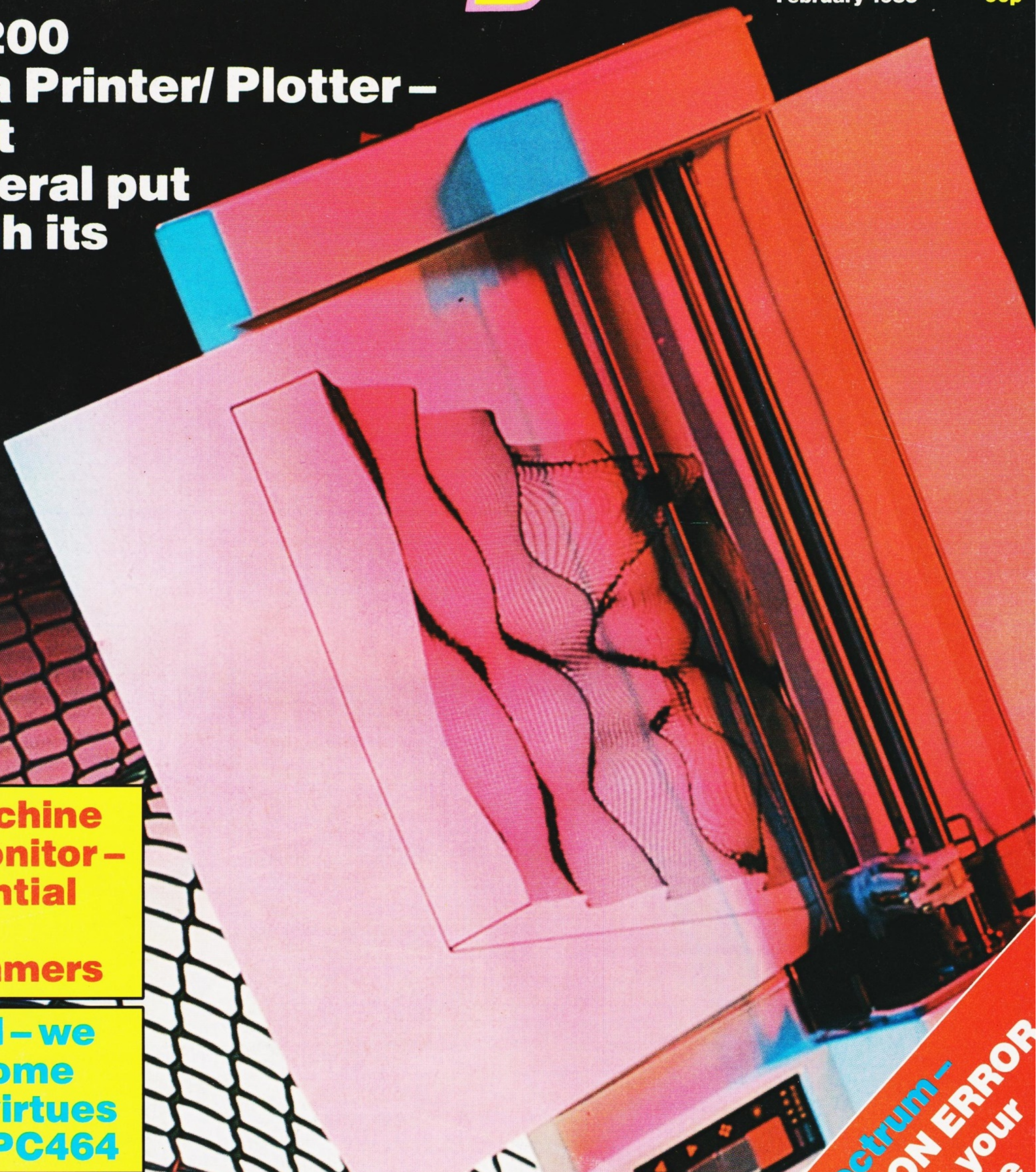
AN ARGUS SPECIALIST PUBLICATION

Computing today

February 1985 90p

The £200 Sakata Printer/ Plotter – budget peripheral put through its paces



BBC machine code monitor – an essential tool for programmers

Amstrad – we reveal some hidden virtues of the CPC464

Dragon – 6809 interrupts explained

Spectrum – add the ON ERROR facility to your machine

BACKNUMBERS

MARCH 1983
Colour Genie revi SOLD OUT son HX-20 review, PEEKing th am, Into Atari's BASIC, Terminol translated.

APRIL 1983
Froglet on the BBC Mic 1251 hand-held review, Valley V SOLD OUT s, Galaxy reviewed, Micro Data Lower case UK101.

MAY 1983
Spectrum Boo SOLD OUT Oric-1 Review, Going FORTH A piter Ace review.

JUNE 1983
Interrupt handling P ulation on the Spectrum, B SOLD OUT RS232 Blues, Lynx review, Inc

JULY 1983
Atari renumber, 16- SOLD OUT , Bomb-proof Tandy, Olive 0 review, Ikon Hobbit tape drive r w.

Computing today
MACHINE CODE MUSIC ON THE APPLE: 6502 routines can be adapted for other systems
In-depth review of the Electron – a quantum jump for Acorn?
Hi res graphics on the Commodore 64 explained
Full screen editor for the TRS-80
Plus programs for the Spectrum, Nascom
Monitor review for the Spectrum
MULTI-TASKING ZX81-FORTH – Can it do the job?
Sounding off on the Commodore 64
Getting to grips with Epson printer graphics
Debugging random number software with non-random sequences
Two useful utilities for PET programmers
The chips are down when you play stud poker against the BBC Micro
If you're baffled by machine language you need our new series on Easycode
Hardware project and software routines for a Spectrum Centronics interface
DIY arcade games on the Spectrum – now there's SCOPE for improvement
MEMOTECH'S MTX500 ON THE TEST BENCH: A black pearl from the peripheral people
Learn how to pull machine code apart with our BBC OS1.2 ROM disassembly feature
Have a Brain Storming session on your CP/M system
More of the same – Recursion on four Apple languages

AUGUST 1983
Speeding up the S SOLD OUT emier Dragon disc drive, Sord w, BBC String Store, Planetfall.

SEPTEMBER 1983
FELIX knowledge sho SOLD OUT ware protection, Torch disc pac Backgammon, Dragon character g utor, Three Tandy computers.

OCTOBER 1983
Slingshot game, Sharp MZ-700 review, Sharp MZ-3541 review, Z80 Disassembler, A better TRSDOS, Improved VIC-20 editor.

NOVEMBER 1983
BBC Word Processor SOLD OUT T review, Laser 200 review, Wri entures, Learning FORTH Part 1 tape append.

DECEMBER 1983
MIKRO assembler review, Getting More from the 64 Part 1, Adventures part 2, Curve-fitting, BBC Touch Typing Tutor.

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.

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 Reader Service

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.

Editor:
Don Thomasson

Editorial Assistant:
Jamie Clary

Advertisement Manager:
Malcolm Wynd

Advertisement Copy Control:
Sue Couchman,
Lynn Collis

Chairman:
Jim Connell

Origination by Design International.

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

Computing Today is normally published on the second Friday in the month preceding 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 rates: UK £15.00 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).

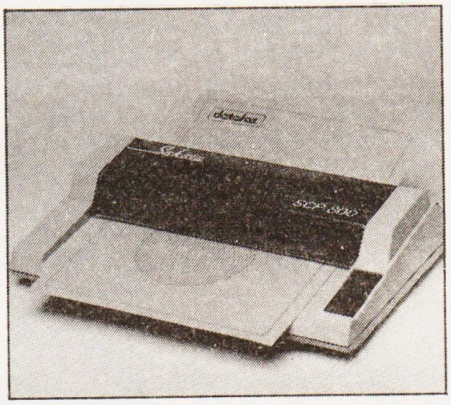
Computing Today is constantly on the look-out for well written articles and programs. If you think that your efforts meet our standards, please feel free to submit your work to us for consideration.

Potential contributors are asked to take note of the points raised in our Program Submissions page, which can be found on page 16 of this issue.

CONTENTS

VOL 6 NO 12 FEBRUARY 1985

EDITORIAL & ADVERTISEMENT OFFICE
No. 1, Golden Square, London W1R 3AB.
Telephone 01-437 0626. Telex 8811896.



Software News

INNOVATIVE SOFTWARE

from the professionals

SANYO 550/5 SOFTWARE

Cashman — Very colourful arcade game for Sanyo.		£ 26.00
DC-10 — Flight Simulator. Over 30 radio aids.		£ 26.00
Demon Seed — Another excellent arcade game.		£ 26.00
DS-DOS — Double sided DOS (1.25)		£ 37.00
DS-DOS80 — Quad DOS 80 track 9 sector (1.25).		£ 39.00
DS-DOS PLUS — Quad DOS 80 track 10 sector (1.25 or 2.11 state)		£ 39.00
Emperor — A full size wargame from Roman times.		£ 26.00
Freeze Frame — Text & graphics screen dump, colour too.		£ 32.00
French Tutor — Just that.		£ 26.00
Grafiti — Computer assisted graphics. 256K.		£ 35.00
Joystick — de Luxe quality	— 2 button	£ 37.46
	— 3 button	£ 46.18
Joystick Extender Cable		£ 14.00
King Arthur — 20 minute wargame.		£ 18.00
Major Motion — A fast road battle arcade game. 192K.		£ 26.00
Master Graph — Even plots 3-D curves.		£ 62.00
Maz — A fantastic four part maze game.		£ 20.50
M-DISK — Extra disk in RAM. 256K.		£ 26.00
MI-KEY — Re-define the whole keyboard or part of it.		£ 27.00
MI-TERM — Communications terminal program.		£ 44.00
Mighty Mail — The best mailing list handler available.		£ 82.50
Mysterious Adventures — A series of 10 classic Adventures. From damsels and dragons to space travel!		
Any 1		£ 18.00
Any 3		£ 45.00
All 10		£120.00
Personal Money Manager — Personal finances handler.		£ 40.00
Picasso — The ultimate Paint program.		£ 77.00
Quick & Simple — Very simple database.		£ 39.00
Soft Spool — A software print spooler. 192K.		£ 26.00
Solitaire — Patience, Poker Squares, Blackjack, etc. Terrific graphics. 192K.		£ 29.00
Super Calc III Zaps — Convert the spreadsheet to the Sanyo.		£ 24.50
Super DS-DOS — DS-DOS plus Utility Disk No. 1.		£ 62.00
Super Zap — Full disk access and edit.		£ 37.00
Thunder Chief — A shooting, bombing arcade game. 192K.		£ 26.00
Utility Disk No. 1 — Collection of utilities.		£ 37.00

All programs supplied on a single sided formatted disk for 128K, unless mentioned, and compatible with colour monitor.
DOS's require ownership of either MS-DOS 1.25 or 2.11.

All prices are VAT exclusive. P&P 75p per program.

TEL: [0424] 220391/223636

TELEX 86736 SOTEX G

MOLIMERX™ LTD
A J HARDING (MOLIMERX)
1 BUCKHURST ROAD, TOWN HALL SQUARE, BEXHILL-ON-SEA, EAST SUSSEX.

Send 24p A4 sized SAE for full list.

TALKING SHOP

Don Thomasson

AN ARGUS SPECIALIST PUBLICATION
Computing today
February 1985
80p
The £200 Sakata Printer/ Plotter – budget peripheral put through its paces
BBC machine code monitor – an essential tool for programmers
Amstrad – we reveal some hidden virtues of the CPC464
Dragon – 6809 interrupts explained
Spectrum – add the ON ERROR facility to your machine

The King is dead — long live the King! Well, two Kings, to be accurate. I do the interesting stuff, and Jamie Clary does the hard work!

You will have seen my name before, at the head of various articles, perhaps, but names don't mean a lot. At the PCW Show, I was amused at the reaction of some people who connected name and person for the first time. They had not visualised a tall, white-bearded gent of sixty-five. Some were kind (?) enough to say that from my articles they had envisaged someone a lot younger . . .

Once somebody asked me why I had decided to go into computing when I started work, and was a little shaken to learn that computers didn't come along for ten years after I left school. Those ten years I spent doing radio work, then I freelanced for seven years, after which five years went by in servo-engineering. I had heard of computers, but knew very little about them. I was nearly forty when my boss asked me to find out why the new Deuce computers we were having installed were still not working. To my surprise, I was able to produce answers, and that convinced me that computers were the way to go.

Computing today
The £200 Sakata Printer/ Plotter – budget peripheral put through its paces
BBC machine code monitor – an essential tool for programmers

The subsequent twenty-five years have seen me move gradually from computer hardware to computer software, though one foot has remained firmly on the hardware side. That has been valuable, because there is always a need for links between hardware and software. Too many people sit on one side of the fence and blame those on the other side. Sitting on the fence may be uncomfortable, but it gives you a better view.

All of which will inevitably influence the way I do my new job. I will ask for a fair balance between software and hardware. If you consider hardware a bore, you still need to know how it influences the performance of your programs. If you wince away from software, remember that your hardware might be pretty useless without it. My very first article for *Computing Today* was "Getting into Print', which dealt with the interaction of Printers with driving software, and that expressed my views fairly precisely.

What changes can you expect in the content of *Computing Today*? Well, you have here an Editorial for the first time in a while, and you will also find a correspondence page. These express my desire to keep in touch. We can't go on putting out the sort of thing you want if you don't provide some feedback.

You will also find a review with a difference. Most reviews of new machines are necessarily superficial, and quite a lot of important facts emerge too late for inclusion. On that basis, the AMSTRAD CPC464 looked quite a nice machine, but on closer inspection a number of very slick ingenuities have emerged. The solution to the bank switching problem is particularly neat. We give you some insight into how all this works. If you are ambitious, you will want to make use of the facilities described. If you are not, you may still be interested to know how it is done.

Computing today
The £200 Sakata Printer/ Plotter – budget peripheral put through its paces
BBC machine code monitor – an essential tool for programmers

Broadly, the objective will be to set a standard that is a little above the run-of-the mill popular journal, while avoiding the trap of aiming too high and becoming too pedantic or too abstruse. Our role, to some extent, will be to act as interpreters. We will look at Artificial Intelligence, but in a slightly cynical way, cutting through the jargon to get at the essentials. We will not accept glossy handouts as evidence of extant hardware, and we will try to encourage manufacturers to supply more useful data. That doesn't mean that we will eschew gossip completely, but it will be labelled 'gossip', and kept separate from hard information.

Computing today
The £200 Sakata Printer/ Plotter – budget peripheral put through its paces
BBC machine code monitor – an essential tool for programmers

If this line of action appeals to you, put in your subscription order right away.
And watch this space . . .

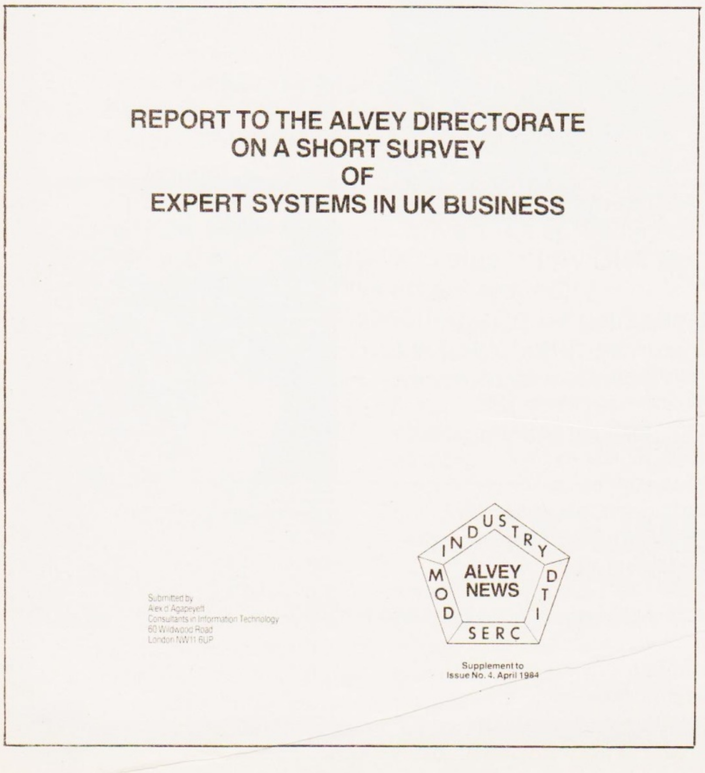
NEWS

HAND-HELD HUNTER — YET MORE APPLICATIONS

Two more extraordinary uses have been found for the Husky Hunter hand-held microcomputer, a machine designed to cope with the harshest of environments.

The British built Hunter is now being used to replace data-loggers in theodolite survey applications, in order to assist land surveyors to construct three-dimensional ground models of sites throughout the UK. Equipped with a land-survey software package developed by **Eclipse Associates** of Newport Pagnell, the Hunter carries out all of the normal tasks performed by conventional data-loggers whilst offering a number of important additional facilities for computation, collation and sorting of data. And, the Hunter's RS232 interface permits surveyors to transmit data from the field, back to a computer at headquarters for subsequent analysis.

Meanwhile, software developed by the Greater London Council for use with the Hunter is bringing 'untold benefits' ***(why won't they tell us? — Ed.)*** to elderly Londoners. A group of 30 home-help supervisors, occupational therapists, and volunteers are using the Hunter to calculate elderly persons benefits to ensure that they are receiving their maximum entitlement. Sceptics who see this as a 'sledgehammer to crack a nut' should bear in mind that more than £700 million goes unclaimed each year, and as Anne Hollows, leader of the GLC's Welfare Rights Campaign, commented: "The computer substantially increases the access of some of the poorest elderly people in the community to vital information and support in claiming their basic rights." Incidentally, any reader who disagrees with the fundamental precept that London is a harsh environment, is invited to write to us for a fuller explanation!



REPORT TO THE ALVEY DIRECTORATE
ON A SHORT SURVEY
OF
EXPERT SYSTEMS IN UK BUSINESS

Submitted by
Alex d'Agapeyeff
Consultants in Information Technology
60 Wildwood Road
London NW11 6UP

INDUSTRY
MOD
DTI
SERC
ALVEY NEWS

Supplement to
Issue No. 4, April 1984

EXPERT SYSTEMS 'NOT INHERENTLY COMPLEX'

A report recently submitted to directors of the ALVEY Project — the Government funded AI think-tank — concludes that "It is necessary to correct the impression ... that Expert Systems are inherently complex, risky and demanding."

The report, which contains the findings and conclusions of a short survey of expert systems in UK business, suggests that simple expert systems are practical and are being implemented **now,** despite the fact that such systems are regarded with some awe by those who use them. Perhaps the most interesting finding of the survey though, is that "The importance of these systems is indicated by the extent of secrecy adopted by user companies which may go beyond the national interest" — a sentence of gobbledegook that reveals a certain dissatisfaction with those 'user companies' unwilling to share their findings with the rest of the UK.

BEEB GETS A MOUSE

A comprehensive computer art program, which can be used for the preparation of illustrations, architectural and engineering drawings, teachers' worksheets and general design work, is just one of the features provided as standard with the first Mouse to be marketed specially for the BBC Micro.

Developed by **Advanced Memory Systems Limited**

KAYPRO OF THE ANTARCTIC

A team of explorers set out this month for the Antarctic, their mission being to retrace the footsteps of Scott's famous 1910 expedition to the South Pole.

A two-man team will attempt to reach the Pole on foot next winter without the benefit of modern aids, but one piece of equipment that has proved essential for the voyage is that untiring workhorse, a microcomputer.

They have chosen a Kaypro 4 personal computer, a portable model specially designed to be carried around which can work off both mains or batteries. The tough steel housing opens out to reveal a 9" screen, twin disk drives, Z80 based processor, and professional style keyboard.

Joanna Gurr, the Expedition Procurement Manager, explained how she set about deciding on her choice: "Both my father and my brother are computer consultants, and when I asked for their advice they immediately suggested Kaypro.

But why take a micro on an expedition?

"The main thing I will be using it for is word processing — correspondence with sponsors, press releases, user reports etc. I wanted a fast efficient machine and the Kaypro is brilliant for that.

"In addition to my requirements, I am sure that some of the others will find uses for it, such as in the calculations for the navigation."

(AMX) in conjunction with **Elliot Software Limited,** the new Mouse is an advanced opto-mechanical device which makes available to the BBC Micro user facilities, which hitherto could only be utilised on the more expensive computers.

AMX believe that their new product will be welcomed particularly by home users and by schools, where the BBC Micro is one of the market leaders. While the Mouse can be used with ordinary programs to replace the cursor keys, with the specially written software provided, it turns the BBC computer into an altogether more friendly device, helping beginners, but also adding new dimensions for the accomplished user.

The AMX Mouse can be used with any BBC Model B computer fitted with operation system 1.2, and it can be simply plugged into the user port, drawings its power from the computer.

The AMX Mouse and software package has a recommended retail price of £89.95 including VAT. The company expects to add further software in the future, the first of which, a desk top package, should be available in early 1985.

For further information contact:

Advanced Memory
Systems Ltd.
Woodside Technology
Centre,
Green Lane,
Appleton,
Warrington WA4 5NG
Cheshire

Telephone: (0925) 62907

EPROMs WITH PAGE ADDRESSING

Just announced by Rapid Recall are two new Intel 512K bit EPROMs with provide high-density firmware storage for a wide range of applications. The devices save circuit board space by allowing more storage to be accommodated in a standard 28-pin package.

Known as 27512 and 27513, these UV erasable PROMs are produced by Intel with their advanced HMOS II-E technology which ensures outstanding reliability.

The 27513 features innovating page addressing and is organised as four pages of 16K 8-bit words. This brings 64K bytes of storage capacity to existing 128K EPROM-based designs, also to 8-bit microprocessor/micro-controller systems with up to 64K bytes of total addressing capability. The other EPROM, 27512, uses the conventional addressing method.

Other features of the 27513 include a 250ns access time; low power consumption (125mA active, 40mA standby); TTL, CMOS and direct 27128A compatibility; and Automatic Page Clear — the page select latch is automatically cleared to page 0 upon system power-up.

Two-line control and industry-standard 28-pin packaging are also common features (on all Intel high-density EPROMs), allowing easy microprocessor interfacing when upgrading, adding or choosing between non-volatile memory alternatives.

NEW MS-DOS COMPUTER

The newest addition to Tandy's range of microcomputers is the Tandy 1000 Personal Computer. It provides capabilities compatible with and equivalent to the IBM PC Computer plus graphics and sound enhancements, and is available with the new Deskmate integrated software package, the MS-DOS operating system and Basic.

The design and 360K floppy disk format of the Model 1000 allow it to run 'off-the-shelf' IBM PC software including Lotus' 1-2-3, Ashton-Tate's DBase III and Multimate International's Multimate, and it is fully compatible with the widest selection of software available.

ACT AND VICTOR GO SEPARATE WAYS

This month Victor Technologies Inc. will be launching their U.K. subsidiary — VICTOR TECHNOLOGIES U.K. LTD. A substantial amount of funding has been allocated to the U.K. subsidiary enabling the formation of a strong management team with proven ability and marketing expertise.

