompatible with any other system.
Poor expansion capability.
Non memory mapped display (means no graphics games).
Use of non-standard character codes (means no direct printer attachment).



System Commands:

CLEAR	Clears variables
CLS	Clears screen
BREAK	Halts run and returns to command stage
EDIT	Returns current line (at pointer) to bottom of screen for editing.
LIST	Lists specified line(s) of program
STOP	Halts execution
NEW	Clears memory and awaits new program
LOAD	Loads from cassette interface
SAVE	Writes current program to cassette interface
RUN	Executes current program
CONT	Continues STOPped program
HOME	Moves cursor to top line of program list
RUBOUT	Deletes character to left of cursor

Statements:

REM	Allows inclusion of text for comments
IF	Comparative
INPUT	Assigns value of keyboard input to variable
PRINT	Prints specified variable to the TV screen
DIM	Dimensions array size
FOR	Used in conjunction with NEXT to execute a defined loop
NEXT	Returns to FOR statement if loop counter not full
GOTO	Jumps to specified statement
POKE	Places assigned value in defined memory location, used in machine code programming
PEEK *	Returns value of specified memory location
GOSUB	Jumps to defined subroutine
RETURN	Returns to main program from subroutine
LET	Assigns specified value to specified variable
RANDOMIZE	Resets seed value of random number generator
NOT, AND, THEN, TO	Comparitive "tokens"
CHR$ *	Produces the character whose code is specified
STR$ *	Produces the decimal value of the specified string
TL$ *	Returns the specified string less the first character
CODE *	Produces the code for the first character in a specified string
RND *	Generates a pseudo-random number within defined range
USR *	Calls machine code routine located at address specified
ABS *	Returns absolute value of specified variable

Table 1. The BASIC functions as implemented on the ZX80, all available at a single keystroke except those marked *.

25

COMPUTING TODAY JUNE 1980

COURSES AND TRAINING

Education around the U.K.

IT CONVERSION COURSES AT STRTHCLYDE

The University of Strathclyde has won support from a wide range of funding agencies to run two post-graduate courses in Information Technology (IT).

The funds are being used to run intensive training and education programmes and to pay the fees of graduates from other subjects who wish to enter the rapidly growing field of It.

Support for the University's course in Information Technology Systems, which leads to the degree of Master of Science (MSc) or the University Diploma, is being given by the Science and Engineering Research Council, the Manpower Services Commission, and the British Council.

The course is being run by the University's Departments of Computer Science and Electronic and Electrical Engineering, with support from Strathclyde Business School, and it is aimed at graduates in numerate subjects other than computer science and electronics. Students who complete the course can expect to find careers as IT designers and innovators, IT applications specialists and advisers in industry and business.

The second course, Business Information Technology Systems (BITS), is being run by Strathclyde Business School along with the two specialist departments of Computer Science and Electronic and Electrical Engineering. The BITS diploma course is receiving financial support from the Scottish Education Department and the Manpower Services Commission and is designed to give non-science graduates an understanding of IT applications in business and management.

Professor Tom Carbery of Strathclyde Business School and chairman of the IT Campaign/Campaign Committee in Scotland said: "The widespread support we are receiving for these courses reflects the national need for skilled personnel in IT, and at the same time the support recognises Strathclyde University's achievements in IT education. The particular strength of Strathclyde is that we have a wider range of departments involved in IT than most other British universities.

Enquiries to:
Mr R E Fairley,
Department of
Electronic and
Electrical Engineering
Tel: 041 552 4400
Ext 2544
Professor T F Carbery OBE,
Telephone 041 552 4400
Ext 3089
Dr J Aitchison, Information Officer
Telephone 041 552 4400
Ext 2182/2370

OPEN LEARNING SOLVES TRAINING PROBLEM

Open learning techniques that allow people to update their skills without disrupting their work will soon be playing a major part in training engineers in CAD-CAM, robotics and automation.

"One of the main problems facing companies is that the staff who most need to learn these new techniques are some of the most vital to their operation, who can't easily be spared for traditional courses," says David Tinsley, Head of MSC's Open Tech Programme.

"In addition, employers are increasingly demanding training that is specific to their operation and which doesn't cost the earth.