For the last three years **A.C.T.** (Applied Computer Technologies PLC) has distributed and supported Victor products in the U.K. High levels of Victor product sales, notably the Sirius microcomputer, are the result of a strong marketing effort by A.C.T. but at the end of December 1984, A.C.T. relinquished their exclusive distribution rights in the U.K.

MicroAPL FOR THE SINCLAIR QL

MicroAPL, the UK company which specializes in the application of the high level programming language APL to microcomputers, is launching a new keyword version of the language to run on the Sinclair QL.

For the very first time this simple yet powerful programming language, traditionally offered only on mainframes and large powerful supermicros, is available to the mass market on a popular, inexpensive microcomputer, making APL easily accessible to both the busy executive and the home computer enthusiast.

This specially adapted version of APL which dispenses with the usual APL programming symbols can be seen for the first time in public at the **Which Computer? Show** in January.

MicroAPL will also be exhibiting a range of micros that support APL including the IBM PC and their own British manufactured supermicro the Spectrum. The Spectrum can support up to 20 simultaneous users, has unlimited hard disk capacity and is expandable up to 14 Mb of RAM. Its APL is IBM VSAPL compatible and includes over 50 enhancements to the standard.

For further information:

Karen Hurl
MicroAPL Limited,
Unit 1F, Nine Elms
Industrial Estate,
87 Kirtling Street
London SW8 5BP.

Tel: 01 622 0395

NEW PRODUCTS FROM TANDY

Tandy's new Xenix multi-user system, the Model 6000 and four new printers are to be launched at the 1985 Which Computer? Show.

The Model 6000 desk top microcomputer system replaces Tandy's 16B to offer simultaneous job handling for up to 51% users on a new version of the Xenix operating system called System 3.

Designed around a 68000 microprocessor the Model 6000 offers 512K of internal memory which is expandable to one megabyte. Two RS-232C serial communications interfaces and a parallel interface allow expansion with a variety of peripherals. Including a self-contained floppy disk and hard disk it offers a total of 50 megabytes.

The **DMP 105** is a full feature 80 column, 80 character per second printer for the Tandy Colour Computer or any Tandy CPU. Price £169.95.

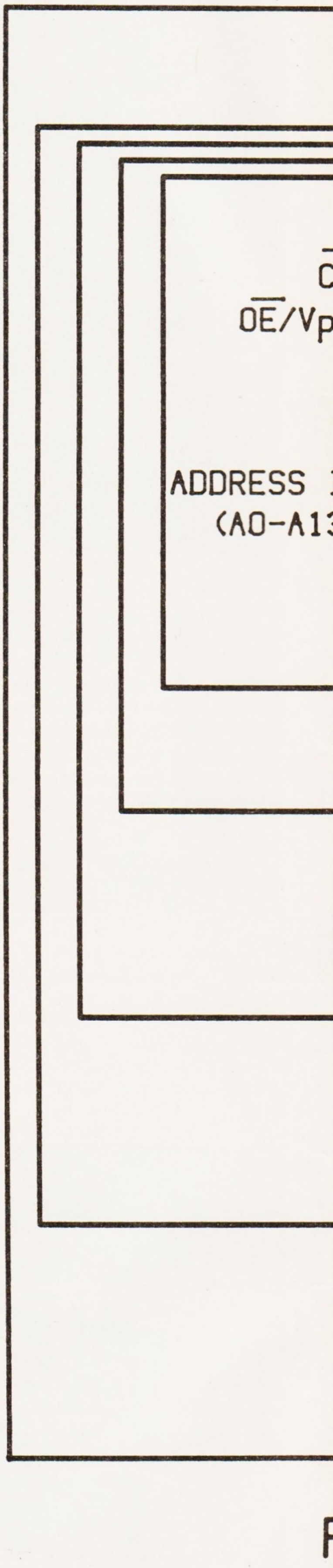
The **TRP100** Thermal ribbon transfer printer for use with plain paper is battery operated and fully portable. It produces 50 characters per second and operates at only 50 decibels. Fully IBM PC compatible it retails at £229.95.

The **DMP430** dot matrix printer is a 15 inch high performance IBM PC compatible printer. It has an 18 pin printhead and produces 180 characters per second. Price £599.00.

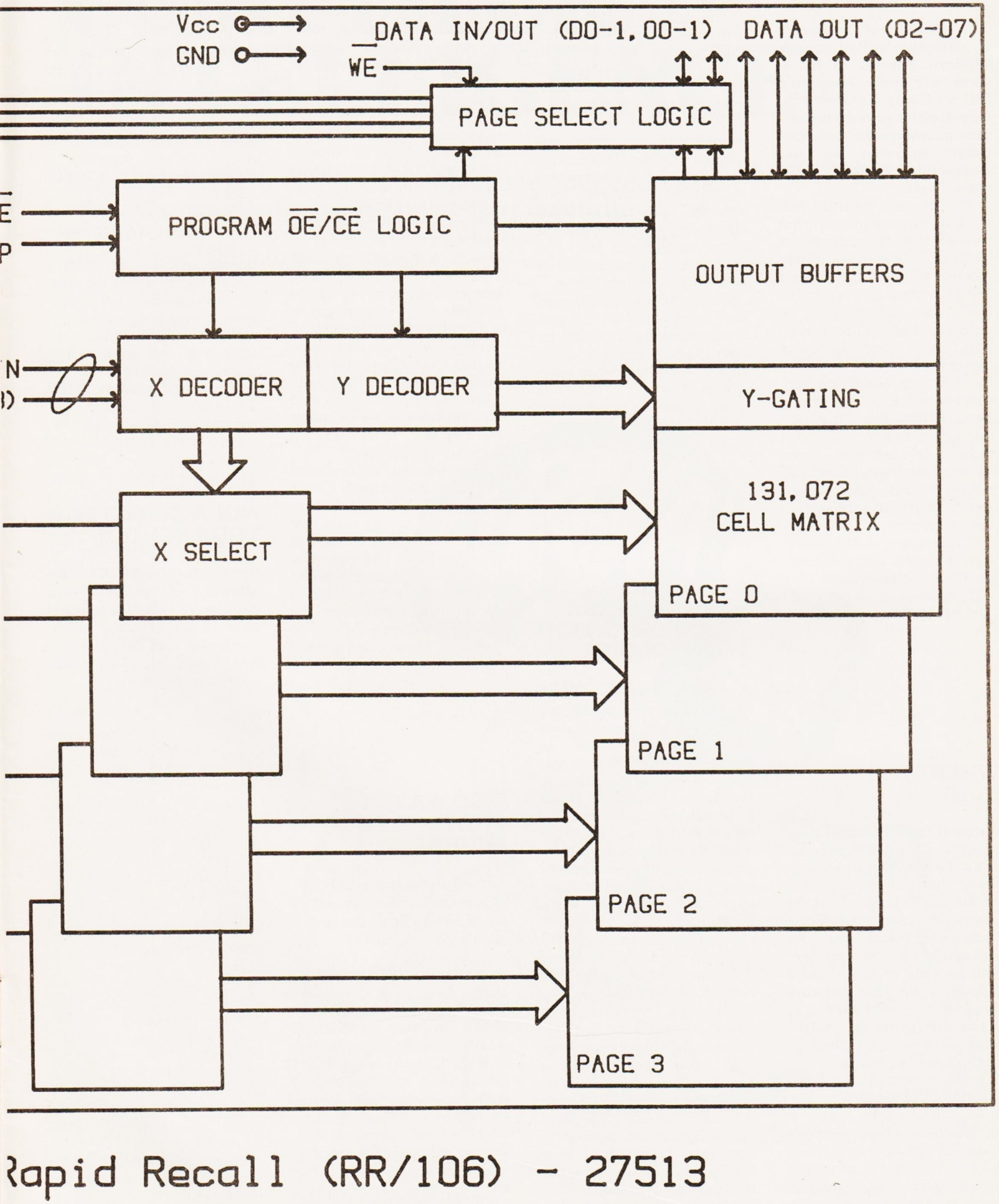
For further information:

Danusia Hutson,
Sheridan Communications
Limited,
15 Greenfield Crescent,
Edgbaston,
Birmingham B15 3AU

Telephone: 021 454 5418



Function diagram of the new 512K bit Eprom from Rapid Recall



SOFTWARE COPYRIGHT – PRIVATE MEMBER'S BILL

The decision of Conservative Member of Parliament, William Powell, to introduce a Private Member's Bill on Computer Software Copyright marks an important milestone in the campaign initiated by the Federation Against Software Theft.

Mr Powell, a barrister and MP for Corby, came sixth in the recent Private Member's Ballot and, as a result, is highly placed in the forthcoming Parliamentary timetable. His Bill is non-contentious in its nature and with all-party support, has a good chance of becoming law.

"A Private Member's Bill" commented FAST chairman Donald MacClean, "is a solid step forward in our campaign to fight software theft, which already costs the industry £150 million each year and poses a direct threat to jobs, investment and innovation.

"Our meetings with Government Ministers make us confident that William Powell's Bill will receive backing from Government departments."

WHERE'S THE FUN?

A new trial service for home computer users has been launched by the magazine Your Computer, supported by British Telecom East. By simply making a telephone call, many micro computer users who possess a modem will be able to download a selection of the software programs published in Your Computer magazine, and save a good deal of "laborious keying in."

Initially, the service is available to BBC Model B and Spectrum 48K owners but it is proposed to extend to other makes of computer in the near future. One of the programs which is immediately available for both computers is a communications package entitled "Dialsoft", which will enable users to communicate with each other via the keyboard and transfer data or programs.

The unique feature of this new service is that it is available to home computer users for the cost of a short telephone call (eg: a 6K byte program should take about one minute with a 1200 bit/sec modem) — no membership or subscription fees are required. This is made possible by the development of a low cost hardware and software package, which allows up to 40 users to download concurrently from one equipment and is thus ideal for short term mass distribution.

Details of the telephone numbers to call, and for the programs currentlyt available for downloading, are now available on a new recorded information service — Microline —which can be accessed on Colchester 8068 (STD code 0206).

A PRACTICAL INTRODUCTION TO INFORMATION TECHNOLOGY

A new 1-week course designed for managers and workers in small and medium-sized firms is being run at the Microcomputer Advisory Centre, at London's South Bank Polytechnic, near Waterloo. 40% of the time will be spent working at microcomputers on business applications, to give people a real insight into the use of such machines.

Some smaller firms are still reluctant to automate their systems, although the outlay may be modest in terms of the potential benefits. The price of this course is being kept low with the help of the European Social fund and the Greater London Council. The charge will be £100 per student. It is hoped that the modest price will encourage a new group of employers to introduce their staff to the computer age. For information contact Jack Flatau, on (01) 928 8989 ext. 2410.

WH SMITH COMPUTER CREDIT PLAN

W H Smith is now offering credit facilities to customers buying computers at its High Street Computer Shops.

Credit is available for purchases of computers, software and accessories to the total retail value of between £400 and £2,000 — 10% of which is taken as a deposit at the time of purchase.

Products wanted on credit by customers need to include an item of computer hardware. The balance is repayable in monthly instalments of 12, 24 or 36 months — depending on a customer's requirement.

"This will help our more serious customers with their purchases, particularly with Christmas approaching", said Mr John Rowland, W H Smith's Merchandise Controller for Personal Computers.

The scheme is operated through the Mercantile Credit Company Limited. W H Smith has more than 50 computer shops, including one at Waterloo main line station in London, where this scheme is in operation. It is not available at the two W H Smith computer shops at Heathrow Airport or the 200 computer departments.

TWO WAYS TO PROTECT YOUR MICRO

Although there is much talk of computer data-security, what do you do if somebody nick's your micro?

Bexleyheath-based mailing and security specialists, **Versapak Ltd.,** have developed **Compusafe** — a robust, lockable computer workstation which can be bolted to the desktop. Constructed from 14-gauge steel with two sliding shelves, the workstation is large enough to hold the computer/keyboard and associated disc-drives, although the monitor would have to be bolted to the top.

Compusafe is supplied stove enamelled in a textured finish to suit the BBC Micro, but other colours are available to suit the Spectrum, Electron, Commodore and other popular micros. Price £80.

For subscribers to that popular theory 'the barn door should be closed after the horse has bolted', **Insure Data Limited** are offering three grades of insurance cover to computer owners. The three types of cover — Bronze, Silver and Gold — range from £95.00 (Bronze) to £160.00 (Gold).

This talk of computer theft reminds me of an unfortunate burglary that occurred at a polytechnic that providence forbids me to name. One of the department's PETs went astray one evening, but it is generally reckoned that the thief's conscience got the better of him, partially at least. The PETs case was found the following morning — minus the built-in monitor and main PCB.

It's a funny old world...

WIN A SANYO FOR YOUR SCHOOL

Sanyo Marubeni (UK) Ltd is offering all schools and colleges the chance to win a new 16 bit MBC555 micro computer and colour monitor in an exclusive free-to-enter competition.

The competition will be run in conjunction with the 1985 Which Computer? Show, to be held at the NEC, Birmingham between 15th and 18th January.

Open to both the teaching staff and student members of any school or college who visit the Sanyo stand, the competition is restricted to one entry per person but no limits will be imposed on the number of entries made on behalf of a particular school or college.

To enter the competition to win the Sanyo computer and colour monitor, a simple entry form — obtainable from the Sanyo stand — should be fully completed with details of entrant and his corresponding school or college.

Completed entry forms should be placed in the drum on the Sanyo stand. The prize draw will be made by a Sanyo representative on the last day of the exhibition.

COMPUTER INTELLIGENCE

Don Thomasson

It's very easy to attribute intelligence to your computer when it consistently beats you at your favourite game or astounds you with answers to complex problems. But is this really intelligence?

The concept of the 'intelligent computer' attracts a lot of attention, with many erudite gentlemen diligently seeking ways and means of creating such a device and several pundits who are always ready to appear on television to explain how difficult it all is. Perhaps a broad-based look at the subject from a rather naive and cynical point of view would be useful.

First, we must decide what we mean by the word 'intelligent' in this context. We talk of 'intelligent peripherals', when we merely mean that they can accept and obey commands calling for complex sequences of action. Such an ability may put them on a par with other computing devices, but does not necessarily make them genuinely intelligent.

The vast majority of computer programs allow a rational response to be derived from external stimuli within a predetermined range. That, again, is not intelligence, since it is achieved by programming in rules which determine the response produced. No decisions are needed, other than those determined by the rules. The program must be able to recognise a given set of situations and take appropriate action, but can only do so if the situations encountered are within the pre-planned range.

It is tempting to attribute intelligence to systems which can learn by experience, acquiring an ability to cope with situations not planned for in the original program, but this can be misleading.

GAME LEARNING

A simple two-dimensional Noughts and Crosses program can be adapted so that winning and losing situations can be recognised and stored as a guide for subsequent play, but a vast amount of storage is needed. In theory, there are 362880 possible play sequences, though the critical stage of the game can probably be covered with only 3024 alternatives. Even the smaller figure implies a lot of storage.

The human mind approaches the matter in a different way. First, it recognises that there are only three initial moves possible; the centre, a corner, or a side. The second move can only create twelve different combinations, where the computer would see seventy-two, because it fails to recognise that some combinations are rotations or reflections of others.

The human mind can devise a set of nine rules which will suffice to stave off defeat, and encourage the chances of victory. A computer can work to these rules, but cannot devise them on its own.

If a third dimension is added to the Noughts and Crosses game, an interesting change takes place, because there is then a subtle difference between the strategy needed to win and the strategy needed to guard against defeat. If either player can occupy the four corners of any plane within the playing cube, the rest of the plane being empty, he can force a win. (There are other possible winning situations, but one will suffice.) To achieve the essential position, however, he must mix defence with attack in carefully-judged proportions, taking particular care to avoid the creation of potential 'double-force' situations in which he is faced with two possible winning lines created simultaneously.

It does not seem feasible to formulate rules for this form of the game, and the best that can be done is to provide the computer with 'weightings' that express the relative urgency of the moves which it can make.

The key point here is that the human player is thinking ahead, playing for an ultimate position, rather than to deal with the immediate situation. He may take twenty moves to achieve his objective, so if a 'tree' approach were used, by which the machine could explore the future possibilities, the tree would have to be rather large.

EXTRAPOLATION

The analysis of possible future situations by the 'tree' method is an example of extrapolation, and the computer can only cope with this kind of process within severely limited bounds. To escape from that limitation, it needs to be able to expand its databank freely and spontaneously. The rate of expansion is likely to grow at an increasing rate, additional data collected or deduced leading to further opportunities to extrapolate.

Once again, the implied problem is one of memory size, coupled in this context with the need for a method of cross-reference from one fact or concept to another. Masses of unrelated data are of little use to the computer or anything else. So much depends on the context of the information.

PYRAMID OF LEARNING

The human mind relies on a vast amount of background information which can be used as a basis for extrapolation. This takes time to acquire. A baby begins by picking up a few essentials, such as the use of loud yells to attract attention, and then appears to make relatively little progress. One day, the complex of acquired knowledge begins to fit together, and there is a big step forward. It may show as a single word uttered clearly and with confidence, in which case a positive spate of words will soon follow. The child will glory in its new-found ability to communicate.

Communication is essential to the learning process, since it allows questions to be asked in an endeavour to fill gaps in the acquired knowledge. It has been postulated that a computer cannot begin to learn properly until it can be taught to ask spontaneous questions. It has also been suggested that the ability to ask spontaneous questions requires a reasonable degree of intelligence. This creates a deadlock situation.

However it must be remembered that there is a key difference between teaching and learning. A pupil may reject the information offered by a teacher, or may fail to understand it. The transfer of knowledge only works when both giver and receiver are co-operating. If data is fed to a computer, no benefit will accrue unless the computer knows how to use the knowledge.

THE BARRIER OF LANGUAGE

One of the most serious impediments to communication, and therefore to learning, is the complexity of human languages. Deaf and dumb children may be adjudged to be mentally handicapped, because they are unable to communicate their thoughts as freely as other children can, and find obtaining information equally difficult. Provided with alternative means of communication, they might begin to show their true capabilities. Adding the handicap of blindness produces a truly horrifying combination. An encounter many years ago with a lady who was deaf, dumb and blind is still remembered vividly. Her only contact with the world of human knowledge was through letters drawn by a finger on the palm of her hand . . .

A computer is almost in the same straits. It can handle words fed into it, but does not understand them. The words are just words, with no associations. Someone blind from birth cannot appreciate the names of colours. A computer will generate colours on command, and it would not be impossible to set up hardware which would allow a computer to identify colours, but linkage with the colour names would be difficult.

Quite simply, a computer would need most of the human senses if it were to appear even mildly intelligent in human terms.

THE BROADER VIEW

Much of the foregoing may seem to be expressive of a pessimistic attitude towards artificial intelligence, but the points made are by no means irrelevant. The biggest problem, perhaps, is one of scale. The degree of intelligence of animals is judged by the size of their brains. In terms of sheer storage capacity, the largest computer on earth would not rate very highly, and its speed of access for random data would be adjudged to be totally inadequate. If you want a computer to do properly what Eliza appears to do, responding to ordinary conversation in a fairly rational manner, you will need both a vast store and a very fast method of data recovery. On the other hand, it is possible to experiment with the principles involved with comparatively modest resources.

The difficulty here is to find a manageable objective. Something on the lines of Animal, Vegetable or Mineral might be suitable, but you would need to place a limit on the possible range of answers. In an entirely different direction, you might computerise the game of Pelmanism, which involves finding cards among a pack spread at random on the floor.

But here is a final thought — in the ordinary way, the computer is a totally obedient servant — sometimes too obedient, because it never queries your intentions if you ask it to do something silly, such as destroy the program you have just typed in. Now, a totally obedient servant can scarcely show intelligence, since that would mean doing something outside the range of the instructions which it received.

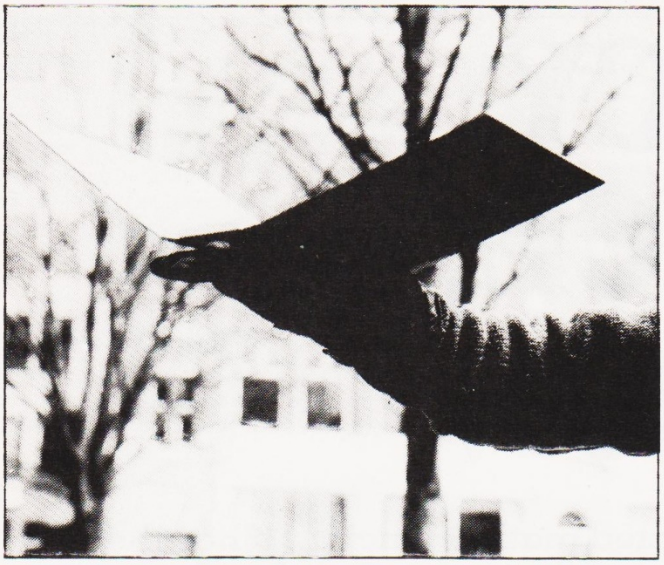
To exhibit intelligence, a computer would have to act independently of instructions given to it, not just by reporting 'Syntax Error', but by acting spontaneously. But by that we imply that the computer must have a desire to act, and that implies something resembling emotion. We ourselves can sometimes sink into a state of torpor, in which we have no wish to do anything. What lifts us out into active service is usually emotion, if only in the form of annoyance at being told that the grass needs cutting . . .

A computer with emotions is not totally inconceivable, but it is not with us yet.

CONCLUSION

On the whole, it must be concluded that artificial intelligence, in the true sense of the term, is still something of a pipe-dream. It may be contended that advances are being made towards realisation of the dream, but perhaps concept is ahead of implementation.

Nor is it by any means certain that an intelligent computer would be particularly easy to live with. Isaac Azimov has done much to suggest to us what living with intelligent robots could be like. Even a robot unable to get up and walk might sometimes be an embarrassing companion.



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 . . . .

ORIC AND SINCLAIR COMPUTERS
Oric 1 computer 48K £95 (£92) £102. Oric Atmos computer 48K £171 (£158) £168. Oric colour printer £134 (£123) £140. Oric disc drive £304 (£278) £298. New Sinclair Flat screen TV £113 (£105) £115. New Sinclair Spectrum Plus Computer with 6 free programs while Sinclair make this offer £182 (£176) £187. New Sinclair QL Computer £406 (£385) £410. Sinclair Spectrum 48K (with 6 free programs while Sinclair's offer lasts) £131 (£131) £143. Microdrive £51 (£50) £60. RS232 interface 1 £51 (£50) £60. Limited period special offer:- Microdrive + Interface 1 + 4 Blank cartridges £102 (£100) £120. Blank microdrive cartridges £5.50 (£6) £7. New standard floppy disc interface for Spectrum £102 (£92) £112. (See Cumana disc section for suitable disc drives). Interface 2 £20.45 (£20) £24. 32K memory upgrade kit for 16K Spectrum (issue 2 and 3 only) £31 (£28) £30. Spectrum printer interface with cables:- Centronics £51 (£48) £52, RS232 £35 (£33) £38. ZX printer has been replaced by the Alphacom 32 £71 (£69) £82. 5 printer rolls (State whether Sinclair or Alphacom) £13 (£16) £21. ZX81 computer £45 (£44) £54. 16K ram packs for ZX81 £28 (£25) £30.

COMMODORE COMPUTERS
Commodore C16 Starter Pack £145 (£142) £162. Commodore Plus/4 £305 (£281) £301. Commodore 64 £222 (£215) £235. Vic 20 starter pack £75 (£85) £115. Convertor to allow most ordinary mono cassette recorders to be used with the Vic 20 and the Commodore 64 £9.78 (£9) £11. Bargain package:- cassette convertor + compatible cassette recorder £37 (£38) £44. Commodore cassette recorder £43 (£44) £50. Printer interfaces for Vic20 and the Commodore 64:- Centronics £45 (£41) £46, RS232 £45 (£41) £46. Disc drive £233 (£209) £234. 1520 printer/plotter £165 (£149) £159. MPS801 Printer £235 (£220) £245. Light pen £29 (£29) £33.

ACORN COMPUTERS
Electron £203 (£209) £229. Electron joystick and printer interface £61 (£62) £69. BBC Model B £404 (£357) £387. BBC Light Pen £29 (£29) £33. 14" Colour Monitor £228 (£299) £319. Kenda double density disk interface system £149 (£131) £141. See Cumana disc section below for suitable disc drives.

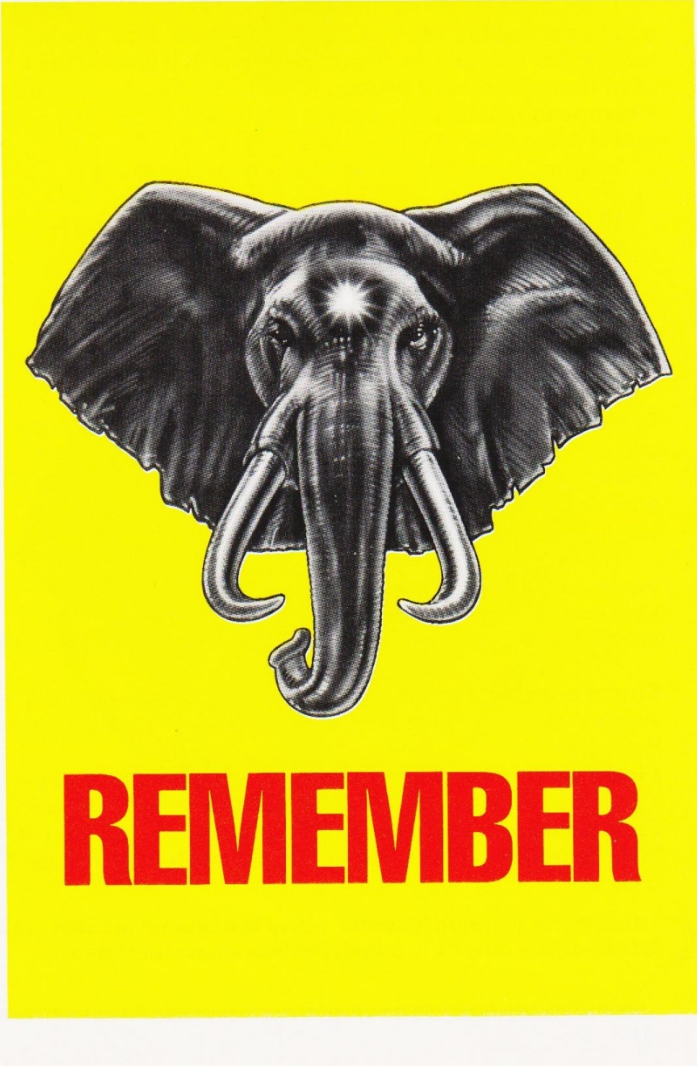
CUMANA DISC DRIVES
5.25" cased with own power supply to suit Sinclair Spectrum, BBC B and genie floppy disc interfaces. Single:- 40 track single sided £176 (£158) £178, 40 tr double sided £218 (£195) £215, 80tr ss £207 (£186) £206. 80tr ds £234 (£109) £229. Dual:- 40tr ss £299 (£280) £320, 40tr ds £395 (£353) £393, 80tr ss £372 (£334) £374, 80tr ds £437 (£390) £430.

PRINTERS
Brother HR5 £162 (£146) £170. Shinwa CTI CP80 (225 (£218) £248. Cannon PW1080A £332 (£299) £329. Epson RX80 £277 (£251) £282. Epson RX80F/T £314 (£286) £316. Epson FX80 £399 (£358) £388. Combined matrix printers and electric typewriters:- Brother EP22 £173 (£166) £186, Brother EP44 £258 (£235) £260. Smith Corona TP1 Daily wheel printer £252 (£225) £255. MCP40 Oric Colour printer/plotter £134 (£123) £140. Interfaces to run the above printers from Vic and the Commodore 64 £45 (£41) £46. We can supply interfaces to run the above printers from Sharp computers £58 (£52) £55.

UK101, SUPERBOARD AND VIDEOGENIE COMPUTERS
We still support these gorgeous machines. Please write for our free list.

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

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



DEFICIENCY, ABUNDANCE, PERFECTION

Victor Nicola

Natural numbers 1, 2, 3 . . . have fascinated mathematicians since the earliest days of civilisation. Investigation of the properties of such numbers often requires large-scale generation and manipulation of vast numeric sequences — a process which until quite recently had to be performed by hand. This article, the first in our occasional series, demonstrates how your micro can let you in on the captivating world of experimental and recreational mathematics.

In this article I shall look at some aspects of the divisibility of one number* by others. By divisibility, we mean the division of one number, say M, by another, N, where there is no remainder. For example, we say that 6 is divisible by 2, while 7 is not divisible by 3. This is because the division of 7 by 3 leaves a remainder of 1 while the division of 6 by 2 does not leave any remainder (in fact the remainder is 0). All numbers which divide a given number, N, are called divisors. Thus, the divisors of 12 are 1, 2, 3, 4, 6 and 12.

Every number can be divided by itself to give the number 1, and by the number 1 to give the number itself. Thus, every number has at least two divisors, itself and 1. If the number has no other divisors we call it **prime** and, if it has other divisors, we call it **composite.** Examples of prime numbers: 2, 3, 31, 101, . . . etc and of composites: 4, 25, 62, 111, . . . etc.

Program 1 prints all the divisors of a given number N.

```
10 INPUT N%
20 PRINT "THE DIVISORS OF ";N%;" ARE:"
30 FOR I%=1 TO N%
40 IF N% MOD I%=0 THEN PRINT I%;
50 NEXT I%
60 PRINT
70 GOTO 10
```

For those readers whose micro does not distinguish between integers and floating point numbers — drop the % sign from the variable names. The operation N% MOD I% gives the remainder after dividing N% by I%. It is equivalent to the expression: N%–INT(N%/I%)*I%.

Through the centuries, mathematicians have posed to themselves and to fellow mathematicians "ticklers" like the following: (bear in mind that until recently, they did not have access to computers and they had to do the search "by hand").

Find a number, the sum of its divisors is a perfect square. The smallest such number is 3 because 1+3=4=2 2. Can the reader find others?
Find a perfect square, the sum of its divisors is also a perfect square. Eg. 81=9 2 : 1+3+9+27+81=121=11 2. Others?
Find a perfect cube, the sum of its divisors is a perfect square.
What number (<1000) has the highest number of divisors? What are they?

*Henceforth the word 'number' refers to a natural number (eg an integer in the range 1 to infinity) unless otherwise stated.

Sometimes it is interesting to look at the divisors of a number excluding the number itself. These are called the **aliquot** divisors of the number. To find the aliquot divisors of N in Program 1, change line 30 to:

```
30 FOR I%=1 TO N%–1
```

or, preferably, to:

```
30 FOR I%=1 TO N%/2
```

(can the reader see why?)

Find a perfect square, the sum of its aliquot divisors is also a perfect square.
Find a number N, the product of its aliquot divisors is N 2.
The same as above, but the product is N 3.
Ditto, but the product is N 4.

THE SUM OF THE DIVISORS

Let us turn our attention to the sum of the aliquot divisors of a given number, N. Call this sum S(N). If we compare S(N) with N, we shall have one of the three following possibilities:

S(N) <N : in this case N is called *deficient.*
S(N) >N : in this case N is called *abundant.*
S(N)=N : in this case N is called *perfect.*

Program 2 prints whether a given number, N, is deficient, abundant, or perfect. (Ignore N=1).

```
10 S%=0
20 INPUT N%
30 FOR I%=1 TO N%/2
40 IF N% MOD I%=0 THEN S%=S%+I%
50 NEXT I%
60 IF S%<N% THEN PRINT N%;" IS DEFICIENT."
70 IF S%>N% THEN PRINT N%;" IS ABUNDANT."
80 IF S%=N% THEN PRINT N%;" IS PERFECT."
90 GOTO 10
```

QUESTIONS

Modify Program 2 to print all numbers ($1<N<100$) and indicate whether they are deficient, abundant or perfect.
What is the smallest odd abundant number?
Generate all odd abundant numbers <10,000.
Are all odd abundant numbers divisible by 5?
Find all perfect numbers <10,000. Incidentally, all these perfect numbers were known to the ancient Greeks. They have also devised a method for generating them.

AMICABLE PAIRS

After generating the number S(N), and if, in turn, we add its aliquot divisors, we get a new number S(S(N)). If we repeat this process, one of the following will happen:

a We shall end up with the number 1.
b. The numbers we get will grow and grow indefinitely.
c. The number we get will repeat itself. In which case we have a perfect number.
d. The numbers we get will enter into a cycle: one number, say N, will generate another, say M, which, in turn, will generate N and so on.

The numbers N and M are known as an **amicable pair.** The smallest such pair is (220,284) and it was known to the Greeks. Over 900 such pairs have been found so far. Can the reader find some of them?

Several formulae were found which give many, but not all, amicable pairs. One of them was discovered by the Arab mathematician Thabet ben Qorrah in the twelfth century:

For n>1 let
a=3*(2 n)–1
b=3*(2 (n–1))–1
c=9*(2 (2*n–1))–1

If a, b and c are prime then ((2 n)*a*b,(2 n)*c) is an amicable pair. For n=2: a=11, b=5 and c=71 which gives the amicable pair (220,284).

e. The numbers we get will enter a larger cycle. In this case we say that we have a **sociable chain.** Several such chains were found including a 28-link chain starting with 14,316.

THE FUNCTIONS TAU AND SIGMA

I left this section, which deals with two functions from number theory, to the end because it involves some algebraic notation which might be difficult for some readers.

The function TAU(N) is defined as the number of divisors of N (including N itself) and SIGMA(N) as the sum of these divisors.

Thus, for deficient numbers SIGMA(N)<2*N, for abundant numbers SIGMA(N)>2*N and for perfect numbers SIGMA(N)=2*N.

The reason why TAU and SIGMA are important is because of their following properties:

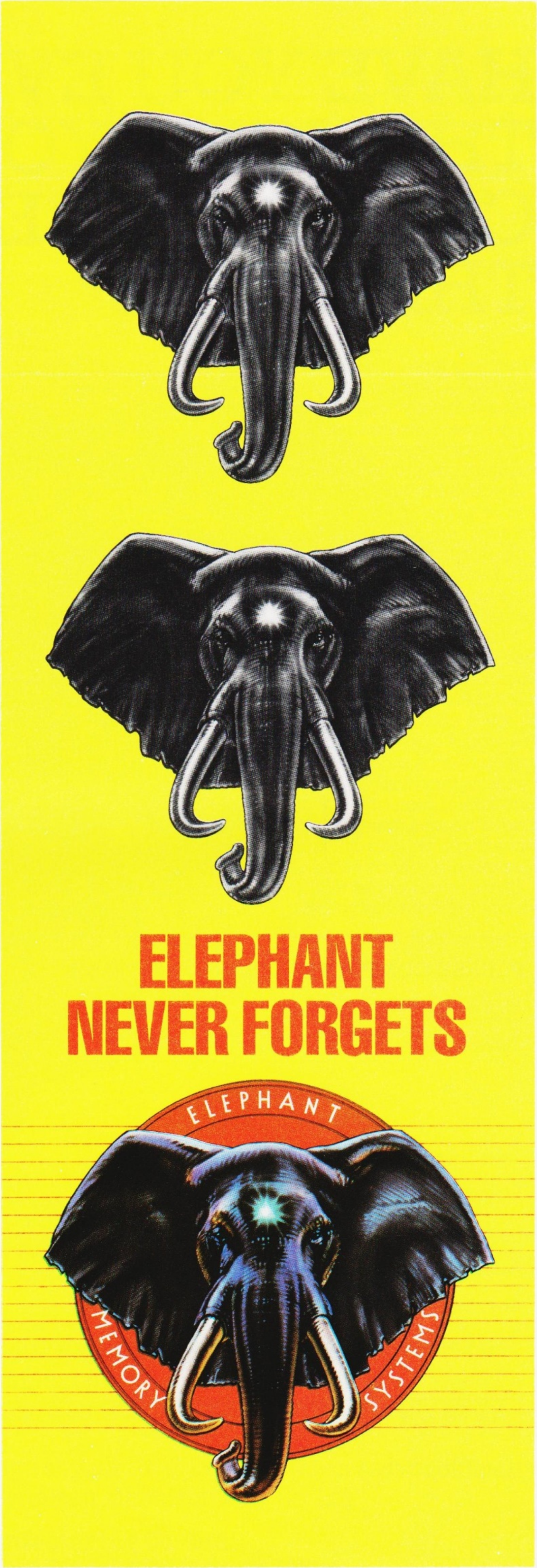
If M and N have no common divisors other than 1 then
a. **TAU(M*N)=TAU(M)*TAU(N)**
b. **SIGMA(M*N)=SIGMA(M)*SIGMA(N).**
Also, if $N=p_1^{\alpha 1}.p_2^{\alpha 2}.p_3^{\alpha 3}...p_k^{\alpha k}$ where $p_1, p_2, ..., p_k$ are prime then:
c. **TAU(N)=** $(\alpha_1+1)(\alpha_2+1)...(\alpha_k+1)$ and

$$\text{d. } \mathbf{SIGMA(N)=}\ \frac{p_1^{\alpha 1+1}-1}{p_1-1}\quad\frac{p_2^{\alpha 2+1}-1}{p_2-1}\quad\frac{p_k^{\alpha k+1}-1}{p_k-1}$$

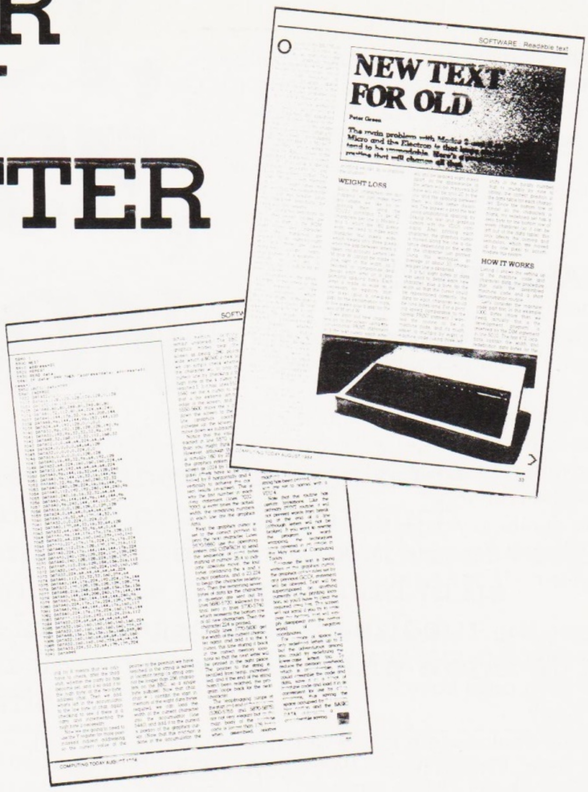
The reader may find formula d useful in trying to solve the penultimate example. The required number should not be divisible by 2 or by 5, therefore it should be of the form:

$$N=3^{\alpha 1}.7^{\alpha 2}.11^{\alpha 3}.13^{\alpha 4}...$$

Any one, or more, of the α's can be 0.



HAVE YOU EVER THOUGHT YOU COULD DO BETTER THAN THIS?



Then why haven't you? Why not put our money where your mouth is? *Computing Today* is always on the lookout for new and interesting programs and articles for publication in the magazine. All submissions will be acknoweldged and the copyright in such works, which will pass to Argus Specialist Publications Ltd, will be paid for.

If you're interested in making your hobby pay its way and you've written a program that you think suits the magazine's content, why not send it to us today with the form below (or a photocopy of it). The address is Computing Today, No. 1 Golden Square, London W1R 3AB: and please mark your envelope clearly 'PROGRAM SUBMISSION' so that it doesn't get confused with all the other mail we receive.

We will need a copy of your program on cassette (or disc, for some systems, if you prefer) together with clear documentation on what it does and how it does it, including a list of the major variables, and if possible some indication of how a conversion to other micros might be attempted. We would appreciate a listing of the program and any screen dumps that you feel might be useful, but not on ZX Printer paper (it doesn't reproduce very well in the magazine). Remember that CT is a general computing magazine and accepts articles for any popular computer including Commodore, Acorn, Atari, Sharp, Amstrad, Sinclair, Oric, Tandy and Genie models.

If you would prefer to make a tentative approach to see if we are interested in your program *before* you put a lot of effort into it (or to check whether we have discs for your particular machine), then that's fine too, provided it is understood that a full write-up will be required before we can publish.

Subject matter can be as broad as you like, bearing in mind that the more readers it will interest, the more likely we are to accept it. A brilliant business program that requires the simultaneous use of four disk drives probably won't be accepted! Also we tend to steer clear of simple arcade games unless, like our Frogger, they demonstrate how to use a particular machine's capabilities to the full.

PLEASE COMPLETE IN BLOCK CAPITALS

Your name: ....

Your address: ....

....

....

Telephone number: ....

Program name: ....

Computer/ memory size it runs on: ....

Amount of memory program occupies: ....

Any special peripherals required? (joystick, discs, printer (etc): ....

Have you sent your submission to another magazine? ....

Is it original or a variation on a theme? ....

Office use only

REMEMBER

ELEPHANT NEVER FORGETS

Get the best from your computer with ELEPHANT disks. Certified 100% error-free and problem-free, and with quality maintained for at least 12 million passes, ELEPHANT disks are guaranteed to meet or exceed every industry standard and are compatible with virtually every computer on the market.

Look for the ELEPHANT sign at your local Dealers – or in case of difficulty, phone or write direct to Dennison Manufacturing Co. Ltd.

Dennison Manufacturing Co. Ltd.
Colonial Way, Watford, Herts WD2 4JY, Tel: Watford (0923) 41244, Telex: 923321
France: Soroclass, 8, Rue Montgolfier - 93115, Rosny-Sous-Bois, Tel: 16 (1) 855-73-70
Germany: Marcom Computerzubehöer GmbH, Podbielskistr. 321, 3000 Hannover 1, Telex: 923818
Other Countries: Dennison International Company, 4006 Erkrath 1, Matthias-Claudius-Strasse 9, Telex: 858 6600

ENTERING THE DRAGON

James Leigh

Dragon users and owners of 6809 home-brew systems should find this article of interest.

This article is an attempt to explore and explain the Dragon 32's system of interrupts in order that the growing numbers of Dragon users now delving into assembly language and machine code may gain some insight into programming with interrupts on this extremely versatile machine. Although this article contains information which should be of use to any Dragon machine code programmer it is not aimed at the newcomer to machine code, and a basic familiarity with the 6809 microprocessor and its instructions are assumed.

RE-ENTRANCY

This term is a term that, like Recursion, we hear a great deal but see very little evidence of it in practice. If a program is 'interrupted' part way through, it is possible that the interrupting routine will corrupt some of the data being operated upon by the main program. For example if the main program fetches data from a location in the 'A' accumulator for some logical or arithmetical operation but is interrupted and diverted to a routine which uses the index register, on return to the main program if an attempt was made to restore the contents of 'A' to the original memory location using indexed addressing, the now corrupted index register could direct the data almost anywhere. However a RE-ENTRANT program is written in such a way that it may be freely interrupted at any point and returned to without corrupting any data or pointers, ideally all your machine code routines should be written this way. (In case you're wondering, Recursion is the ability of a routine to call itself).

Fortunately the Dragon's 6809 microprocessor, unlike many, is fully supportive of re-entrancy, for instance all interrupts except FIRQ, (Fast Interrupt Request), push on to the 'S' stack all internal registers except 'S', and on return from interrupt these registers are pulled from the 'S' stack. (FIRQ saves only the program counter and the condition codes register in order that a fast response to interrupt can be made). This means that as long as you ensure that the 'S' stack pointer is not altered and that the stack itself is intact you should have few problems. It is also a good idea to allocate a small area of RAM to any routine that deeds it for the storage of local variables.

INTERRUPTS IN GENERAL

When an interrupt request is generated (by hardware or software), the microprocessor can be made to suspend program execution and jump to some other program or routine. The sequence of events involved are as follows:–

(1) **The microprocessor** receives the interrupt but continues program execution until the current instruction is completed.

(2) **A check** is made to see if the interrupt flag is clear, (interrupt enabled) or set, (interrupt disabled). If interrupt flag is set continue with main program, if flag clear then go to next step. Step (3)

(3) **Processor interrupt flag** is set (disable further interrupts) and internal registers and program counter is saved by pushing their contents onto the stack.

(4) **An address** is fetched from a vector at the top of memory and a jump is made to the program at this address.

(5) **When an RTI** (return from interrupt) instruction is encountered those registers saved in step (3) are restored and a jump back to the main program is made to the instruction that would have been executed next if the interrupt had not occurred.

The steps listed above show how a typical microprocessor responds to a typical interrupt, however there are some small differences in interrupt handling between different types of mpu.

The 6809 microprocessor used in the Dragon has a total of six interrupts. Most of which are accessible to the machine code user, and these interrupts can be separated into two categories — Hardware interrupts and Software interrupts.

Software Interrupts, as the name suggests, are generated by software and can be programmed to occur at any part of a program simply by inserting a SWI (software interrupt) instruction. When an SWI instruction is encountered by the processor it is dealt with in exactly the same way as a hardware interrupt except that these are non-maskable, in other words there is no interrupt flag associated with software interrupts and consequently they can not be disabled. This is not an oversight on the part of the designers of the 6809 and imposes no restrictions or hardships on the programmer as the SWI instruction can be used in conjunction with other instructions so that if a certain condition is met the SWI is executed and if those conditions are not met then it can be skipped over. It can also be used in a program as a break point so that when executed a routine is jumped to which prints out the contents of the internal registers of the mpu, which have conveniently been pushed onto the stack by the interrupt process, where they can easily be accessed for dumping to the screen. If the stacked register contents are altered they can be returned to their respective registers by executing a RTI (return from interrupt) instruction. This technique is used by many 68xx series monitor programs to give the user the ability to display and alter the contents of the mpu's internal registers.