"Open learning is often the answer because it allows those key personnel to learn about CADCAM, robotics and automation when it is convenient to employer and employee, either on the firm's premises or at the employee's home.

"Open learning materials are cost-effective and they can be customised to be company-specific. Many firms are also using them to provide training in the basics of these emerging technologies, before selecting candidates for further, advanced work."

The Open Tech Programme has given financial support to companies, colleges and professional institutions to develop open learning materials in new technology subjects.

This involves the production of study texts, videos, audio tapes, computer programmes and kits of hardware that learners can use at work or home.

Five of the projects being supported by Open Tech are in the field of CADCAM, robotics and automation, and they will soon be making a major contribution to training in this field.

"Not all these projects are fully operational at the moment, but I expect that by next year they will be reaching thousands of people." said Mr Tinsley. "After that the scope is enormous, and I fully expect companies to learn the open learning lesson and start to develop their own systems."

MicroAPL Training Courses

Beginners APL Course - Wednesday April 17th

An introduction to APL for the complete beginner, starting with keyboard practice and working up to writing simple programs to perform business calculations and produce reports.

Immediate APL Course - Wednesday June 5th

Aimed at those who have taken the Beginner's course, or others of similar standard, this course builds on existing knowledge to cover more advanced APL facilities, including the details of function definition, program structure, optimising the use of the workspace, and how APL is used in interactive systems.

Advanced APL Course - Wednesday July 3rd

The third of MicroAPL's graduated APL courses, showing the budding APL expert how to use the utilities and features of MicroAPL systems, how to file data and programs, and how to apply APL knowledge in the writing of productive business and scientific systems.

Beginners Mirage Course - Wednesday May 1st

An introduction to MIRAGE, the multiuser, multitasking operating system. This course considers such topics as defining jobs, terminals and devices, memory allocation, system security and running programs.

Advanced Mirage Course - Wednesday June 9th

Aimed at system managers and others wanting a detailed insight into the Mirage operating system, this course considers such topics as allocating CPU time and disc resources, housekeeping, networking, writing assembler programs such as terminal drivers, and Mirage software other than APl.

Writing Multi-User Systems - Wednesday May 8th

This course is aimed at the expert APL programmer. Topics covered include establishing user signons, sharing data between users, setting up foolproof multi-user file systems, and running background jobs. Some knowledge of Mirage is assumed.

Writing Auxiliary Processors - Wednesday May 22nd

For the adventurous user who wants to take advantage of APL.68000's unusual flexibility in this area, this course describes how to write your own auxiliary processor and interface it to APL. Motorola 68000 assembler knowledge needed.

Course dates and availability may be subject to change.

MKA TRAINING COURSES

A new guide from MKA Training outlines the number of courses running until July this year.

The guide describes MKA's scheduled public training courses and is issued on a six monthly basis.

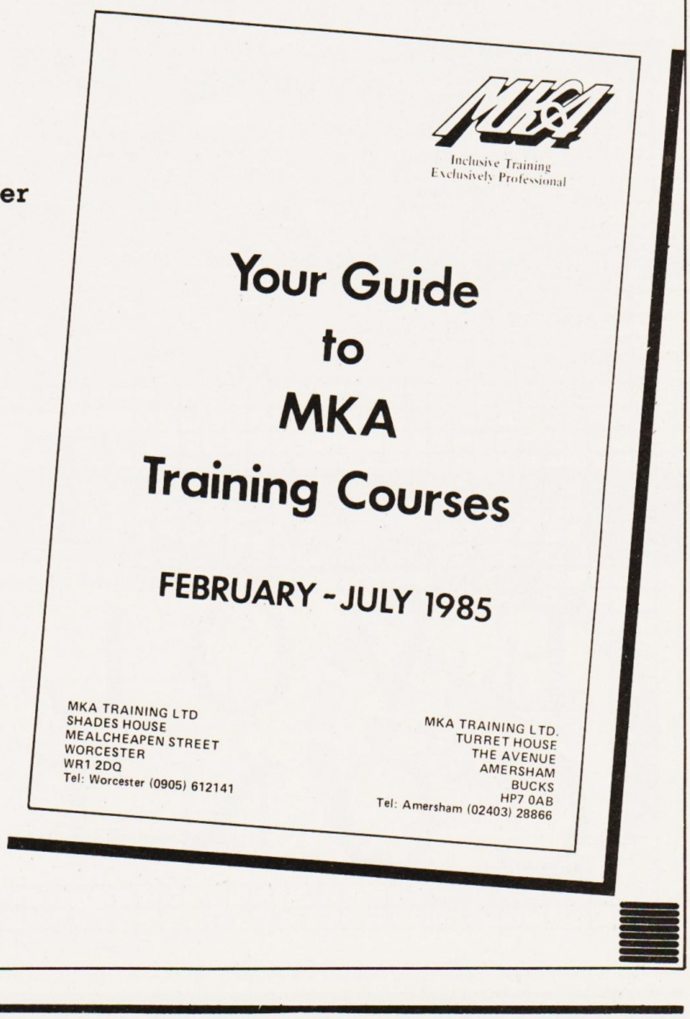
Courses on offer from MKA include:

- **Program planning and techniques**
- **Disk file techniques (BASIC)**
- **Basic machine code programming**
- **Advanced machine code programming**
- **Using the UNIX operating system**

and

- **Programming in 'C'**

Enquiries about any of the courses should be directed to MKA on (0905) 612141. □



MKA
Inclusive Training
Exclusively Professional

Your Guide to MKA Training Courses

FEBRUARY - JULY 1985

MKA TRAINING LTD
SHADES HOUSE
MEALCHEAPEN STREET
WORCESTER
WR1 2DQ
Tel: Worcester (0905) 612141

MKA TRAINING LTD.
TURRET HOUSE
THE AVENUE
AMERSHAM
BUCKS
HP7 0AB
Tel: Amersham (02403) 28866

COMPUTING Today

NEXT ISSUE OUT FRIDAY JUNE 18TH

MAPLE - of the many uses a micro can be put to, this has got to be one of the most extraordinary. Detecting changes in landscape usage by digitising aerial photographs and comparing the data to known records is what this little package does. Find out more in the next issue..

CRIBBAGE - another new series gets under-way next issue, this time showing how to teach your micro Cribbage.

PASCAL - we take a look at Oxford Computer System's implementation of this language.

Z800 - Zilog have kept very quiet about their up-and-coming processor, so we asked them to tell us what's going on.

All this and more in the next edition of

COMPUTING TODAY

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

THE EVOLUTION CONTINUES..

MAIL ORDER ADVERTISING

British Code of Advertising Practice

Advertisements in this publication are required to conform to the British Code of Advertising Practice. In respect of mail order advertisements where money is paid in advance, the code requires advertisers to fulfil orders within 28 days, unless a longer delivery period is stated. Where goods are returned undamaged within seven days, the purchaser's money must be refunded. Please retain proof of postage/despatch, as this may be needed.

Mail Order Protection Scheme

If you order goods from Mail Order advertisements in this magazine and pay by post in advance of delivery, Argus Specialist Publications Ltd will consider you for compensation if the Advertiser should become insolvent or bankrupt, provided:

(1) You have not received the goods or had your money returned; and

(2) You write to the Publisher of this publication, summarising the situation not earlier than 28 days from the day you sent your order and not later than two months from that day.

Please do not wait until the last moment to inform us. When you write, we will tell you how to make your claim and what evidence of payment is required.

We guarantee to meet claims from readers made in accordance with the above procedure as soon as possible after the Advertiser has been declared bankrupt or insolvent (up to a limit of £2,000 per annum for any one Advertiser so affected and up to £6,000 per annum in respect of all insolvent Advertisers. Claims may be paid for higher amounts, or when the above procedure has not been complied with, at the discretion of this publication but we do not guarantee to do so in view of the need to set some limit to this commitment and to learn quickly of readers' difficulties).

This guarantee covers only advance payment sent in direct response to an advertisement in this magazine (not, for example, payment made in response to catalogues etc., received as a result of answering such advertisements). Classified advertisements are excluded.

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.

MICRODEALER — ORDER FORM

Please include my business details in the next available issue of Computing Today:

Business Name: ....

Address: ....

....

Details: ....

Tel. No.: ....

Open Hrs: ....

Contact (Office Use Only): ....

Post To: Computamart, Computing Today, 1 Golden Square, London W1.