HARDWARE INTERRUPTS

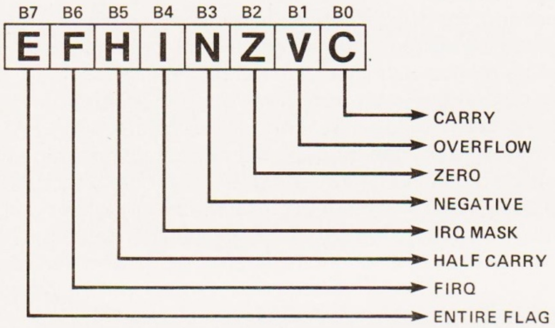
The 6809 has three hardware interrupts and these interrupts are prioritized, this means that an interrupt of a higher priority can interrupt the processor even if a lower priority interrupt has only just occurred, but if a lower priority interrupt occurs after an interrupt of higher priority, then it has to wait until the processor returns from the higher priority interrupt routine. In this way the programmer can ensure that operations that need to be carried out immediately (such as capturing data from an external interrupting device) or operations which are time dependant (such as an interrupt driven real time clock) are seen to immediately.

The 6809 hardware interrupts in order of priority are as follows:–

- **NMI** — Non Maskable Interrupt. Top priority cannot be disabled (no mask flag). All internal registers saved on 'S' stack.
- **FIRQ** — Fast Interrupt Request. 2nd in priority mask = bit 6 of Cond. Codes reg. Saves only Prog. Cntr. Cond. Codes and Direct Page register for faster response.
- **IRQ** — Interrupt Request. Lowest priority mask = bit 4 of Cond. Codes reg. Saves all internal registers.

NOTE: All interrupts use the stack pointed to by the 16 bit 'S' register. The 'S' register, obviously, is not pushed onto the stack.

The Condition Codes register of the 6809 has the following format:–



Example 1. Mixed text and graphics screens

If an interrupt mask bit is set then the associated interrupt will have no effect, if the mask bit is clear then the interrupt is enabled.

The 'E' flag is used by the processor to see if the 'Entire' register contents were saved by the interrupt, or if only the P.C., C.C., and D.P. registers were saved by a Fast Interrupt Request. This is so that the correct number of registers can be restored on a RTI instruction.

DRAGON INTERRUPT VECTORS

The interrupt vectors at the top of the Dragons ROM point to locations in RAM which are set up with the addresses of the interrupt routines on power up. When an interrupt occurs the 6809 fetches an address from the interrupt vectors at $FFE2 to $FFFF, and jumps to the program at this address. In the Dragon this address contains a single jump instruction which re-directs the processor to the actual interrupt routine, however as this jump instruction is in RAM it can be altered to point to one of your own routines. This means that the interrupt will now cause your routine to be executed.

The locations at which the jump addresses are located are as follows:–

$0100	SWI 3	;Unused by Dragon
$0103	SWI 2	;Unused by Dragon
$0106	SWI	;Unused by Dragon
$0109	NMI	;Unused by Dragon
$010C	IRQ	;Normally used to update TIMER and for the PLAY routines.
$010F	FIRQ	;Normally used for cartridge initialisation.

The hardware interrupts can be accessed in the following ways:–

NMI ; accessed directly from pin 4of the expansion port edge connector.

FIRQ ; accessed indirectly (via CB1 of PIA1) at pin 8 of expansion port.

IRQ — This interrupt is not accessible by the user as it is used for and by the Dragon's own internal timing, ie. to make sure the screen is not accessed at times that could cause interference (snow) on the display, and also to update the TIMER locations $0112-3 every 20mS.

USING THE INTERRUPTS

In order to use the Dragons interrupt system for your own machine code programs and routines you must first load your program into memory and only then should you change the jump addresses located at $0100–$010F. If this is done in reverse order by changing the jumps first, then if an interrupt occurs a jump will be made to a routine that does not yet exist in memory.

When using the software interrupts, of which there are three, SWI, SWI2 and SWI3, then the interrupt routine can be of any length. However, when using a hardware interrupt the interrupt routine must be kept short in order that a return from the interrupt to the main program can be made before the next interrupt occurs. Imagine what would happen if this were so; the first interrupt would cause a jump to the routine, stacking the cpu registers on the way, before the routine had been completed another interrupt occurs stacking the cpu registers and again jumping to the start of the interrupt routine and so on. Apart from the fact that the interrupt routine would never be completely run, tying up processor time without achieving anything, also it would not be long before the user stack grew so big as to overwrite the program area, data etc., or to run out of memory altogether! This should be borne in mind particularly when attempting to use the IRQ interrupt as this line is pulsed once every 20 mS (50 times a second) by the Dragons hardware, in order to update the TIMER locations in RAM. This does not mean that you will have 20 mS running time available for your interrupt routine when using IRQ as there may also be other things to be done. For instance supposing you wish to have the time displayed while running a basic program, this entails using IRQ to cause a jump to a machine code routine which updates the time and displays it on screen. In order that the Dragon Basic continues to run at a reasonable speed the machine code must be as short and as fast as possible.

It must be remembered that when a machine code program is run using EXEC or USR that the IRQ line is still being pulsed causing the machine code program to be interrupted every 20 mS. Normally this will present no problems, however if you require maximum speed from your machine code in order to process a large amount of data for instance, or for fast graphics, then it is a good idea to disable IRQ. Also there may be certain applications where IRQ must be disabled, if for example you are using any interrupt driven hardware, or even software where timing is crucial. Fortunately the 6809 microprocessors prioritized interrupt system greatly simplifies things, especially as Dragon uses the lowest priority 'IRQ' for its own use.

DISABLING IRQ

If your machine code does not call any of the subroutines in the Basic ROMs then disabling IRQ is a simple matter of "masking" the interrupt by setting bit 4 of the Condition Codes register when your code is first entered. This is done by using an ORCC # $10 instruction as the first instruction of your code. This method can also be used if you are calling subroutines in the Basic ROMs of which you are certain contain no instructions which re-enable IRQ.

The last instruction of your machine code before the final RTS (return from subroutine) or RTI (return from interrupt) should be an ANDCC # $EF which clears the interrupt mask thus re-enabling the IRQ.

When interfacing the Dragon to an external interrupt driven device which uses either NMI (non-maskable interrupt — highest priority) or FIRQ (fast interrupt request — 2nd in order of priority) then any interrupt of lower priority will be automatically masked (disabled). Problems will only arise if subroutines in the Basic ROMs are called which re-enable interrupts (see above).

If you would like to use any of these ROM based routines but you are not sure of them, then you can always disable IRQ at the source. This would prevent an interrupt from occuring even if it was enabled at the mask level. This can be done by clearing bit 0 of control registers A and B of PIA 0, which are located at addresses $FF01 and $FF03 respectively; this will prevent the TV/monitor field synchronisation pulses on CA1 (pin 40 of PIA 0), form causing IRQ to be pulsed.

An alternative but less elegant solution is take the risk of IRQ being re-enabled when a ROM subroutine is called but to change the contents of the IRQ vector from JMP $9D3D (7E 9D3D) to RTI (3B) so that an IRQ does an immediate return from interrupt. The only penalty being a dozen or so lost mpu cycles.

USING IRQ

If you intend to make use of the 50Hz IRQ in your own routines then you will need to change the contents of the IRQ vector so that it causes a jump to your own machine code routine. If this routine is required to work in conjunction with the Basic then instead of exiting your routine by use of RTS or RTI, you must use JMP $9D3D so that the interrupt routine in Basic is executed also. If this is not done the 6809 will get locked in executing and endless loop with the only way out being to switch off.

The best way of re-directing the IRQ is to use a short machine code routine to change the contents of the IRQ vector ($010D & $010E). It is possible to do this with POKE instructions but if an interrupt should occur when only one byte of the address has been altered the system will crash.

The Dragons 50Hz IRQ is well known and various machine code routines which make use of it have been published such as real time clocks and keyboards auto repeat. Less well known is the fact that the interrupt frequency can be stepped up to that of the TV/monitor horizontal synchronization pulses. To do this clear bit 0 of PIA 0 (addr $FF01), and set bit 0 of PIA0 (addr $FF03). The IRQ frequency is now approximately 15.6Hz which gives you only about 50 machine cycles for your code, so keep it short and avoid loops!! Fortunately the IRQ mask will be set on every interrupt which occurs while the mask is clear, this means that the program will be free from interrupts until the mask is cleared by software.

One possible use for this high speed interrupt would be for a driver routine for a light pen. The horizontal sync signal causes an interrupt request after every line scan of the display, this causes a jump to a routine which counts the line scans from 0 to 191 until a higher priority interrupt occurs such as FIRQ, which is caused by a very fast photodiode and associated circuitry in the actual light pen. The FIRQ routine runs a single pixel the length of the line indicated by the line counter until a second FIRQ pulse is received. The routine now contains the line number and pixel position. This is given only as an example, I'll leave you to work out the details.

6809 STACKING ORDER

When an interrupt occurs the 6809 automatically stacks its internal registers on to an area in RAM pointed to by the 16 bit 'S' register in the following sequence:–

PC lo, PC hi, U lo, U hi, Y lo, Y hi, X lo, X hi, DP, B, A, CC

Registers are pushed into DECREASING memory addresses, and pulled from INCREASING memory addresses. This means that if we take 's' as the address contained in the 'S' stack pointer, then PC lo is stored at address s-1 and PC hi is stored at s-2 and so on until we reach CC which is stored at s-12, which is now the new stack address contained in the 'S' register.

EXAMPLES

The following short machine code routine (example 1) enables the Dragon to display two lines of text from the text screen followed by the top four fifths of the hi-res Pmode 3 screen, there is however a short gap between the two types of screen which is in the unsupported "mode 24". This is due to the 6883 SAM chip being switched from one mode to the other during display time.

The program works by switching to text mode at the beginning of every screen and then a delay loop is run through a number of times before switching back to the hi-res mode. The length of the delay loop determines how many lines of the text screen are to be displayed but this should be kept short or there will be little time left for actual Basic program execution.

A short routine 'INIT', initialises the interrupt vector at $010D with the start address of the screen switch routine "TEXTRES", and another short routine is provided, "EXIT", which replaces the original interrupt vector contents to enable normal running of the Dragon.

```
$               ORG    $7F80

7F80  INIT      LDX  #TEXTRES     8E 7F8E     ; SETUP IRQ VECTOR
7F83            STX   $010D       BF 010D     ;
7F86            RTS               39

7F87  EXIT      LDX  #$9D3D       8E 9D3D     ; REPLACE IRQ VECTOR ORIGINAL
7F8A            STX   $010D       BF 010D     ; CONTENTS
7F8D            RTS               39

7F8E  TEXTRES   LDA  #$07         86 07       ; SET TO TEXT MODE
7F90            ANDA  $FF22       B4 FF22     ;
7F93            STA   $FF22       B7 FF22     ;
7F96            STA   $FFC6       B7 FFC6     ;
7F99            STA   $FFC4       B7 FFC4     ;
7F9C            STA   $FFC2       B7 FFC2     ;
7F9F            LDX  #$03FF       8E 03FF     ; DELAY TIME
7FA2  DELAY     LEAX  -1.X        30 1F       ; DELAY LOOP
7FA4            BNE   DELAY       26 FC       :
7FA6  PMODE3    LDA  #$E7         86 E7       ; SET TO PMODE
7FA8            ORA   $FF22       BA FF22
7FAB            STA   $FF22       B7 FF22
7FAE            STA   $FFC7       B7 FFC7
7FB1            STA   $FFC5       B7 FFC5
7FB4            STA   $FFC3       B7 FFC3
7FB7  END       JMP   $9D3D       7E 9D3D     ; RETURN TO BASIC
```

Example 1a. Mixed text and graphics demonstration program

```
5 REM ** GRAPHICS & TEXT MIXED DEMO **

10 PRINT@ 0,"******* THIS IS A DEMONSTRATION*************";

20 PMODE 3,1:SCREEN 1,1:PCLS 2:CLS:EXEC &H7F80

30 POKE &HBA,&H04:POKE &HBB,&H40:REM LOCATE GRAPHIC SCREEN UNDER TEXT LINES

40 FOR X= 5 TO 100:COLOR RND(4):CIRCLE (128,96),X:NEXT X

50 IF INKEY$ = "1" THEN EXEC &H7F87 ELSE 50:REM $7F87 = EXIT ROUTINE
```

Example 1b. BASIC TEXTRES loader.

When using TEXTRES with Basic, use EXEC & H7F80 to enter dual screen mode and EXEC & H7F87 to return to normal text screen only.

```
999  REM ** POKE 'TEXTRES' CODE INTO RAM **

1000 CLEAR 200.&H7F7F

1010 FOR X = &H7F80 TO &H7FB9:READ A$:POKE X,VAL( "&H"+A$):NEXT X

1020 RETURN

1030 DATA 8E,7F,8E,BF,01,0D,39,8E,9D,3D,BF,01,0D,39

1040 DATA 86,07,B4,FF,22,B7,FF,22,B7,FF,C6,B7,FF,C4

1050 DATA B7,FF,C2,8E,03,FF,30,1F,26,FC,86,E7,BA,FF

1060 DATA 22,B7,FF,22,B7,FF,C7,B7,FF,C5,B7,FF,C3,7E,9D,3D
```

Example 2. Routine to give text and colour in PMODE4

As an example of graphics and text try the Basic program in Example 1a. Line 30 moves the graphics screen up by poking the 'start of graphics page' locations with $0440 in order that graphics page starts just under the two lines of text. These text lines are at the top of the text screen and so all PRINT commands must be preceded by a CLS or preferably use a PRINT @ command as this will not affect the graphics screen.

As it stands a certain amount of graphics occur off of the viewing area, if you wish you could change the 'end of graphics page' pointers which are located at address $00B7 and $00B8.

When using this method of displaying text and graphics a certain amount of processing time is lost due to the delay loop being executed while text is on the screen. However the loss in speed is not noticeable and if your machine can handle the double speed POKE you could always use it. Also, if you break out of a program before it gets to the line where EXIT is executed you will still be able to operate Dragon in direct command mode but you will only see the two top lines of text.

This machine code routine at last provides Dragon with a relatively easy way of displaying scores and prompts for your hi-res programs, and because of the small amount of code involved it could easily be read from data and poked into memory at the beginning of any program that uses it.

The short Basic routing (example 1b) can be tagged on to the end of any program which is to use the TEXTRES routine, and this will POKE the code into reserved memory. This subroutine should be called at the start of the program.

Another method by which Dragons internal 50Hz interrupt may be used to display text and graphics screens simultaneously, and thus have text on the hi-res screen, is to switch from text mode to Pmode and vice-versa on every other interrupt (example 2). This means that instead of having a display which is refreshed every 20 mS we have alternate text and graphics sharing the screen each mode being displayed for 20 mS out of every 40 mS. The problem with this alternating display is that a 25Hz flicker occurs and depending on the screen colour in use can be a little distracting. There are however some screen colours which are quite useable, also the combination of text screen colours and Pmode colours give an incredible range of modes and colours. Using this technique it is also possible to have nine two colour 256 by 192 pixels modes. This is done by alternating Pmode 4 and a text screen cleared to the chosen colour. If white on black is used instead of black on white for the hi-res screen then another shade of the same 9 colours is available as the background. It is also possible to use the green and black screen to obtain yet another range of colours. Best results are obtained with the darker colour backgrounds and with the contrast and brightness of the monitor/TV adjusted for least flicker.

The COLOR4 machine code routine tests Dragons upper case/lower case flag to see if the text screen should be inverted or not, and this feature can be used in direct command mode if required simply by using Dragons SHIFT 0 case change function. Try the following:– EXEC &H7F80:PMODE 4:PCLS enter, now type in a few assorted graphics commands in direct mode (no line number). You will see the circles, lines etc appear superimposed on the text. Now use the CLS n command to clear the text screen to any color . . . sorry, colour! When you have done this use the SHIFT 0 and you will see the screen colour change but the graphics will still be displayed. Experiment with different text screen colours and the SHIFT 0, you may be surprised at the range of colours available, 16 in all and another 16 different shades in SCREEN n,1, but to get these will require that the contents of $7F95 and $7FC7 be changed to $FF and $0F respectively.

If you wish to use PMODE 3 then change the contents of $7F95 to $E7 (screen 0) or $EF (screen 1). Only minor changes to the code are required in order to use it with any PMODE, but I'll leave you to work that out for yourself. Do not forget that when in inverse/lower case mode the commands will have to be entered using the SHIFT key or a syntax error will occur, this can however be changed so that inverse (green on black) is the normal mode simply by POKE &H7FA8, &H27, which changes the BNE, (branch if U case flag not 0) to a BEQ, (branch if flag is 0) command. Using SHIFT 0 will now give access to the lighter shades and the text screen will be green on black.

CONCLUSION

Using Dragons system of interrupts it is possible to have all sorts of text and graphics modes mixed for instance using the method of example 1 it is possible to place text at almost any position on the graphics screen, while example 2 shows just how versatile the Dragons hardware can be although this in no way makes up for the lack of decent text and graphics mixed.

```
               ORG    $7F80
       FLAG    EQU    $00E8
7F80   INIT    LDX    #COLOR4   8E         ; SETUP IRQ VECTOR
7F83           STX     $010D    BF 010D    ;
7F86           CLR     FLAG     0F E8      ; CLEAR TEXT / PMODE FLAG
7F88           RTS              39         ;
7F89   EXIT    LDX    # $9D3D   8E 9D3D    ; REPLACE IRQ VECTOR ORIGINAL
7F8C           STX     $010D    BF 010D    ; CONTENTS.
7F8F           RTS              39         :

7F90   COLOR4  COM     FLAG     03 E8      : COMPLEMENT TEXT / PMODE FLAG.
7F92           BNE     TEXT     26 32      : IF FLAG = 0 THEN HI-RES ELSE TEXT
7F94   HIRES   LDA    #$F7      86 F7      ; SET PMODE 4 .
7F96           STA     $FF22    B7 FF22    ; SET PIA PARMS.
7F99           STA     $FFC7    B7 FFC7    ; SET SAM PARMS.
7F9C           STA     $FFC5    B7 FFC5    ;
7F9F           STA     $FFC3    B7 FFC3    ;
7FA2           LDX    #$0400    8E 0400    : GET TEXT SCRN ADDR INTO X'REG.
7FA5           TST     $0149    7D 0149    : TEST U.CASE / L.CASE FLAG.
7FA8           BNE     INV2     26 0E      : IF 0 THEN INVERTED CHARS ELSE NORMAL..
7FAA   INVERT  LDA    #$BF      86 BF      ; INVERT TEXT SCRN CONTENTS (GREEN ON BLACK)
7FAC           ANDA    ,X       A4 84      ;
7FAE           STA     ,X+      A7 80      ;
7FB0           CMPX   #$0600    8C 0600    ; IF NOT END OF SCREEN
7FB3           BNE     INVERT   26 F5      ; DO NEXT CHAR.
7FB5           JMP     $9D3D    7E 9D3D    : RETURN TO BASIC
7FB8   INV2    LDA    #$40      86 40      ; RE-INVERT TEXT SCREEN
7FBA           ORA     ,X       AA 84      ; TO BLACK ON GREEN
7FBC           STA     ,X+      A7 80      ;
7FBE           CMPX   #$0600    8C 0600    ; IF NOT END OF SCREEN
7FC1           BNE     INV2     26 F5      ; DO NEXT CHAR.
7FC3           JMP     $9D3D    7E 9D3D    : RETURN TO BASIC
7FC6   TEXT    LDA    #7        86 07      ; SET TEXT MODE ( SCREENO)
7FC8           STA     $FF22    B7 FF22    ; SET PIA PARMS
7FCB           STA     $FFC6    B7 FFC6    ; SET SAM PARMS
7FCE           STA     $FFC4    B7 FFC4    ;
7FD1           STA     $FFC2    B7 FFC2    ;
7FD4           JMP     $9D3D    7E 9D3D    : RETURN TO BASIC
7FD7
** BASIC DEMO ROUTINE FOR PMODE 4. TEXT AND TWO COLORS **

10 PMODE 4,1:SCREEN 1,0:PCLS:EXEC &H7F80

20 FOR R=5 TO 80 STEP 5:CIRCLE (128,96),R:NEXT R

30 FOR C=0 TO 8:CLS C:PRINT @96,"PMODE 4 WITH COLOUR !";

40 FOR D=0 TO 500:NEXT D

50 IF PEEK(&H149)=0 THEN POKE &H149,255 ELSE POKE &H149,0

60 NEXT C:POKE &H149,255:EXEC &H7F89

70 END
```

INSIDE BASIC :1

Don Thomasson

Books which explain (?) how to write BASIC programs can be found in abundance. But what happens to a program once RUN is typed remains a mystery to most of us. This month we begin a series which we hope will provide an insight into the workings of the BASIC interpreter and that legendary and mystical beast, the Machine Operating System.

Many computer users have little or no desire to know what goes on inside their machines. Providing the response to the commands they input is satisfactory, they are quite satisfied. Now and then, however, a situation arises in which they are puzzled by the way their equipment responds, or by limitations imposed by the instruction book, and they would like to understand more. We are therefore offering a series of articles which explain the inner workings of some familiar programs. The explanations will be kept as simple as possible, pointing out, where appropriate, the difference between various systems.

Not surprisingly, we are going to make a start with the BASIC interpreter.

BASIC is an interpretive language. That means that its programs are interpreted at the time they are run, using fairly readable text-like program symbolism which is understandable by both man and machine. This contrasts with compiling languages, which are written in 'source' code and then used to compile 'object' code, which the computer can understand directly. As we will show later, BASIC can be used as a compiling language, but not with extreme efficiency.

Whatever type of language is used, the computer will only respond to 'machine code', and to the particular form of machine code that suits its central processor. In compiled languages, the object code is in exactly the right form for the computer to understand, and it is executed at maximum speed. An interpretive language works in a different way. The 'reserved words' used to express the programmer's wishes are linked to blocks of machine code. If the word PRINT is found, the interpreter calls into action a block of code which will pass data to the display. The need to identify the reserved words and access the associated block of code is one of the reasons why interpretive languages tend to be relatively slow, but it is by no means a major reason.

A BASIC interpreter has many tasks to perform. It must assist the user to set up his programs, and provide facilities for storing the programs on tape or disc. It must perform the task of interpretation, recognising reserved words and variable names. It must be able to display a listing of the current program. These tasks are obvious, but there are others that hide themselves more successfully. Most BASIC interpreters can handle floating point numbers, and this requires complex processes. Even the handling of strings is unexpectedly complicated, by comparison with what might be imagined.

Given a command, the interpeter will go away and execute it, and then return to the quiescent condition in which it is ready to accept further instructions. We will begin with this 'Ready' condition, marked by the display of a prompt which may be the word 'Ready' or perhaps a single character having the same meaning.

THE READY STATE

The ready state is entered at initial switch-on, when a task has been executed, and in some systems when a Break or Escape key has been pressed to abort the process in hand. The system is then prepared to accept keyboard data into a RAM buffer set up for the purpose. Most of the input characters are stored in the buffer in sequence, but there are a few exceptions.

For example, there is Delete, which wipes the last input character off the screen and removes it from the input buffer, allowing you to have afterthoughts or correct mistakes. Some systems implement a character code which wipes out the whole line, letting you begin again. All systems have a code that indicates completion of the line. It used to be called RETURN, because it moved the display cursor to the left-hand end of the next line, like Carriage Return on a typewriter, but that led to confusion with other meanings of the word, and ENTER is now used more generally.

When ENTER is pressed, the system switches to a new mode and begins to analyse the data set up in the input buffer. This should consist of a command statement which may or may not be prefaced by a line number. If there is no number at the start of the line, the system assumes that the statement is to be executed immediately, in 'direct' mode. If a number is present at the start, it is assumed that the statement is to be added to the existing stored program, being inserted at a point indicated by the magnitude of the number.

Assuming there is no line number, the system proceeds to 'edit' the line into suitable form for execution. It will first look for a reserved word at the start of the line, and if it finds one the word will be replaced by a 'token' byte. This occupies less space than the original word, and — as we will see — makes access to the executive code blocks simpler.

Conversion from a text word to a token usually involves looking through a table of reserved words seeking a match. The **Sinclair ZX** systems avoid the need for this by arranging for the token to be set up directly from the keyboard. Pressing the RUN key generates the token, not the letters of the word, which only appear when the program is listed.

The ZX system is possible because the body of the input line should always begin with a reserved word representing a command, so when the first non-numeric character (other than space) is input, it can be interpreted as a command, not in terms of any other meaning of the same key.

This has an odd side-effect. In most systems, absence of a reserved word at the beginning of a line leads to the assumption that the word LET is to be understood. The ZX system cannot allow this assumption, because it depends on the presence of the initial command, and LET must be input to fulfil that requirement.

Another consequence of the system is that there is never any danger of mistakes arising through the presence of a reserved word within a variable name. Where a system has to look up a table, it may come to some odd conclusions. A variable name such as NORTH may get translatede as N OR TH. This is the more likely in systems which permit the dangerous practice of leaving out spaces, which makes some statements totally unreadable and can lead to misunderstandings by both man and machine.

Because of this, different machines have different rules regarding variable names. Most systems require an initial letter, which may be followed by letters or numbers. Older systems allow only two significant characters in a name, and at least one more recent system allows only one. That allows economy of storage, but is sometimes a serious limitation. On the other hand, it is rarely sensible to use names up to forty characters in length.

The 'editing' process continues down the input line, copying variable names unaltered, or converted to upper case in some systems. Any text enclosed in quotes is copied unaltered. The treatment of numbers varies a great deal. Some systems convert even a one-digit number into floating point form, with a preliminary warning byte as a prefix. This is done to speed up interpretation, but it uses up unnecessary storage space. Other systems are content to leave the number unchanged, converting it to the required form during interpretation. Remember, it will have been input in decimal form in many instances, and that is not directly readable by the computer, which requires binary numerics.

During the editing process, the length of the input line changes, and this may be coped with by the use of two buffers, the second taking the edited form of the line, or by using two pointers with a single buffer, in which event it may be necessary to move the original input data up and down to avoid overwriting it.

After the edit process has been completed, the line is in the form required for execution. However, it also needs to be in a form suitable for the generation of a listing, and this may entail compromise. In one notable case, mathematical expressions were converted into Reverse Polish format, to speed up calculation, but this created serious problems with the listing process, since the original line had to be recreated.

For 'direct' action, the line as edited can be executed without more ado. All that is necessary is to set the 'scan pointer' to the start of the line. We will come back to the scan pointer later, when we look at the interpretation process.

LINE NUMBERS

When the line begins with a number, a lot more action is needed. The number is converted to binary and stored. Then a search is made through the existing stored program to see whether a line of that number already exists. If it does, it is erased by copying all the higher part of the program down to cover it. The editing process can then proceed. If it finds nothing following the line number, the system returns to the Ready state. Erasure of the line is all that is required.

The need to search for a given line number arises in other BASIC functions, and there is an interesting difference in the way it is implemented in older and newer systems. The old method used a line format which began with two bytes giving the line number in binary, then two bytes giving the address of the start of the next line. This allowed the line search program to read a line number, read the address of the next line, and jump to it directly, without having to scan through all the intervening data. Fine, but what happened when a line was erased or inserted? It was then necessary to go through the whole program revising the link addresses.

More recently, it has been realised that it is simpler to replace the link address by a single byte giving the length of the line. That is unaffected by changes to other lines, and also occupies less space. It is still possible to perform a skipping search through the program, the address of each line number being found by adding the line length to the address of the previous number.

To the user, the only difference is that he has slightly more room to play with, due to the saving of a byte, but he will rarely notice the saving in time, because all these processes are executed in the twinkling of an eye.

The next process is insertion of a new line. Remember, any line of the same number has already been eliminated, so the process is the same for an entirely new line or a replacement. The search process will have established a store address at which the new line is to be entered, whether it found a line of the same number or not, because the search ends when a line number equal to or greater than the number of the input line is found. During the edit process the length of the edited line has been determined, and the new line number and length byte are edited into the edited line.

A space must now be created by moving upwards all program lines above the insertion point. Then the new line can be copied into place. When all these processes are considered, it is incredible that the time taken to execute them can be so brief.

When they have been carried out, the new line is safely set into position, and the system returns to the Ready state.

With a direct command, the situation may be different. If the command is RUN, for example, the system will begin to interpret the stored program, and that is what we must examine next.

INTERPRETATION

During the interpretation process, a system variable called the scan pointer is used to read the characters and symbols of the stored BASIC program. In response to the command RUN, which is interpreted as a direct or immediate command, the scan pointer is set to point to the start of the program. It will skip past the line number, which is of no interest for the moment, and look for a token representing a command. Failing that, it will assume that the command LET is understood.

In either case, the response must be to enter a block of machine code which will execute the command. A convenient way of doing this is to use the token as a displacement pointer to a table of link addresses. The required block can then be entered by a jump or a subroutine call. The latter method is more convenient for those who wish to use BASIC functions in support of their machine code programs, but is not a consideration that looms large in the minds of those who write the interpreters.

The most common command is LET, which is why it is allowed to be omitted. The related executive routine first looks for a variable name, this being the dependent variable which is to be set according to the remainder of the statement. Following the variable, there must be an 'equals' sign, and following that there must be an 'expression', which may be a single number or may be considerably more complex.

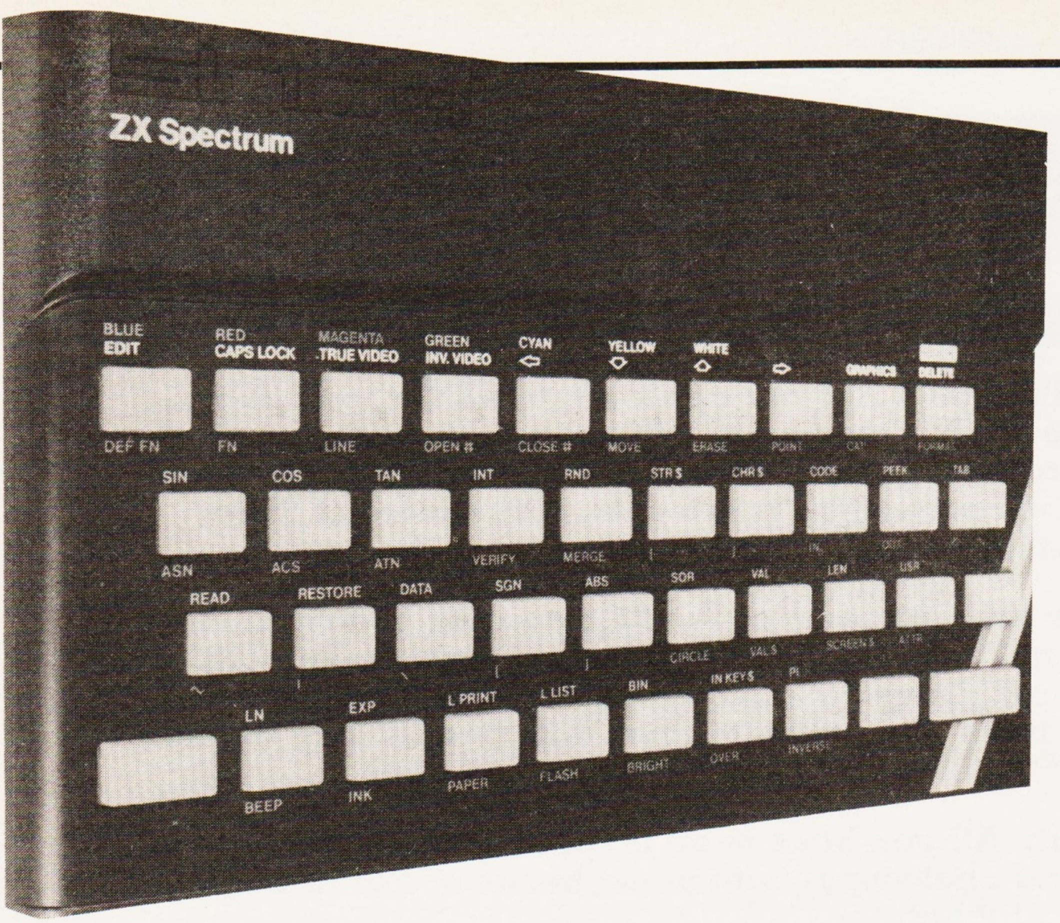
The LET routine stores away the variable name or the location in store which is reserved for the value of the variable, and proceeds to evaluate the expression. This can be a complex process, because the various mathematical functions must be performed in the right order, which is not necessarily the order in which they appear in the BASIC statement. First, the contents of brackets must be resolved, beginning with the innermost brackets if bracketed expressions are nested within one another. Then multiplications and divisions can be performed, and lastly any additions or subtractions.

The way this 'precedence' is maintained varies widely. In older systems, data and operators were put on to the machine 'stack', a last-in-first-out store that is also used to hold return addresses and other data in most systems. By juggling the stacked data, it was possible to bring the highest priority actions to the top of the stack and execute them first. Another approach is to create a ladder or nest of subroutines, each routine calling the next in order of increasing precedence, and then performing its own function when the higher order routine returns.

This is usually the most difficult part of the interpreter system to work out, especially as each individual action can be quite complex in itself. The efficiency with which the actions are executed has a strong bearing on the overall execution speed of the machine.

The evaluation process is not used exclusively by LET, so it is usually called as a subroutine. It also arises in FOR statements, for example. The command FOR is followed by the equivalent of a LET statement to set up the initial value of the index variable. Then comes the word 'TO', which warns that the end value of the index variable must now be set, this involving a further process similar to LET, though in this instance with an understood identity for the dependent variable. Then the word STEP may or may not follow. If it does, a third LET-like process is required to set up the STEP variable.

And here we reach a point of critical difference between one system and another. Some



maintain a special area of store in which the key values of a FOR statement are held, and this area is necessarily of limited size. It is therefore necessary to perform a mopping-up operation when the index variable reaches the final value. Otherwise, the area would soon fill up. For this reason, perhaps misguidedly, BASIC tutors advise that you must never jump out of a FOR loop, but must ensure that it reaches its proper completion. Failure to observe this rule leads to an accumulation of abandoned loops and their data, and eventually to a report that you have used too many.

But this is not true of all systems. The **AMSTRAD CPC464,** for instance, will allow you to jump out of FOR loops with impunity. Firstly, it does not use a special area for storing the variables, removing one limitation, and secondly it will cheerfully use the same variable holds again and again for a given index variable. So it is unwise to believe the tutors unreservedly. To refrain from jumping out of FOR loops is a safe course, but may sometimes be inconvenient, and it is then that you need to know whether your system will take kindly to breaking the rule. The best way to find out is to try it and see. The User's Manual is unlikely to state the facts in direct terms, deeming the point too complex.

GOTO and GOSUB

Purists object to the GOTO function of BASIC, as a matter of principle, but it would be difficult to write a program of any complexity that did not use the function. It is true that there are other ways of creating loops, but the GOTO, especially if it is conditional, is a very useful weapon, even in the most sophisticated systems. However, there are objections to it in another sense.

The instruction to GOTO line so-and-so requires that the position of the line in store should be located. This means searching through the program until the required line is identified, the routine used often being that used to search for a line during program entry. With a big program, this search takes time, and is another reason for the limited speed of BASIC. There are ingenuities which minimise the problem. One system checks whether the GOTO is a forward or backward jump, and limits the search to the appropriate part of the stored program, but that reduces the problem without removing it completely.

Once the required line has been located, the scan pointer can be changed to point to the line, and there are no further problems.

A GOSUB is a GOTO with the previous scan pointer saved, probably by pushing it on to the stack.

The provision of Procedures as an alternative to subroutines has been fashionable of late, but the practical difference is not all that great. The address of a procedure may be stored as a variable linked to the procedure name, which saves the need to search for the destination line, but the variable has to be located. The real advantage lies in the technique of passing variables to procedures, and the provision for establishing variables which are local to the procedure.

Other jump and loop mechanisms work in broadly the manner which has been described for GOTO and GOSUB. As with FOR loops, some systems will allow you to jump out of such loops, others will not. The reason is the same as that given for FOR loops: It depends on the way the loop data is stored.

CONCLUSION

Apart from the more complex mathematical functions, we have covered the more familiar actions of the interpreter, and that is a good point at which to end our first excursion into the internals of BASIC. In the next part, we will be looking at the relationship between BASIC and the main operating system of the computer, and at the way floating point is handled.

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, software listings, 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 1984 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)

£15.00 for 12 issues UK	□
£17.50 for 12 issues Overseas Surface	□
£50.00 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 ....

CT Feb '85

TASWORD 464

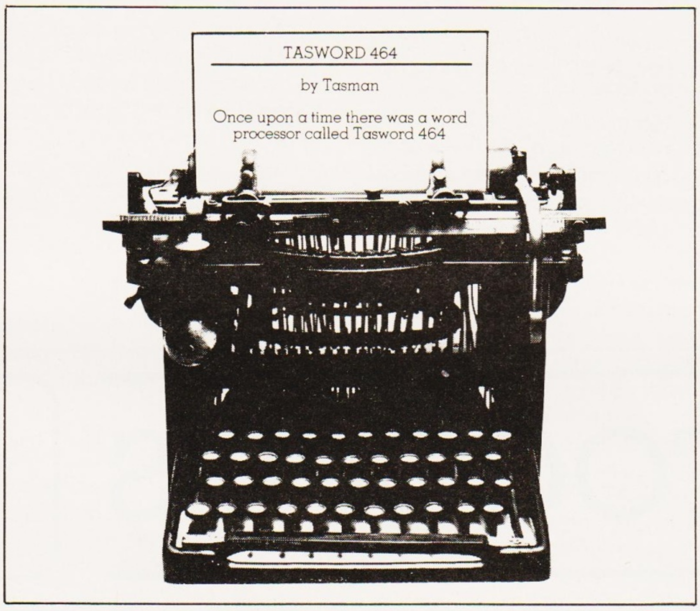
J J W Spencer

Many home micros are used for word processing these days, so we've taken a look at Tasman's offering for the Amstrad CPC 464.

One of my main reasons for buying an Amstrad was its potential as a home word processor. To date Tasword is the only package available but having fallen out with my typewriter I simply could not wait to see what other packages might offer. Relying on Tasman's reputation founded upon the Tasword package for the Spectrum I sent a cheque; their reputation remains intact. I was, however, disappointed with the presentation until I read the letter explaining that this was not the final program. It says a great deal for any software house when they send out a temporary version of a package rather than simply announcing that the final version will be delayed. The only difference was that the copy I had been sent would only hold around 9K of text while the final program should hold about 14K. The manual despite being a photocopy was complete. It was well laid out and informative, the index linked the commands listed on the screen to the relevant part of the manual.

STARTING UP

Tasword loads at the slower baud rate but Tasman have chosen not to protect their program so that the user can make back up copies with a choice of saving in fast or normal mode. One can also customise the page layout, special characters, printer control characters, a second character set as well as page, pen and border colours. In fact, one can even change the shape of the cursor. The facility to make back up copies is thus very useful — I hope it will not be abused. In fast mode the program takes about 3 minutes to load and on completion the user is presented with six lines of control characters, an inverse line that serves as a key to the control symbols, 16 blank text lines, another inverse line showing margin and tab information followed by the last line which shows the state of various command switches. The last line also shows the position of the cursor by giving the line and column number. One feature that Amstrad left out is a CAPS LOCK indicator, Tasword has one. When the CAPS LOCK is on the bottom line messages are displayed in capital letters.


TASWORD 464
by Tasman
Once upon a time there was a word processor called Tasword 464

MAIN FEATURES

The package really does seem to have all of the features that one could want. Files can be saved and loaded as well as merged. The cursor can be moved by character, word, line or by page. The whole screen can be scrolled sideways in order to view text at the maximum width of 128 characters. Pieces of text can be moved left and right, blocks of text can be copied or repositioned. Text can be deleted as characters, words, lines or blocks. Margins can be set for any width up to 128 positions; this facility can be used to write paragraphs of different widths by setting the margin and then rejustifying the paragraph. A line can be centralised but there is no command to centre a paragraph. TAB commands are present and are very easy to set, the positions are marked on the bottom line. Headers and footers are available and the user can choose to have each page numbered. Page breaks can be shown on the screen the length of the page being entered via the CUSTOMISE command. There is a command to search and exchange text which is easy to use but the user has to stop the search otherwise the program searches all of the empty lines. If text has been exchanged then rejustification will take place to fit the current margin settings, so be careful.

The method of inserting text was the feature that I liked least. Inserting a letter within a word was fine but adding text within a line caused the whole line to be broken up, I found myself constantly rejustifying and then having to find where I was before I started. I do appreciate that on a package of this type any other system would have taken far too long. If one only wants to change a letter then simply type over the old text.

One final option that caught my eye was the second character set, this is displayed at the top of the screen after CTRL & / are pressed after which any one of the 92 weird and wonderful shapes can be printed to the screen, though I doubt by my printer. Operating the system is simple, one only has to look up at the HELP window to find the two keys needed to initiate a command. One feature that is valuable is the ability to leave the program, define one of the numeric keys as a frequently used word, or words, and then return to the program.

TYPING BEGINS

The Amstrad keyboard makes typing a pleasure and despite reaching quite high speeds I never managed to defeat the program. The wordwrap feature works very smoothly and for any margin settings. The only problem with wordwrap is that long words are often better hyphenated, especially in narrow margin text. I could only manage this by turning off the wordwrap, entering the word and then turning wordwrap back on again.

The method of moving or copying text can be very clumsy unless the area of text you want to move happens to be a paragraph. Right hand justification uses excessive spacing especially if the text is within narrow margins and has been edited. I often found that if I unjustified a line I could fit the first word of the next line onto it with ease. Perhaps this is the price of the fairly quick justification process. The move command was easily the slowest, a block would be dealt with at the rate of about one line per second. I had problems when I wanted a piece of 40 column text rejustified into 120 column, my last paragraph appeared badly chewed, this may be due to the fact that I already had over 7k of text and the program needs 2k of workspace.

All of the commands are very easy to find from either the HELP window or the HELP page. Each command can be implemented by holding down either the SHIFT or the CTRL key and then pressing the relevant character key.

WILL IT WORK YOUR PRINTER?

The printer control codes are set up for the EPSON RX80 but they are all fairly easy to change for any other printer. All of the commands that your printer will accept can be sent with the text by means of inverse characters placed in the appropriate places within the text. To underline simply press CTRL & space bar and a list of the printer functions is displayed, enter J and it will appear at the cursor position as an inverse character, underline off is a lower case j. The other printer functions can be accessed in the same manner. Having spent hours hacking at my printer cable to stop the double line spacing problem I was unable to take advantage of the option that should force double spacing, instead I had to increase the line spacing.

VERDICT

The speed of the program is impressive, I was rarely bored waiting for a command to be completed. Justification of say 23 lines is complete in a little over 8 seconds. Other commands such as moving text and searching take longer but one uses them infrequently and I did not find the delay serious. The package is not comprehensive but it is very well equipped with all of the important functions, even the ability to unjustify a line has not been forgotten. At first the package may not seem that easy to operate but in a very short while I felt completely at home with it. In fact I am so confident that I intend to teach my Mother how to operate it, an otherwise daunting task. In short Tasword is excellent value and I would be surprised if a substantially better package appears.

FEATURES	5
DOCUMENTATION	4
PERFORMANCE	4
USABILITY	4
RELIABILITY	5
VALUE	5

FACTSHEET:	Tasword 464 £19.95
MACHINE:	AMSTRAD CPC464
FORMAT:	Cassette
SUPPLIER:	Tasman Software Ltd. Springfield House Hyde Terrace Leeds
OUTLETS:	Mail Order by September

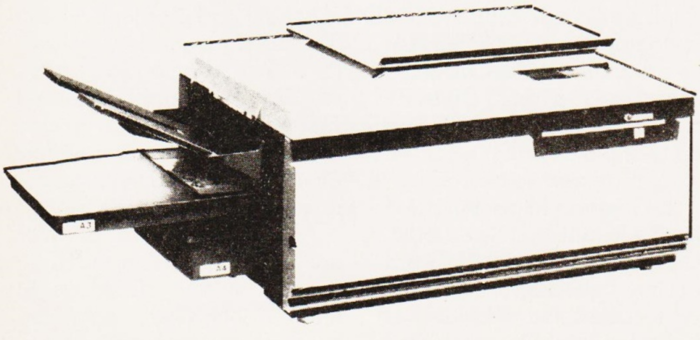
PHOTOCOPIES

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,
145 Charing Cross Road,
London WC2H 0EE.

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



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 £1.25 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.

CT PHOTOCOPIES

NAME ....

ADDRESS ....

....

.... POSTCODE ....

Please send me Photocopies of the following items

ISSUE	ARTICLE	PAGES

At £1.25 each, I enclose £ ....

Cheques and Postal Orders should be made payable to ASP Ltd. **82/10**

MAGAZINE ARTICLE INDEX

James Tyler

Spend a little time now using this useful program and save yourself hours looking for that elusive article among your crowded bookshelves.

If, like me, you're one of those people who has collected quite a large number of the various computing magazines over the last couple of years, then you've no doubt noticed the increasing difficulty experienced when trying to locate a particular article or listing which (you think), exists somewhere in your collection. If indeed, you are one of those people then read on, since the following program should be the solution to your problems. On the other hand, if you're not one of those people, read on anyway since it probably won't be long before you are!

YOUR PROBLEM SOLVED

The facilities provided by the following program allow you to basically do two things:

1. Add data to file, ie, store the necessary items of information regarding a particular article, or group of articles on disc, thus allowing the user to retrieve the required information at a later date.

2. Lookup article details, ie, retrieve the details regarding a specified article (such as magazine name, issue & page number), to allow it to be located on the bookshelf. (The speed and efficiency with which the last operation may be performed does, of course, rely greatly upon the neatness of the bookshelf in question!)

The Article Index Program has been written to run on a 32K BBC Micro equipped with at least one disc drive. Using the program on a tape based system would be impractical really since its convenience of use relies upon the fast access times of discs.