ONLY £20.00

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.

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

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
ASP LTD, 1 GOLDEN SQUARE,
LONDON W1

HARDWARE

NEWBRAIN & SANYO
Professional Micro Computers for the prices of hobby machines.

NEWBRAIN ON SPECIAL OFFER
£99 PLUS VAT
Model AD with free beginners guide and tape, investment software and UK postage. As above plus about **£100** value of business/general software
£129 + VAT.
Limited offer-ring now
SANYO 550/555 COMPUTERS
Micropro Wordstar, Calcstar etc at no extra cost! **Printers:** Epson, KDC, Juki, Kaga, Daisystep 2000 etc.
Monitors & Recorders.
Call STEVENAGE (0438) 812439
anytime for hardware/software lists
Mail Order and **Access** facilities
ANGELA ENTERPRISES
4 Ninnings Lane, Rabley Heath, Welwyn, Herts AL6 9TD.

AMSTRAD accessories, video and power extension lead:- £9.95 stereo amplifier, connect two speakers for stereo:- £15.95 inclusive. Amware, 15 Heath End Road, Flackwell Heath, Bucks, HP10 9DT.

TO FILL THIS SPACE

PHONE CAROLINE ON 01-437-0699

FLOPPY DISKS

AMA

authorised **APRICOT** and **SANYO** dealers

FLOPPY DISKS

	DYSAN*	3M	NASHUA	SONY
SS40	: 17.00	14.50	13.15	**3.5"**
DS40	: 24.50	18.00	16.50	↓
SS80	: 24.50	21.00	17.45	34.50
DS80	: 29.50	23.50	18.95	45.50

*Special offer **DYSAN** branded disks deduct £1.50. Prices per box of 10 excluding VAT. **POST FREE** discounts for **any** larger quantity. £2.00 extra in "SEE 10" Library case. Send C.W.O. or write for full list of supplies to:

A. M. A. COMPUTER SYSTEMS AND SUPPLIES
Dept F. 8 Glebe Street, Beeston, Nottingham NG9 1BZ.
Tel: (0602) 255415.

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

COMPUTER REPAIRS IN SOUTHEND
SPECTRUM.BBC.ELECTRON.
CBM 64. VIC 20
Minnie Micro Electronics
12 Eastern Esplanade, Southend
Tel: 0702 62033/615809
Open 7 days a week

SOFTWARE

VIC-20 USERS
A renumber facility for your basic programs requiring no memory expansion. Cassette £2.00 including post and package.
Dept 2, Bolling Software, PO Box 42, Bradford, West Yorkshire, BD4 7UD

FUZE
FUZED
DEFUZED
CONFUZED
CONFUZION

COLOUR GENIE OWNERS Quiz Game £2.99 — Coming Soon — Quiz Master £4.99. Fires of Mordor, 41 Pexwood Road, Todmorden, Lancs.

FOR SALE

TRS 80 MOD III 48K dual disk drive, with profile III + S/scripsit, visicalc, p/ledger s/ledger, payroll, assembler, s/utility + business graphics, g/ledger, hyper zap, hypercross, many more £700. Tel: 0346 32683.

XEROX COMPUTER
with Diablo Printer, Line-Feed plus stationery etc, plus £5000 of software which includes a full accounts package — an absolute bargain at **£2750**!
TEL: 01-979-5889

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).Please state classification and post to **COMPUTING TODAY, CLASSIFIED DEPT., 1 GOLDEN SQUARE, LONDON W.1.**

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.

COMPUTAMART

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

CHESHIRE

Computer Junk Shop

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

ESSEX

ESTUARY
PERSONAL COMPUTERS

Complete range of Acorn products, peripherals and interesting add ons. BBC repairs and spares.
Educational, Business and Leisure Software.
318 CHARTWELL Nth, VICTORIA SHOPPING CENTRE, SOUTHEND-ON-SEA, ESSEX. TEL: (0702) 614131.

HERTFORDSHIRE

NEWBRAIN & SANYO
HARDWARE & SOFTWARE

Printers, Epson, KDC, 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

LONDON

LEABUS
legal and busines 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.

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.

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, Memo-tech, 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.