Conversion of the program onto other systems with discs should be fairly straightforward. All of the disc accessing commands used merely load or save blocks of memory — no complicated filing commands are involved that might only be specific to a few machines.

TECHNICAL DETAILS

Although it is not essential to know how the Article Index Program works, it will be of interest if you intend to modify any of the routines involved or wish to run the program on another system.

When data is added to the file concerning different articles, the program generates a 'record' for each article title which the user wishes to store on disc.

Although records may be of different sizes, they all share the same structure:

A Single Record (RE$)
TITLE+TITLE END+MAGAZINE BYTE+YEAR BYTE+PAGE BYTE+END OF RECORD BYTE
ie.
(I$)+CHR$(13)+MB%+YE%+PG%+CHR$(255)

*** MAGAZINE BYTE**
TOP 4 BITS – MAGAZINE NAME (0-15)
LOWER 4 BITS – MONTH OF ISSUE (0-11)

As records like this are generated, they are stored in a section of the 'spare memory' area which exists above the program listing. (In the program, variable S% holds the starting address of this area and L% its length in bytes). When records are stored, the contents of P% (a pointer) are examined to determine where the next free memory location is.

When the user has finished updating the file, or the spare memory area has been filled up (12K of data), all of the data between S% and S%+L% is *SAVEd as if it were machine code. The filename of this data is preceeded with 'I.' to place it in disc directory 'I', and the actual filename is just a single digit number of zero or more. Whenever the entire 12K of data in one disc file is filled up, the number represeting the latest filename (F%) is incremented by one ande the pointer P% is reset to zero. This effectively starts a new file for data storage.

The number of the most recent filename being used and the current value of the free memory pointer are both stored in a short file called INFO. This block of data loaded into the machine whenever the program is run, and is updated and resaved back to disc whenever new data has been added.

SEARCHING FOR DATA

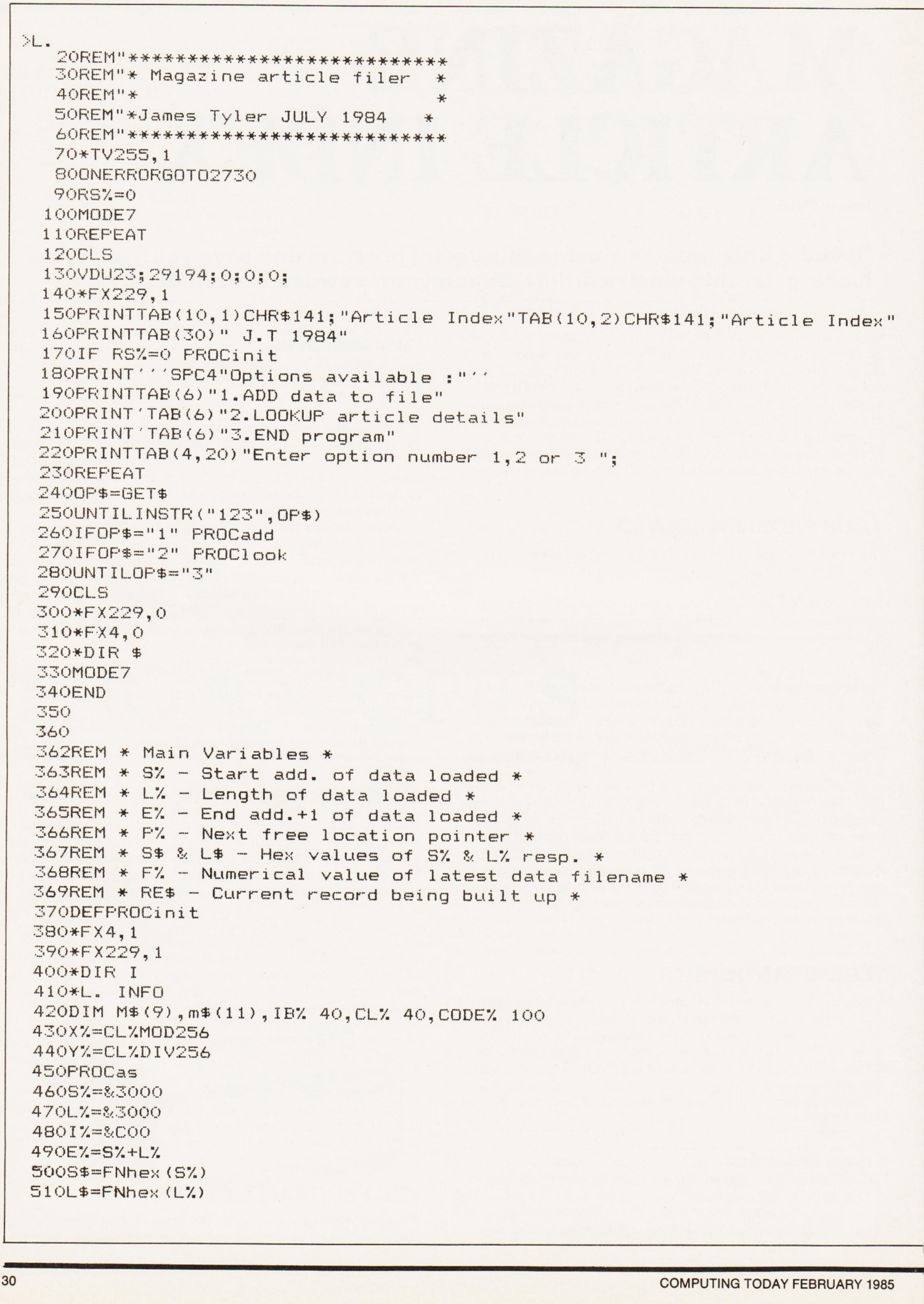
When the user wishes to search for details about a particular article, he or she merely has to type in a word or part of a word which exists in the title. The program will respond by searching through all of the titles saved to dated to see if any contain the word entered by the user. If any such titles are found they are printed out in whole together with the corresponding magazine name, the month and year of that issue, and page number.

The program lists relevant article details by loading each numbered data file (in numerical order) in turn and searching all the titles stored, to see if they contain the word entered by the user. For speed, the actual searching is done using a machine code routine. Whenever the specified word is found, the program backtracks to the beginning of the title and the entire heading is printed, along with the magazine details which are, of course, stored in the following bytes making up the rest of that record. The search is repeated until all of the data file(s) have been loaded and looked through.

The result of all this is a pretty fast lookup routine,which owes its speed to a combination of a machine code search doing the donkey work and the use of discs for data storage.

THE PROGRAM LISTING

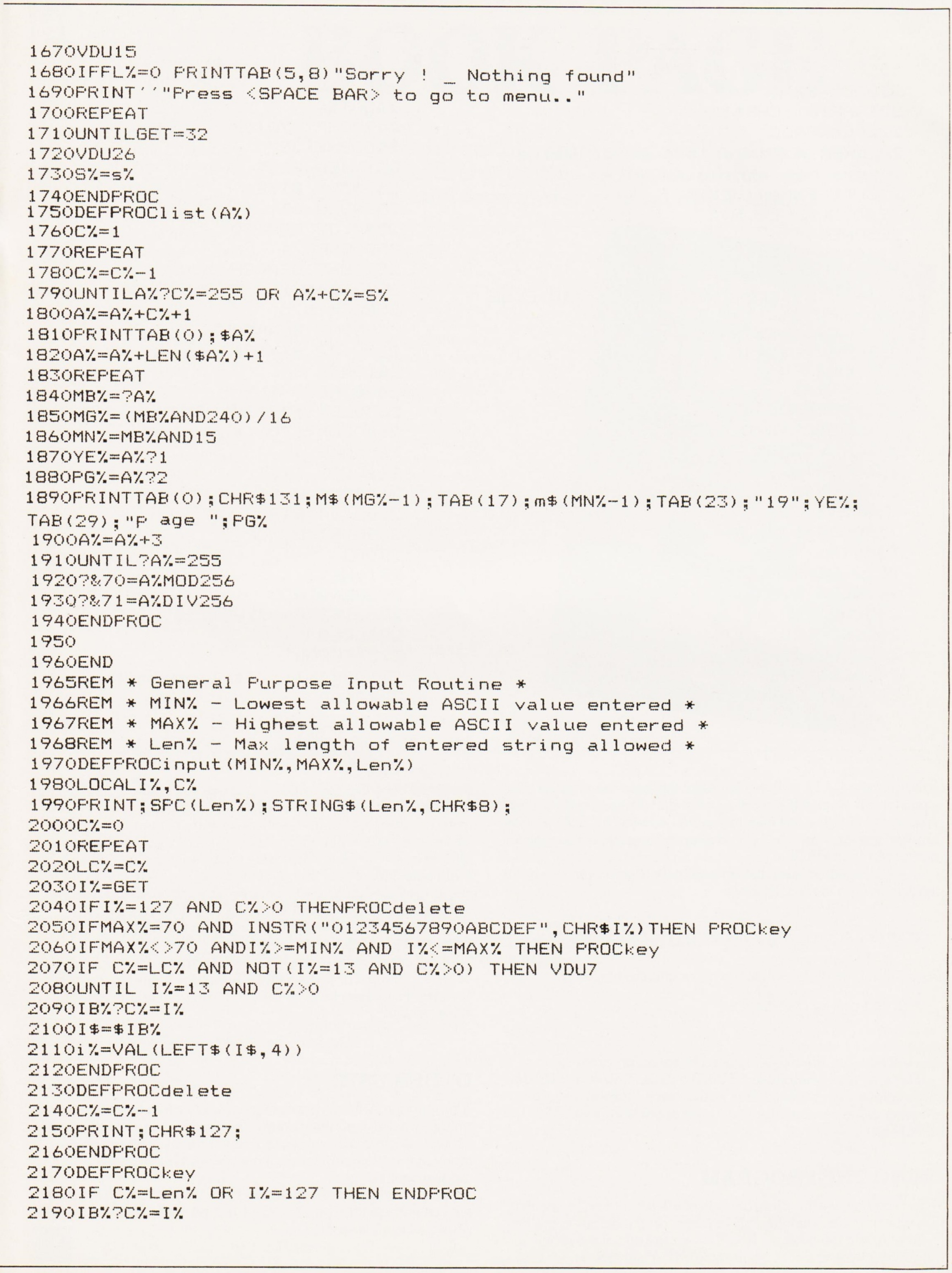
When you type in the listing for the Article Index Program, don't forget to put the names of the magazines that you are likely to



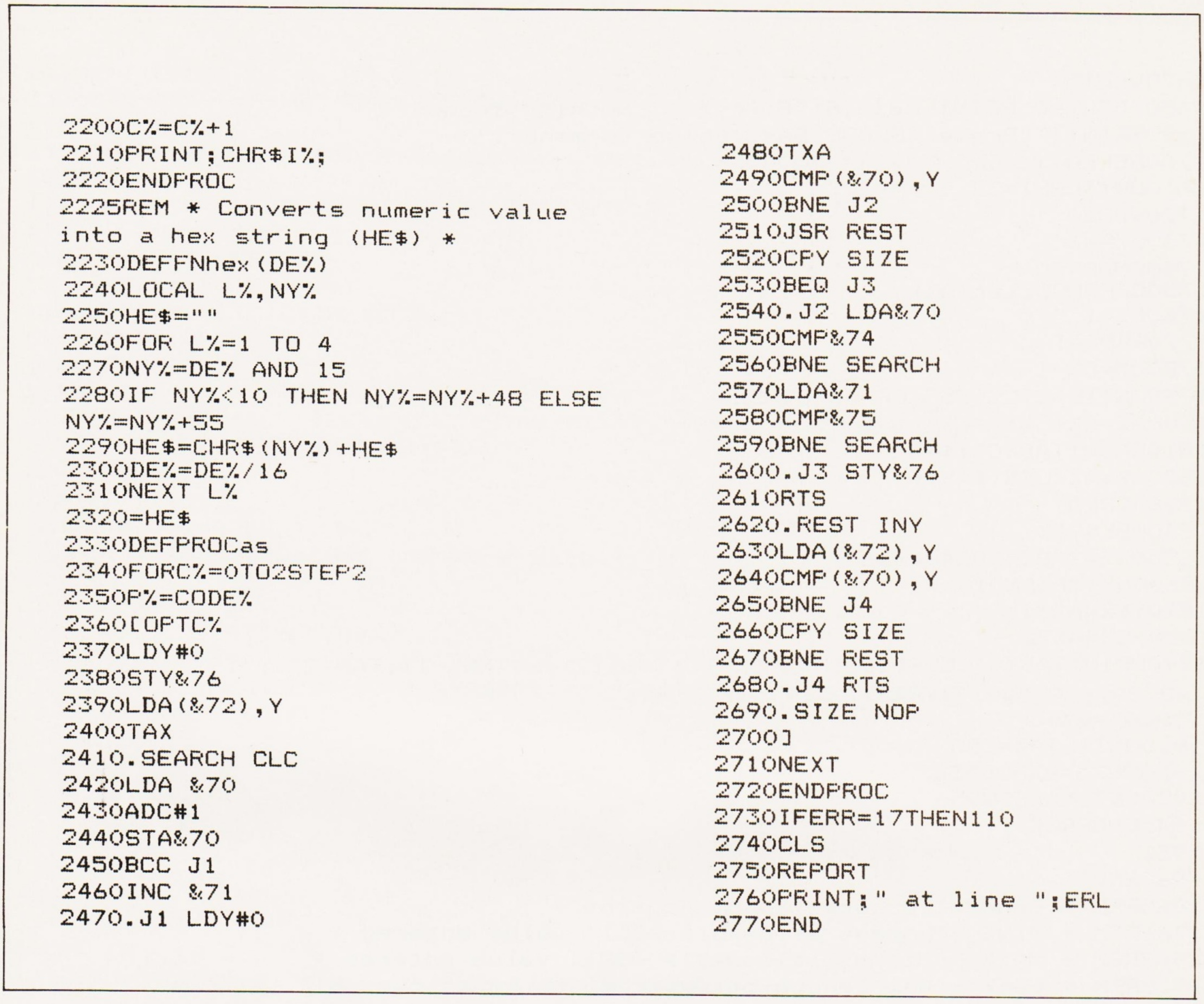
```
>L.
   20REM"****************************
   30REM"* Magazine article filer   *
   40REM"*                          *
   50REM"*James Tyler JULY 1984    *
   60REM"****************************
   70*TV255,1
   80ONERRORGOTO2730
   90RS%=0
  100MODE7
  110REPEAT
  120CLS
  130VDU23;29194;0;0;0;
  140*FX229,1
  150PRINTTAB(10,1)CHR$141;"Article Index"TAB(10,2)CHR$141;"Article Index"
  160PRINTTAB(30)" J.T 1984"
  170IF RS%=0 PROCinit
  180PRINT'''SPC4"Options available :"''
  190PRINTTAB(6)"1.ADD data to file"
  200PRINT'TAB(6)"2.LOOKUP article details"
  210PRINT'TAB(6)"3.END program"
  220PRINTTAB(4,20)"Enter option number 1,2 or 3 ";
  230REPEAT
  240OP$=GET$
  250UNTILINSTR("123",OP$)
  260IFOP$="1" PROCadd
  270IFOP$="2" PROClook
  280UNTILOP$="3"
  290CLS
  300*FX229,0
  310*FX4,0
  320*DIR $
  330MODE7
  340END
  350
  360
  362REM * Main Variables *
  363REM * S% - Start add. of data loaded *
  364REM * L% - Length of data loaded *
  365REM * E% - End add.+1 of data loaded *
  366REM * P% - Next free location pointer *
  367REM * S$ & L$ - Hex values of S% & L% resp. *
  368REM * F% - Numerical value of latest data filename *
  369REM * RE$ - Current record being built up *
  370DEFPROCinit
  380*FX4,1
  390*FX229,1
  400*DIR I
  410*L. INFO
  420DIM M$(9),m$(11),IB% 40,CL% 40,CODE% 100
  430X%=CL%MOD256
  440Y%=CL%DIV256
  450PROCas
  460S%=&3000
  470L%=&3000
  480I%=&C00
  490E%=S%+L%
  500S$=FNhex(S%)
  510L$=FNhex(L%)
```

```
 520F%=?I%
 530P%=I%!1
 540LF%=0
 550TI$=STRING$(39,CHR$32)
 560RE$=STRING$(76,CHR$32)
 570C%=0
 580REPEAT
 590READG$
 600M$(C%)=G$
 610C%=C%+1
 620UNTILG$="*"
 630NM%=C%-2
 640FORC%=0TO11
 650READm$(C%)
 660NEXT
 670RS%=1
 680ENDPROC
 690
 700
 710DEFPROCadd
 720F%=?I%
 730P%=I%!1
 740IFS%+P%+76>E% F%=F%+1:P%=0 ELSEIFF%+P%>0 THEN$CL%="L."+STR$(F%):CALL&FFF7
 750REPEAT
 760CLS
 770PRINT'TAB(0)"Title ?"
 780PROCinput(32,127,39)
 790RE$=I$+CHR$13
 800VDU28,0,24,39,5
 810C%=0
 820REPEAT
 830CLS
 840PRINTTAB(0,0)"Magazine name ?(Use ← →cursor keys)"
 850c%=0
 860REPEAT
 870PRINTTAB(0,1)M$(c%);
 880G%=GET
 890IFG%=&88 c%=c%-1:IFc%<0 c%=NM%
 900IFG%=&89 c%=c%+1:IFc%>NM% c%=0
 910UNTILG%=13
 920MG$=M$(c%)
 930MG%=c%+1
 935REM * You're Own Choice Here ! *
 940DATACOMPUTING TODAY,"A&B COMPUTING  ","PERSONAL S/WARE","ORCHARD COMPTNG",*
 950REPEAT
 960PRINTTAB(0,2)"Month ?(1_12)"
 970PROCinput(48,58,2)
 980UNTILi%>0 ANDi%<13
 990MN%=i%
1000PRINTTAB(0,3)m$(MN%-1)
1010DATAJAN,FEB,MARCH,APRIL,MAY,JUNE,JULY,AUGUST,SEPT,OCT,NOVE,DEC
1020MB%=MG%*16+MN%
1030PRINTTAB(0)"Year ?"'"19";
1040PROCinput(48,58,2)
1050YE%=i%
1060REPEAT
1070PRINTTAB(0,6)"Page number ?"
1080PROCinput(48,58,3)
1090PG%=i%
```

```
1100UNTILPG%<256
1110RE$=RE$+CHR$MB%+CHR$YE%+CHR$PG%
1120C%=C%+1
1130IFC%<10PRINT''"More magazines under this title (Y/N)";:REPEAT:
I$=GET$:UNTIL INSTR("YN",I$)
1140UNTILI$="N"ORC%=10
1150VDU26
1160RE$=RE$+CHR$255
1170FORC%=1TOLEN(RE$)
1180S%?P%=ASC(MID$(RE$,C%,1))
1190P%=P%+1
1200NEXT
1210LF%=1
1220IFS%+P%+76>E% $CL%="S. "+STR$(F%)+" "+S$+" +"+L$:CALL&FFF7
1230IFS%+P%+76>E% F%=F%+1:P%=0
1240?I%=F%
1250I%!1=P%
1260CLS
1270PRINTTAB(3,10)"Add any more data (Y/N) ?";
1280REPEAT
1290G$=GET$
1300UNTILINSTR("YN",G$)
1310UNTILG$="N"
1320IFP%>0 THEN$CL%="S. "+STR$(F%)+" "+S$+" +"+L$:CALL&FFF7
1330*S."INFO"C00 CFF
1340ENDPROC
1350
1360DEFPROClook
1370*FX229,0
1380F%=0
1390FL%=0
1400s%=S%
1410S%=S%-1
1420VDU12,14
1430PRINTTAB(0,0)"Enter the word(s) to search for:"
1440PROCinput(32,127,39)
1450CLS
1460PRINTTAB(0,0)I$
1470VDU28,0,24,39,2
1480VDU23;8202;0;0;0;
1490?SIZE=LEN(I$)
1500!&72=CL%
1510REPEAT
1520?&70=S%MOD256
1530?&71=S%DIV256
1540IFF%=?I% !&74=S%+P% ELSE!&74=E%
1550IFLF%=1 AND?I%=0THEN1590
1560LF%=1
1570$CL%="L."+STR$(F%)
1580CALL&FFF7
1590F%=F%+1
1600$CL%=I$
1610REPEAT
1620CALLCODE%
1630IF?&76=LEN(I$) PROClist(?&70+256*?&71):FL%=1
1640UNTIL?&70+256*?&71>=?&74+256*?&75
1650UNTILF%>?I%
1660S%=S%-1
```



```
1670VDU15
1680IFFL%=0 PRINTTAB(5,8)"Sorry ! _ Nothing found"
1690PRINT''"Press <SPACE BAR> to go to menu.."
1700REPEAT
1710UNTILGET=32
1720VDU26
1730S%=s%
1740ENDPROC
1750DEFPROClist(A%)
1760C%=1
1770REPEAT
1780C%=C%-1
1790UNTILA%?C%=255 OR A%+C%=S%
1800A%=A%+C%+1
1810PRINTTAB(0);$A%
1820A%=A%+LEN($A%)+1
1830REPEAT
1840MB%=?A%
1850MG%=(MB%AND240)/16
1860MN%=MB%AND15
1870YE%=A%?1
1880PG%=A%?2
1890PRINTTAB(0);CHR$131;M$(MG%-1);TAB(17);m$(MN%-1);TAB(23);"19";YE%;
TAB(29);"P age ";PG%
1900A%=A%+3
1910UNTIL?A%=255
1920?&70=A%MOD256
1930?&71=A%DIV256
1940ENDPROC
1950
1960END
1965REM * General Purpose Input Routine *
1966REM * MIN% - Lowest allowable ASCII value entered *
1967REM * MAX% - Highest allowable ASCII value entered *
1968REM * Len% - Max length of entered string allowed *
1970DEFPROCinput(MIN%,MAX%,Len%)
1980LOCALI%,C%
1990PRINT;SPC(Len%);STRING$(Len%,CHR$8);
2000C%=0
2010REPEAT
2020LC%=C%
2030I%=GET
2040IFI%=127 AND C%>0 THENPROCdelete
2050IFMAX%=70 AND INSTR("01234567890ABCDEF",CHR$I%)THEN PROCkey
2060IFMAX%<>70 ANDI%>=MIN% AND I%<=MAX% THEN PROCkey
2070IF C%=LC% AND NOT(I%=13 AND C%>0) THEN VDU7
2080UNTIL I%=13 AND C%>0
2090IB%?C%=I%
2100I$=$IB%
2110i%=VAL(LEFT$(I$,4))
2120ENDPROC
2130DEFPROCdelete
2140C%=C%-1
2150PRINT;CHR$127;
2160ENDPROC
2170DEFPROCkey
2180IF C%=Len% OR I%=127 THEN ENDPROC
2190IB%?C%=I%
```



```
2200C%=C%+1
2210PRINT;CHR$I%;
2220ENDPROC
2225REM * Converts numeric value
into a hex string (HE$) *
2230DEFFNhex(DE%)
2240LOCAL L%,NY%
2250HE$=""
2260FOR L%=1 TO 4
2270NY%=DE% AND 15
2280IF NY%<10 THEN NY%=NY%+48 ELSE
NY%=NY%+55
2290HE$=CHR$(NY%)+HE$
2300DE%=DE%/16
2310NEXT L%
2320=HE$
2330DEFPROCas
2340FORC%=0TO2STEP2
2350P%=CODE%
2360[OPTC%
2370LDY#0
2380STY&76
2390LDA(&72),Y
2400TAX
2410.SEARCH CLC
2420LDA &70
2430ADC#1
2440STA&70
2450BCC J1
2460INC &71
2470.J1 LDY#0
2480TXA
2490CMP(&70),Y
2500BNE J2
2510JSR REST
2520CPY SIZE
2530BEQ J3
2540.J2 LDA&70
2550CMP&74
2560BNE SEARCH
2570LDA&71
2580CMP&75
2590BNE SEARCH
2600.J3 STY&76
2610RTS
2620.REST INY
2630LDA(&72),Y
2640CMP(&70),Y
2650BNE J4
2660CPY SIZE
2670BNE REST
2680.J4 RTS
2690.SIZE NOP
2700]
2710NEXT
2720ENDPROC
2730IFERR=17THEN110
2740CLS
2750REPORT
2760PRINT;" at line ";ERL
2770END
```

collect, in the DATA at line 940. All of these names must be the same length, so pad the shorter ones with with spaces if required. Also, make sure the last item in the DATA line is a '*'. As the listing is at the moment, you can have up to ten different magazine names stored in DATA.

SAVE the listing onto disc immediately (a new one if possible), then type in the following:

```
FOR I%=&C00 to &CFF:?I%=0: NEXT <RETURN>
```

Check the same disc is still in the drive and then type:

```
*SAVE I.INFOR C00 CFF
```

This will initialise the INFO file used by the main program.

You may find it helpful to *BUILD a !BOOT file that CHAINs the Article Index Program. This would mean that running the program in future would just be a case of pressing <SHIFT BREAK>.

USING THE PROGRAM

When you first use the program, select the first option on the displayed menu. You can then add the first article details to the file.

In return to the prompt 'Title ?', enter the title of the article in question. A tip is to put in as many words or phrases as possible that help describe the contents and nature of the article. This will increase the chances of the right information being found when you are later looking up details. The entry of the title (upto 39 characters) is completed by pressing <RETURN>. After this, simply enter the magazine name (selected using the left and right cursor keys), the month of the issue, the year and the starting page number. You will be given the option of entering the details of several magazines under one article heading too.

When you have finished adding data to the file, it will be stored in its updated form back onto the disc.

Looking up data in the file, as said earlier, just requires you to type in a word or phrase that you hope is in the article or feature you're looking for. The program will then list all possible titles for you.

OTHER USES

Although the Article Index program was written for filing computing articles, it could of course, be used for magazines on any subject; electronics, cooking, car mechanics — or probably many of the other subjects that are covered by hobbyist magazines (!?)

Well, that's my part of the job done. It's just up to you to put all those article titles on disc now (and, of course, make sure that your bookshelf is tidy!)

BOOK PAGE

Garry Marshall

'The mainstream of computing' is the phrase to sum up this month's selection of books, which range from LOGO to microcomputer control applications.

The publishers of computer books are rather conservative in their choice of books to publish. Looking back through the Computing Today book reviews of the past twelve months reveals that for each of the books considered this month at least two very similar ones have been reviewed in that time.

But if this month's review books are all of a similar type, they are all good or, at the least, interesting examples of their types. Also, taken together they spell out a theme that describes a mainstreem of computer activities and developments.

The first book provides an introduction to specific personal computers and to programming them in BASIC. The market seems to show an insatiable demand for such books, and they seem to be regarded as providing an introduction to computer literacy. In a way they do, of course, but an awareness of word processors, databases and spreadsheets, of how to use them, and of what they can be used for is quite as important. In other words a familiarity with the major programs that have already been written for the computer can make it much more useful than if you only know how to program it.

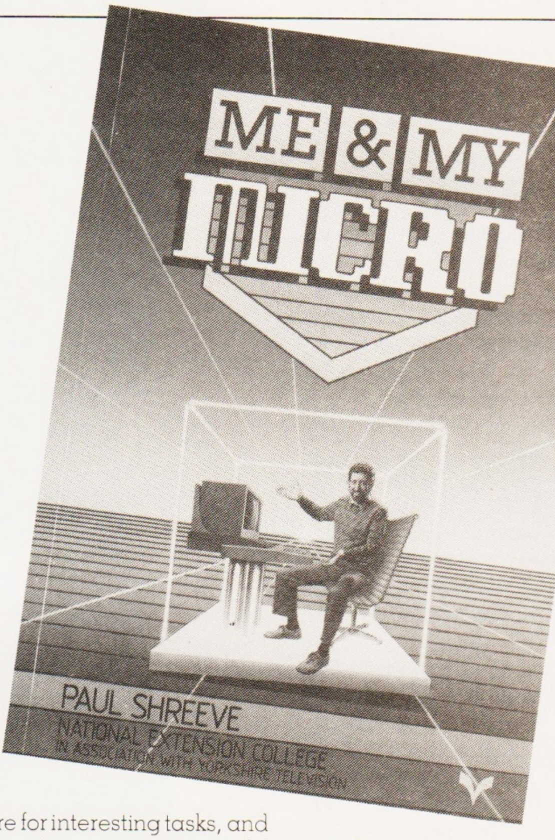
Many of the developments that are involved in making the computer easier to use are being achieved by programming the computer in higher level languages than BASIC. These include making it accept speech input and giving it the capability to understand ordinary speech. Again, there is a contrast between programming the computer for these developments and using them. It is rarely articulated that only a few people need to be able to write the programs. The rest of us need only know how to make use of them.

Our second book is about LOGO. This isn't exactly a super-high-level language, although it's level is higher than that of BASIC. It can be used to convert the computer into a device that allows children (and anyone else) to learn and, further, to learn in the same natural and unforced way as they learn to speak. The book presents some nicely anarchic images of children learning by using computers without the interference, or even the presence, of teachers.

The techniques upon which making computers easier to use are based are drawn from artificial intelligence studies. It has been said that artificial intelligence is the study of how to make the computer do things that we do not yet know how to make it do. And also that as soon as we do know how to do something, that thing moves from being a research topic to being commonplace. Artificial intelligence programs are expressions of the theories of the practitioners of AI; they are seldom, if ever, written in BASIC. LISP and PROLOG are the *lingua franca* here. But our third book presents BASIC programs for some of the simpler activities that have become possible as a result of the findings of AI.

Another area that artificial intelligence has contributed to in a considerable way is robotics. Here the computer is used to provide the intelligence for, and to control, a device capable of performing physical actions. The fourth book is concerned with control, explaining very clearly how a micro can be used for this purpose.

Our final book provides an overview of personal computing, but it is disappointingly lacking in any coverage of more recent developments in computing. This is apart from the fact that, as more than one of the other books points out, the developments in the personal computer sphere can be very deceptive if taken to represent what is happening in computing generally.


ME & MY MICRO
PAUL SHREEVE
NATIONAL EXTENSION COLLEGE
IN ASSOCIATION WITH YORKSHIRE TELEVISION

Me and My Micro by Paul Shreeve is designed to accompany Yorkshire Television's series of the same name. But the book can be read without seeing the TV series. It provides an introduction to using the BBC, Electron and Spectrum, and then shows how to program them in BASIC. It follows a, by now, very well-trodden path, but it does so with competence. The programs that are presented are for interesting tasks, and they contain some very useful techniques, so there is a good deal to be learnt from them by the beginner. Starting with simple games programs for moving targets, firing lasers and hitting targets, it moves on to mazes and maze running, showing how graphics can be introduced to advantage. It then moves from games to anagrams and finally to a program for Pelmanism. The last leads to the development of a substantial program for which the reader can observe the sensible development and structuring.

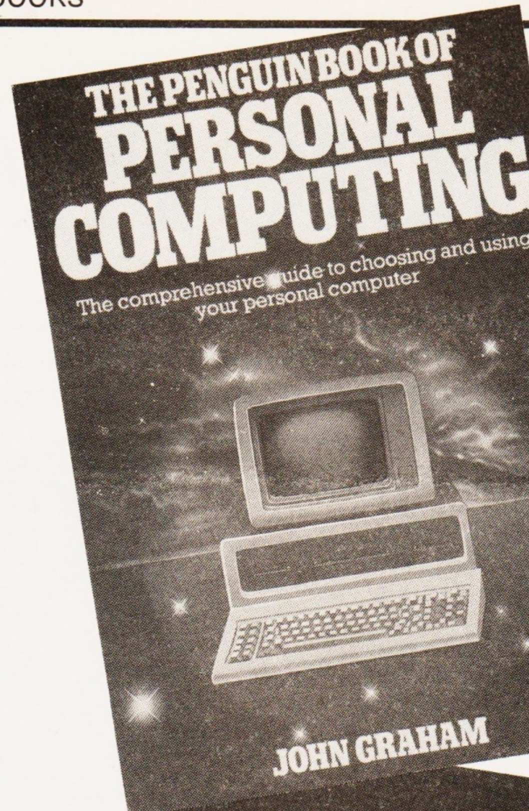
But the book has no really clear aims and objectives, and one is left wondering where one has got to and what there is to do next. Perhaps the programmers in Yorkshire can write BASIC programs to control automatic coal-winning equipment. I doubt it somehow, though, and not just on computing grounds.

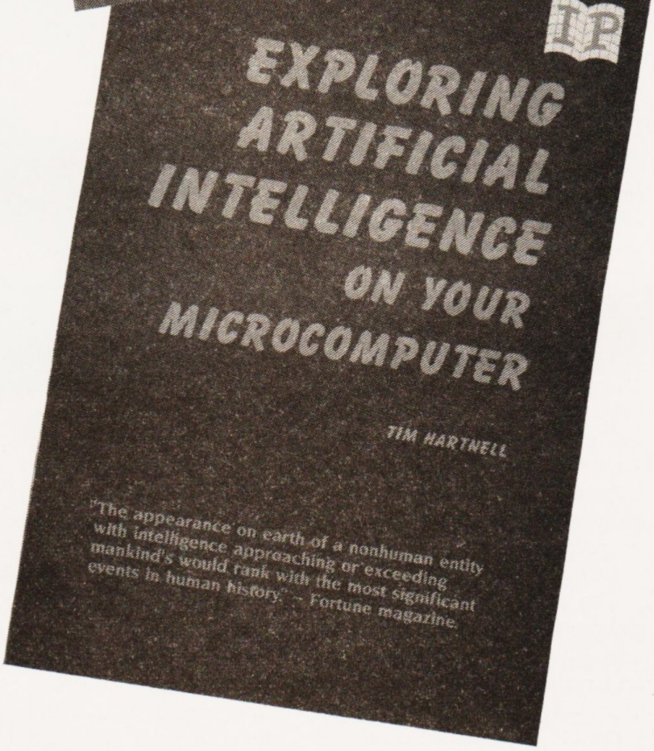
Logo — a guide to learning through programming by Peter Goodyear contains as its core an introduction to programming in LOGO. The chapters surrounding this core are much more interesting, though, in explaining the author's views on the educational value and application of LOGO. I feel that there is little new in what he says, but everything that he presents is clearly drawn from his own experience and all the ideas are moulded by his struggle to articulate the value of LOGO for learning and then to persuade his readers. As the author admits, the book is a paradox, for he feels that LOGO is something to experiment with and experience rather than to be taught in a formal way. He would like to see children left to their own devices with LOGO to explore as they will. In this way, they can learn Mathematics from a 'Mathsland' supported by LOGO in the same way as they naturally learn a language. In the same way as adults know language and surround the child with experience of it, interaction in it and correction in its use, so the computer can do the same with Maths, and other topics.

Allowing children to explore and learn in an undirected way will arouse certain worries. In the primary school, where there are no constraints from examinations and their syllabi, it may not be such a problem. Researchers investigating how children learn with LOGO claim to have identified the three general stages into which learning can be divided in the way that children use LOGO. These are the 'enactive' stage of learning by physically manipulating objects, the 'iconic' stage of learning from images, and the 'symbolic' stage of learning with language. LOGO can provide each stage and transitions between them. Moving a floor turtle with individual commands *is* to physically manipulate an object. Writing a program in LOGO is to learn with language, in particular, how to describe some future actions that a turtle must take.

Although the identification of children's development with LOGO with general learning development is probably too pat it does hint at the power of LOGO to provide a natural learning environment. It is possible that classrooms of the future will contain only computers and that teachers may not be needed?

To complement this book, **LOGO In The Classroom** by Shirley Torgerson contains many ideas for activities, projects and problems to solve with the use of LOGO. Actually, the ideas could be used in the home quite as well as in the classroom. (Goodyear remarks that the place for LOGO may be the home, not least because of the availability of a computer there on a one-to-one basis, which is far from likely in school). Also, the projects are often as suitable for BASIC as for LOGO, so this book can be seen as a plentiful source of ideas for what to do with any computer anywhere.


THE PENGUIN BOOK OF
PERSONAL
COMPUTING
The comprehensive guide to choosing and using your personal computer
JOHN GRAHAM


IP
EXPLORING
ARTIFICIAL
INTELLIGENCE
ON YOUR
MICROCOMPUTER
TIM HARTNELL
"The appearance on earth of a nonhuman entity with intelligence approaching or exceeding mankind's would rank with the most significant events in human history." — Fortune magazine.

Exploring Artificial Intelligence On Your Computer by Tim Hartnell gives me the impression that its author has read a few books on artificial intelligence to learn about it, and has then decided to write a book embodying the simpler things that he has read about and can write BASIC programs for. An appendix contains a list of AI books for further reading. It would be far better to read a few of the books given here for enlightenment on AI matters than to read the bits between the programs in this book.

The value of the book, though, is in its programs. There are BASIC programs for understanding natural language, for learning and for an expert system. Their development is described in each case. They all work in their own fashion and, as long as you do not expect much of them, are quite impressive.

Micros In Control by Gerald Bettridge describes in terms that are just about as simple as they possibly could be how to attach, and control, items such as lamps, motors and stepper motors to a micro. It deals with 6502-based and Z-80 based micros, giving assembly code routines for each, but gives a rather dated feel with references to Acorn Atoms, PETs and the Jupiter Ace. In the 80 smqll pages of the body of its text it deals with several projects and rather a lot of computers, and so the treatments are necessarily brief. A final section on the Ace is mercifully brief given the fate of that machine. But some of the projects are both interesting and useful. In all, this provides a fair introduction to the use of micros for control that requires as little expertise in electronics as it is possible to imagine.

The Penguin Book Of Personal Computing by John Graham is, like many of Penguin's computer books, a disappointment to me. My judgement is basically unfair since there is nothing wrong with the book. It provides a summary of computing activities biased towards personal computer user. But it is too conventional, too conservative and, for a book published in 1984, too far behind the game. One has always read Penguins and so come to associate a standard with Penguin books, which this one does not meet.

Its chapter on languages covers BASIC and COBOL, but doesn't mention LISP and PROLOG. LOGO isn't mentioned, and although there is a brief section on computers in education, it hardly strays from distance learning and computer-assisted instruction. The chapter on personal computers in business just mentions spreadsheets, without really saying what they are, and doesn't mention databases at all. (They are mentioned in the following chapter on networks). Artificial intelligence doesn't rate a mention anywhere.

I would summarise the book by saying that it gives a good account of where we've been, a sketchy one of where we are, and nothing of where we might be going.

This month's books are:

Me and My Micro by Paul Shreeve (NEC with Yorkshire Television) 115 pages, £2.95.

LOGO — A Guide to Learning Through Programming by Peter Goodyear (Ellis-Horwood) 206 pages, £6.50.

LOGO In The Classroom by Shirley Torgerson (ICCE Publications) 202 pages.

Exploring Artificial Intelligence On Your Microcomputer by Tim Hartnell (Interface Publications) 357 pages, £4.95.

Micros In Control by Gerald Bettridge (Longman) 96 pages, £4.95.

The Penguin Book Of Personal Computing by John Graham (Penguin) 304 pages, £3.95.

NEXT MONTH

computing today

MARCH ISSUE
ON SALE
FEBRUARY 8TH

In next month's *Computing Today,* we take a hard look at the principles behind memory expansion, and the ways in which memory 'paging' is used to extend the normal addressable range of a microprocessor. And next issue we throw open the doors of the all-new *Computing Today* Algorithm Department, a section of the magazine entirely devoted to tried and trusted algorithmic solutions to important problems. We begin by looking at ways to add missing trigonometrical functions to your machine. Our feature on printers and printer interfaces will be invaluable to all those in search of a printer, or just as a useful reference-sheet, and . . . Toshiba MSX reviewed. Yes! *Computing Today* gets its very first MSX machine (not counting the Spectravideo) to look at. Also reviewed is the Interlekt Portman modem — a nice device by all accounts — AND we offer an enlightening introduction to modems and their uses. For a little light relief we will be publishing, for the very first time, Horace Grows Up; an astonishingly nightmarish tale of a computer that . . . Well. Wait and see . . .

GIVE THIS TO YOUR NEWSAGENT
Please reserve me a regular copy of
Computing Today

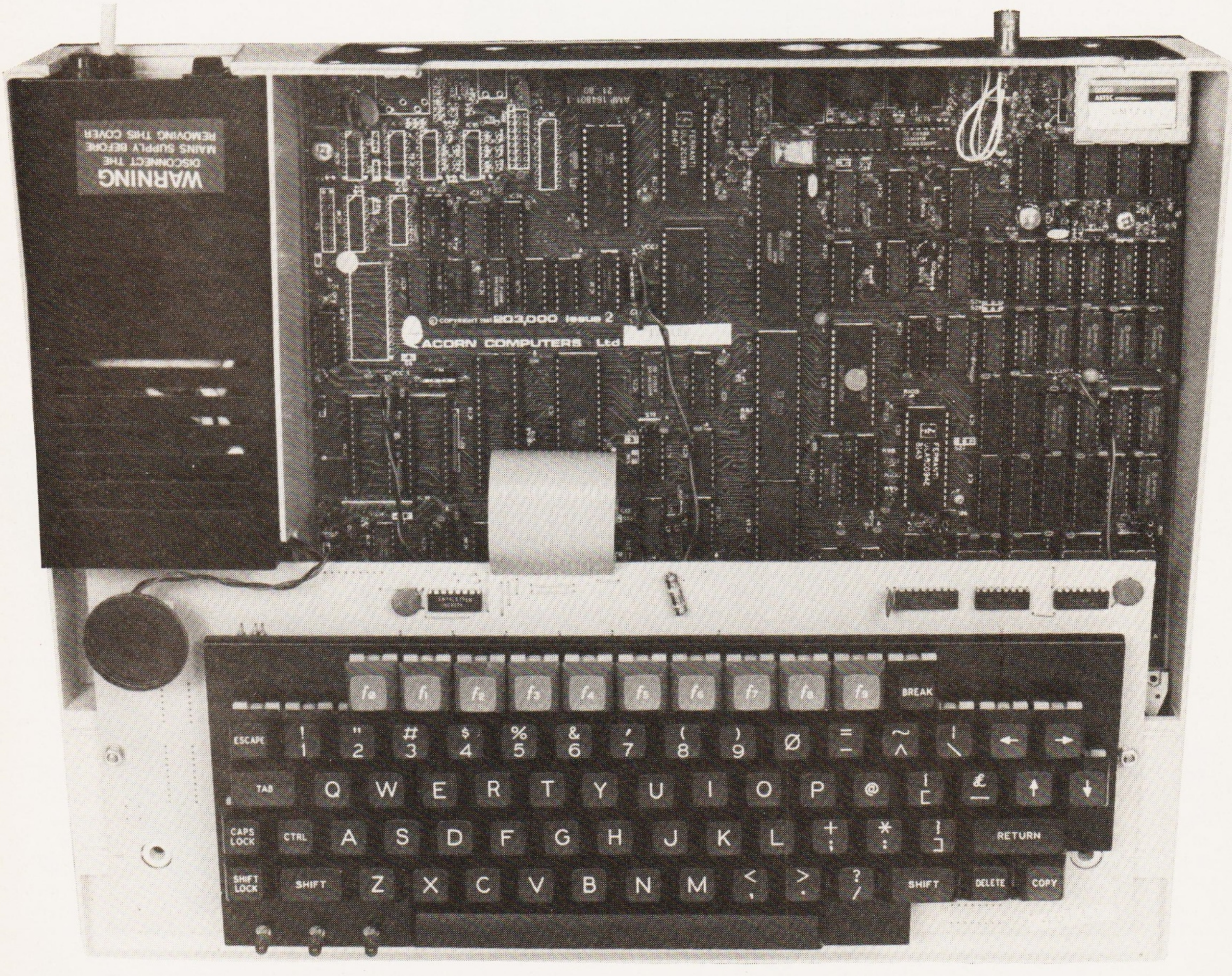
NAME ....
ADDRESS ....
POSTCODE ....

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

BBC MACHINE CODE MONITOR

John Owen

Although the BBC micro sports a remarkably usable machine code assembler, it is almost mysterious to discover that the machine does not carry a resident firmware monitor program. We rectify this deficiency this issue, with a monitor program which occupies less than 600 bytes of your precious RAM.



Using the BBC micro's built-in assembler makes it very simple to write machine-code programs. Unfortunately, the same cannot be said for de-bugging the completed program when it fails to work as expected. Even short and apparently simple programs written in machine-code often produce unexpected results when they are run and, without a means of easily examining and modifying individual memory locations, it can be very difficult to locate the source of the trouble — especially for those of us who are new to 6502 machine-code. There are, of course, many aids available for the machine-code programmer ranging from plug-in "toolkit" ROMs to extensive utility routines written in BASIC. It is difficult to justify the initial expense of one of the ROMs if only a small percentage of your programming is in machine-code and I find that utility routines written in BASIC are very inconvenient to use because of the constant necessity to change "PAGE". In addition, they tend to take up a disproportionately large amount of memory leaving little for your program's source code (ie the assembler listing) and the object code (ie the machine-code itself).

The utility program presented here is written entirely in machine-code enabling it to be 'tucked-away' somewhere out of harm's way. I have deliberately kept it relatively short (and simple) so that it occupies as little of the Beeb's precious memory as possible. The assembled code occupies a little over half a kilobyte and I usually assemble it at &900 where it uses memory from &900 to &B54. Pages &9 and &A are used mainly by the RS423 and cassette systems and page &B is used to store the 'function key' definitions. If your machine-code program is to use any of these functions then the 'Monitor' will have to be LOADed elsewhere. One possibility is at &8A0. This leaves page &B clear but does occupy the printer buffer. Do not be tempted to assemble at &800, by the way. This area is used by the Sound system and assembling the Monitor here will produce some interesting — but totally useless — effects!

Once you have selected an area suitable for your purposes, you should save a machine-code file of the assembled program with, for example:

***SAVE MONITOR 08A0 0AFD**

It can then be run from BASIC by typing *RUN MONITOR or, if you have discs, by simply typing *MONITOR.