SCOTLAND

VICTOR MORRIS GLASGOW

TANDY TRS 80, VIC 20, VIDEO GENIE, APPLE PANASONIC, CUMANA, EPSOM ETC.
340 Argyle Street, Glasgow G2: 041 221 8958

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.

CLASSIFIED ORDER FORM COMPUTAMART

Please include my business details in the next available issue of Computing Today:

Business Name: ....

Address: ....

....

....

ONLY £17.50!

Details: ....

....

....

....

Tel. No.: ....

Open Hrs: ....

Contact (Office Use Only): ....

Post To: Computamart, Computing Today, 1 Golden Square, London W1.

AVAILABLE FOR COMMODORE 64 WITH DISC DRIVE

Disc Drive?

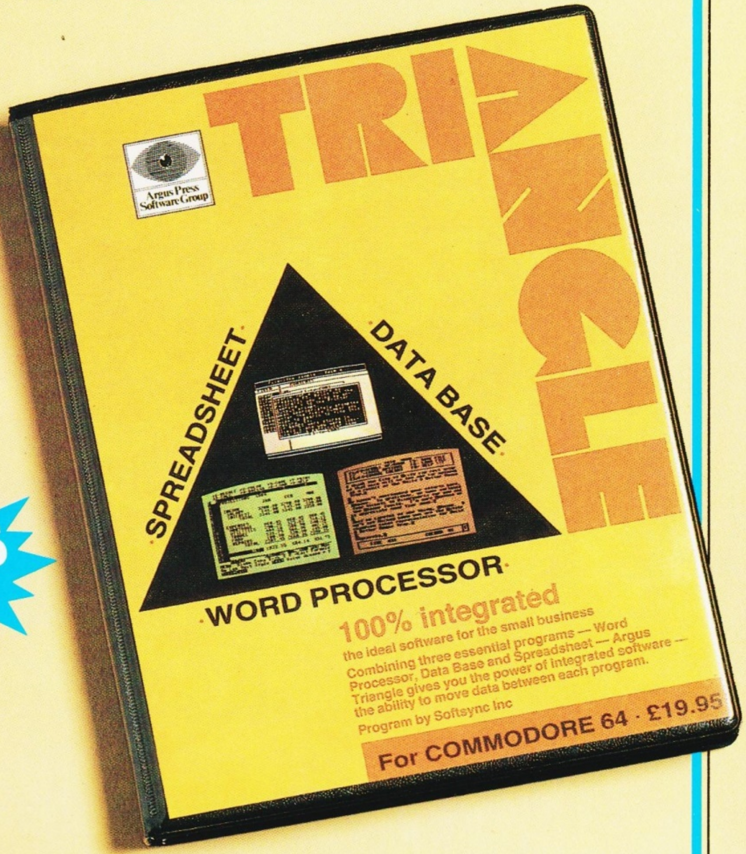
DOMESTIC DRIVE BUSINESS DRIVE

Realise the full potential of your disc drive with Triangle.

A powerful, fast and totally integrated suite of programs — WORD PROCESSOR, DATA BASE and SPREADSHEET.

Keep accounts, set-up household and business cash flow projections, store mailing lists.

All for only **£19.95** RRP



- **Three totally integrated programs**
- **Write and edit text easily**
- **Store your records, lists and inventories**
- **Answer your financial "What If's"**
- **Exchange data between programs**
- **Tutorial and on-screen "Help" windows**
- **100% machine-code for fast response**
- **Comprehensive manual that's easy to follow**

CREATIVE DRIVE

Stretch your mind and your disc drive with Activity Centre.

An integrated, stimulating and wide-ranging toolkit. Write, draw, animate drawings even compose music.

Widen your creative skills, write and illustrate reports for school or at work. Whether you're 7 or 70 years, you'll find Activity Centre great fun.

The price? It's a breakthrough at **£14.95** RRP

- **Write and illustrate text**
- **Paint and animate your pictures**
- **Compose your own music**
- **Save and copy images**
- **Large picture library memory**
- **Music fanfare library memory**
- **Easy to use manual**
- **100% integrated**

Argus Press Software Group

ARGUS PRESS SOFTWARE LIMITED
Liberty House, 222 Regent Street,
London W1R 7DB. Telephone 01 439 0666

AVAILABLE FOR COMMODORE 64 WITH DISC DRIVE