It is also worth saving the assembler code version as a BASIC program so that you can re-assemble it to a different location should the need arise.

When the Monitor is *RUN, it sets the 'resident integer variable' M% to its own start address so that, when you quit the Monitor and go back to BASIC, you can easily call up the Monitor again by typing CALL M%. This means that you don't have to remember the Monitor's start address.

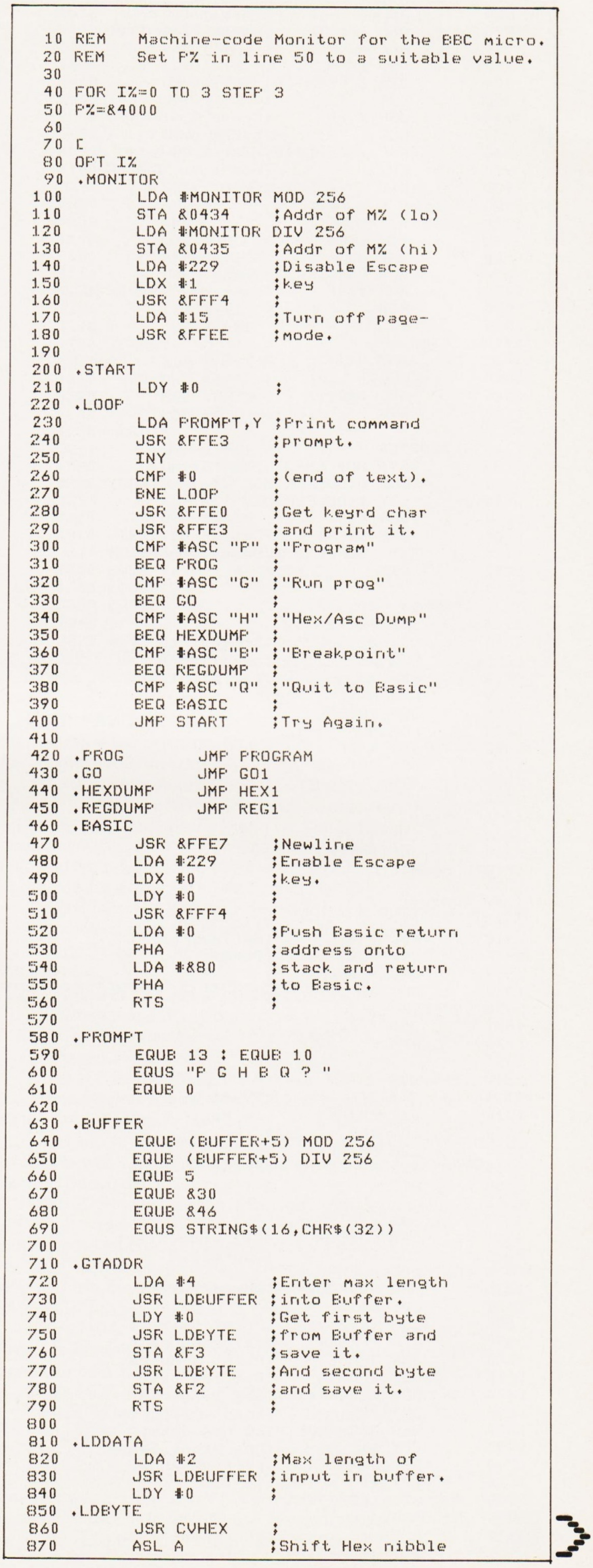
USING THE MONITOR

*RUN the Monitor and the screen will display:

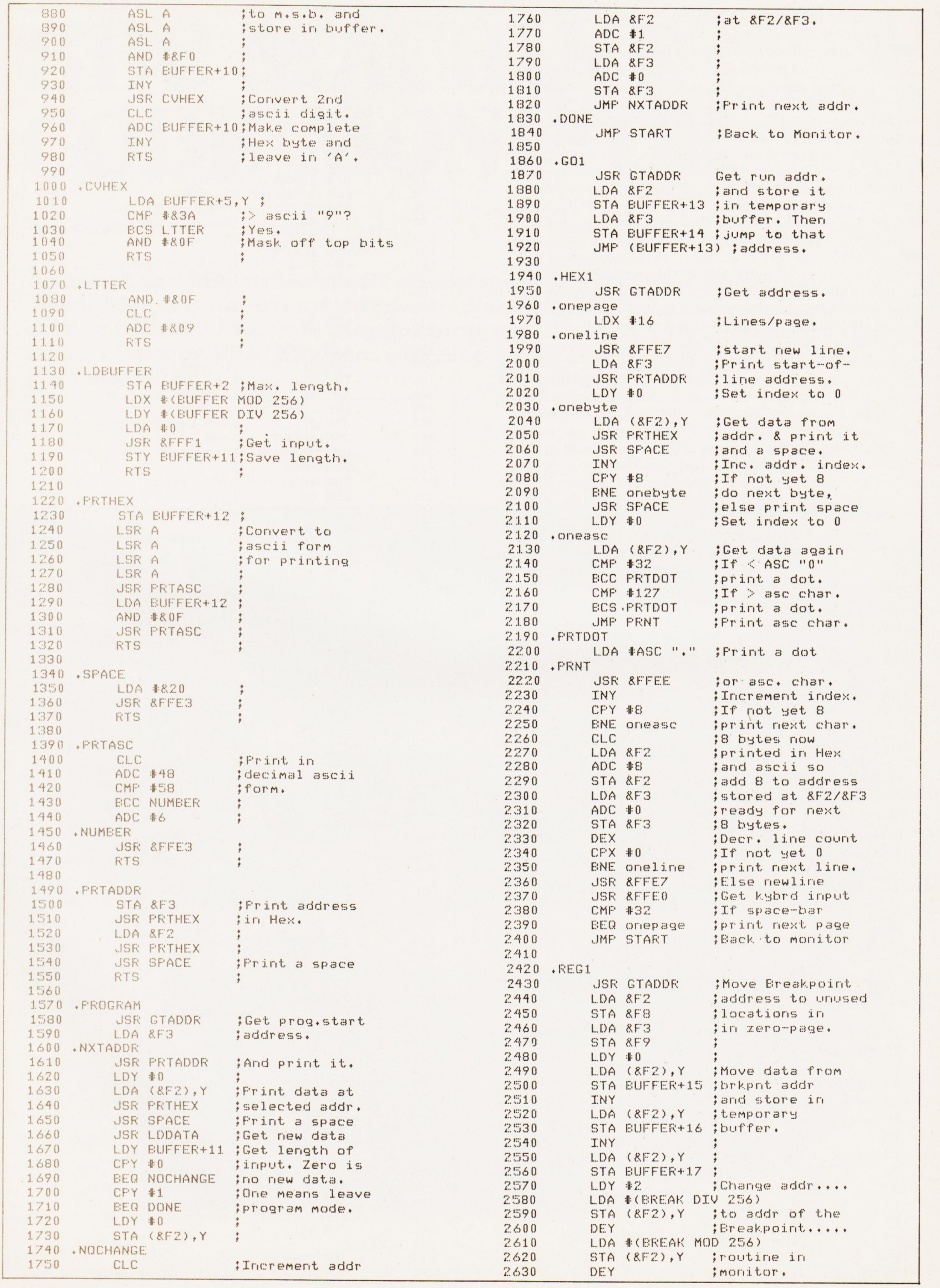
```
PGHBQ?_
```

Type Q to quit back to BASIC and type in the following program by way of a demonstration:

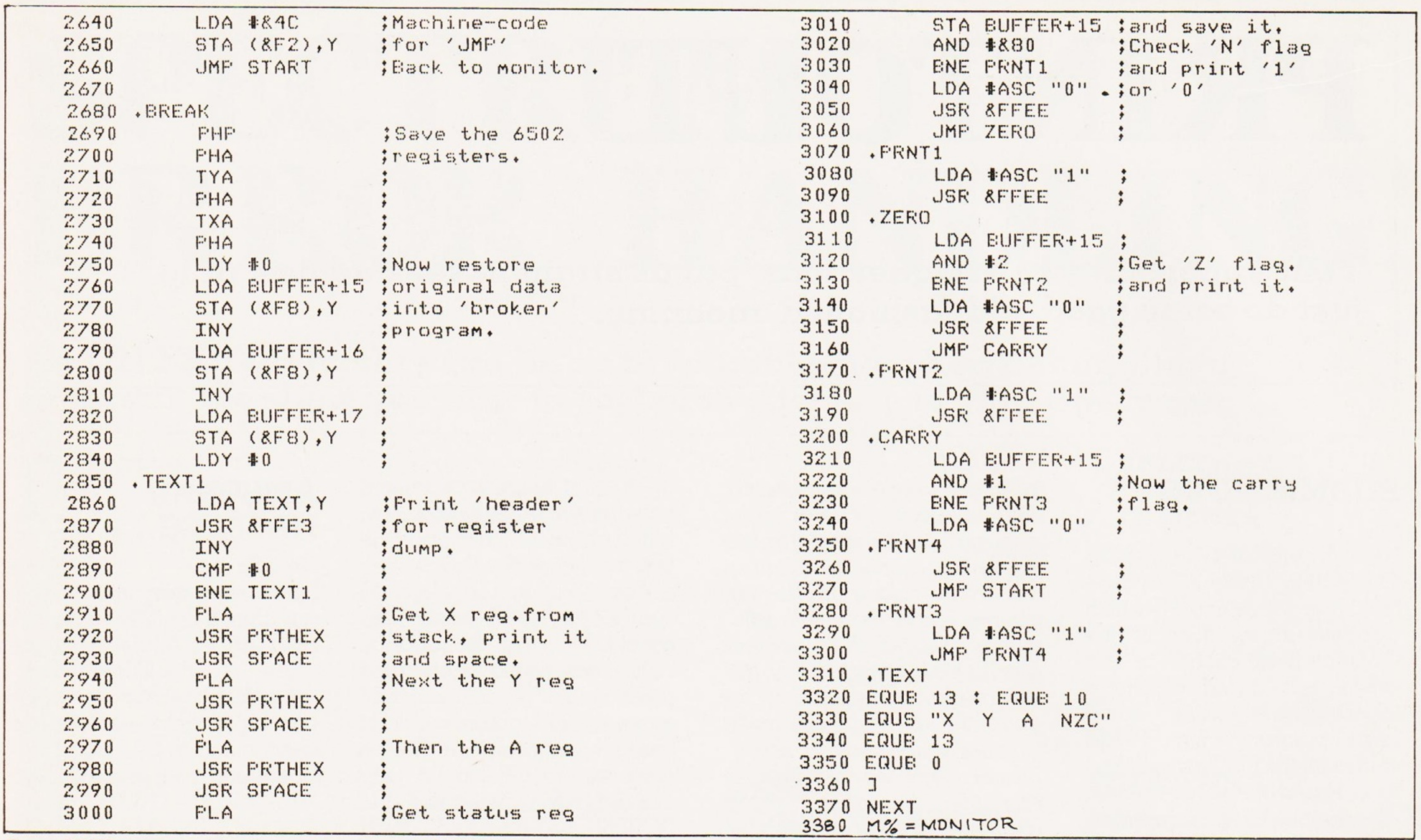
```
 10 FOR I% = 0 TO 3 STEP 3
 20 P% = &4000
 30 [
 40 OPT I%
 50 .start   LDY     #0
 60 .loop    LDA     text,Y
 70          CMP     #ASC "K"
 80          BCC     finish
 90          JSR     &FFE3
100          INY
110          JMP     loop
120 .finish  JMP     M%
130 .text    EQUS    "ABCDEFGHIJK"
140 ]
150 NEXT
```



```
 10 REM   Machine-code Monitor for the BBC micro.
 20 REM   Set P% in line 50 to a suitable value.
 30
 40 FOR I%=0 TO 3 STEP 3
 50 P%=&4000
 60
 70 [
 80 OPT I%
 90 .MONITOR
100         LDA #MONITOR MOD 256
110         STA &0434     ;Addr of M% (lo)
120         LDA #MONITOR DIV 256
130         STA &0435     ;Addr of M% (hi)
140         LDA #229      ;Disable Escape
150         LDX #1        ;key
160         JSR &FFF4     ;
170         LDA #15       ;Turn off page-
180         JSR &FFEE     ;mode.
190
200 .START
210         LDY #0        ;
220 .LOOP
230         LDA PROMPT,Y ;Print command
240         JSR &FFE3     ;prompt.
250         INY           ;
260         CMP #0        ;(end of text).
270         BNE LOOP      ;
280         JSR &FFE0     ;Get keyrd char
290         JSR &FFE3     ;and print it.
300         CMP #ASC "P" ;"Program"
310         BEQ PROG      ;
320         CMP #ASC "G" ;"Run prog"
330         BEQ GO        ;
340         CMP #ASC "H" ;"Hex/Asc Dump"
350         BEQ HEXDUMP   ;
360         CMP #ASC "B" ;"Breakpoint"
370         BEQ REGDUMP   ;
380         CMP #ASC "Q" ;"Quit to Basic"
390         BEQ BASIC     ;
400         JMP START     ;Try Again.
410
420 .PROG          JMP PROGRAM
430 .GO            JMP GO1
440 .HEXDUMP       JMP HEX1
450 .REGDUMP       JMP REG1
460 .BASIC
470         JSR &FFE7     ;Newline
480         LDA #229      ;Enable Escape
490         LDX #0        ;key.
500         LDY #0        ;
510         JSR &FFF4     ;
520         LDA #0        ;Push Basic return
530         PHA           ;address onto
540         LDA #&80      ;stack and return
550         PHA           ;to Basic.
560         RTS           ;
570
580 .PROMPT
590         EQUB 13 : EQUB 10
600         EQUS "P G H B Q ? "
610         EQUB 0
620
630 .BUFFER
640         EQUB (BUFFER+5) MOD 256
650         EQUB (BUFFER+5) DIV 256
660         EQUB 5
670         EQUB &30
680         EQUB &46
690         EQUS STRING$(16,CHR$(32))
700
710 .GTADDR
720         LDA #4        ;Enter max length
730         JSR LDBUFFER ;into Buffer.
740         LDY #0        ;Get first byte
750         JSR LDBYTE    ;from Buffer and
760         STA &F3       ;save it.
770         JSR LDBYTE    ;And second byte
780         STA &F2       ;and save it.
790         RTS           ;
800
810 .LDDATA
820         LDA #2        ;Max length of
830         JSR LDBUFFER ;input in buffer.
840         LDY #0        ;
850 .LDBYTE
860         JSR CVHEX     ;
870         ASL A         ;Shift Hex nibble
```



```
 880        ASL A          ;to m.s.b. and
 890        ASL A          ;store in buffer.
 900        ASL A          ;
 910        AND #&F0       ;
 920        STA BUFFER+10;
 930        INY            ;
 940        JSR CVHEX      ;Convert 2nd
 950        CLC            ;ascii digit.
 960        ADC BUFFER+10;Make complete
 970        INY            ;Hex byte and
 980        RTS            ;leave in 'A'.
 990
1000 .CVHEX
1010        LDA BUFFER+5,Y ;
1020        CMP #&3A       ;> ascii "9"?
1030        BCS LTTER      ;Yes.
1040        AND #&0F       ;Mask off top bits
1050        RTS            ;
1060
1070 .LTTER
1080        AND. #&0F      ;
1090        CLC            ;
1100        ADC #&09       ;
1110        RTS            ;
1120
1130 .LDBUFFER
1140        STA BUFFER+2 ;Max. length.
1150        LDX #(BUFFER MOD 256)
1160        LDY #(BUFFER DIV 256)
1170        LDA #0         ;
1180        JSR &FFF1      ;Get input.
1190        STY BUFFER+11;Save length.
1200        RTS            ;
1210
1220 .PRTHEX
1230        STA BUFFER+12 ;
1240        LSR A          ;Convert to
1250        LSR A          ;ascii form
1260        LSR A          ;for printing
1270        LSR A          ;
1280        JSR PRTASC     ;
1290        LDA BUFFER+12 ;
1300        AND #&0F       ;
1310        JSR PRTASC     ;
1320        RTS            ;
1330
1340 .SPACE
1350        LDA #&20       ;
1360        JSR &FFE3      ;
1370        RTS            ;
1380
1390 .PRTASC
1400        CLC            ;Print in
1410        ADC #48        ;decimal ascii
1420        CMP #58        ;form.
1430        BCC NUMBER     ;
1440        ADC #6         ;
1450 .NUMBER
1460        JSR &FFE3      ;
1470        RTS            ;
1480
1490 .PRTADDR
1500        STA &F3        ;Print address
1510        JSR PRTHEX     ;in Hex.
1520        LDA &F2        ;
1530        JSR PRTHEX     ;
1540        JSR SPACE      ;Print a space
1550        RTS            ;
1560
1570 .PROGRAM
1580        JSR GTADDR     ;Get prog.start
1590        LDA &F3        ;address.
1600 .NXTADDR
1610        JSR PRTADDR    ;And print it.
1620        LDY #0         ;
1630        LDA (&F2),Y    ;Print data at
1640        JSR PRTHEX     ;selected addr.
1650        JSR SPACE      ;Print a space
1660        JSR LDDATA     ;Get new data
1670        LDY BUFFER+11 ;Get length of
1680        CPY #0         ;input. Zero is
1690        BEQ NOCHANGE   ;no new data.
1700        CPY #1         ;One means leave
1710        BEQ DONE       ;program mode.
1720        LDY #0         ;
1730        STA (&F2),Y    ;
1740 .NOCHANGE
1750        CLC            ;Increment addr
1760        LDA &F2        ;at &F2/&F3.
1770        ADC #1         ;
1780        STA &F2        ;
1790        LDA &F3        ;
1800        ADC #0         ;
1810        STA &F3        ;
1820        JMP NXTADDR    ;Print next addr.
1830 .DONE
1840        JMP START      ;Back to Monitor.
1850
1860 .GO1
1870        JSR GTADDR     Get run addr.
1880        LDA &F2        ;and store it
1890        STA BUFFER+13 ;in temporary
1900        LDA &F3        ;buffer. Then
1910        STA BUFFER+14 ;jump to that
1920        JMP (BUFFER+13) ;address.
1930
1940 .HEX1
1950        JSR GTADDR     ;Get address.
1960 .onepage
1970        LDX #16        ;Lines/page.
1980 .oneline
1990        JSR &FFE7      ;start new line.
2000        LDA &F3        ;Print start-of-
2010        JSR PRTADDR    ;line address.
2020        LDY #0         ;Set index to 0
2030 .onebyte
2040        LDA (&F2),Y    ;Get data from
2050        JSR PRTHEX     ;addr. & print it
2060        JSR SPACE      ;and a space.
2070        INY            ;Inc. addr. index.
2080        CPY #8         ;If not yet 8
2090        BNE onebyte    ;do next byte,
2100        JSR SPACE      ;else print space
2110        LDY #0         ;Set index to 0
2120 .oneasc
2130        LDA (&F2),Y    ;Get data again
2140        CMP #32        ;If < ASC "0"
2150        BCC PRTDOT     ;print a dot.
2160        CMP #127       ;If > asc char.
2170        BCS.PRTDOT     ;print a dot.
2180        JMP PRNT       ;Print asc char.
2190 .PRTDOT
2200        LDA #ASC "."   ;Print a dot
2210 .PRNT
2220        JSR &FFEE      ;or asc. char.
2230        INY            ;Increment index.
2240        CPY #8         ;If not yet 8
2250        BNE oneasc     ;print next char.
2260        CLC            ;8 bytes now
2270        LDA &F2        ;printed in Hex
2280        ADC #8         ;and ascii so
2290        STA &F2        ;add 8 to address
2300        LDA &F3        ;stored at &F2/&F3
2310        ADC #0         ;ready for next
2320        STA &F3        ;8 bytes.
2330        DEX            ;Decr. line count
2340        CPX #0         ;If not yet 0
2350        BNE oneline    ;print next line.
2360        JSR &FFE7      ;Else newline
2370        JSR &FFE0      ;Get kybrd input
2380        CMP #32        ;If space-bar
2390        BEQ onepage    ;print next page
2400        JMP START      ;Back to monitor
2410
2420 .REG1
2430        JSR GTADDR     ;Move Breakpoint
2440        LDA &F2        ;address to unused
2450        STA &F8        ;locations in
2460        LDA &F3        ;in zero-page.
2470        STA &F9        ;
2480        LDY #0         ;
2490        LDA (&F2),Y    ;Move data from
2500        STA BUFFER+15 ;brkpnt addr
2510        INY            ;and store in
2520        LDA (&F2),Y    ;temporary
2530        STA BUFFER+16 ;buffer.
2540        INY            ;
2550        LDA (&F2),Y    ;
2560        STA BUFFER+17 ;
2570        LDY #2         ;Change addr....
2580        LDA #(BREAK DIV 256)
2590        STA (&F2),Y    ;to addr of the
2600        DEY            ;Breakpoint.....
2610        LDA #(BREAK MOD 256)
2620        STA (&F2),Y    ;routine in
2630        DEY            ;monitor.
```



```
2640        LDA #&4C        ;Machine-code
2650        STA (&F2),Y     ;for 'JMP'
2660        JMP START       ;Back to monitor.
2670
2680 .BREAK
2690        PHP             ;Save the 6502
2700        PHA             ;registers.
2710        TYA             ;
2720        PHA             ;
2730        TXA             ;
2740        PHA             ;
2750        LDY #0          ;Now restore
2760        LDA BUFFER+15 ;original data
2770        STA (&F8),Y     ;into 'broken'
2780        INY             ;program.
2790        LDA BUFFER+16 ;
2800        STA (&F8),Y     ;
2810        INY             ;
2820        LDA BUFFER+17 ;
2830        STA (&F8),Y     ;
2840        LDY #0          ;
2850 .TEXT1
2860        LDA TEXT,Y      ;Print 'header'
2870        JSR &FFE3       ;for register
2880        INY             ;dump.
2890        CMP #0          ;
2900        BNE TEXT1       ;
2910        PLA             ;Get X reg.from
2920        JSR PRTHEX      ;stack, print it
2930        JSR SPACE       ;and space.
2940        PLA             ;Next the Y reg
2950        JSR PRTHEX      ;
2960        JSR SPACE       ;
2970        PLA             ;Then the A reg
2980        JSR PRTHEX      ;
2990        JSR SPACE       ;
3000        PLA             ;Get status reg
```

```
3010        STA BUFFER+15 ;and save it.
3020        AND #&80        ;Check 'N' flag
3030        BNE PRNT1       ;and print '1'
3040        LDA #ASC "0" .;or '0'
3050        JSR &FFEE       ;
3060        JMP ZERO        ;
3070 .PRNT1
3080        LDA #ASC "1"    ;
3090        JSR &FFEE       ;
3100 .ZERO
3110        LDA BUFFER+15 ;
3120        AND #2          ;Get 'Z' flag.
3130        BNE PRNT2       ;and print it.
3140        LDA #ASC "0"    ;
3150        JSR &FFEE       ;
3160        JMP CARRY       ;
3170 .PRNT2
3180        LDA #ASC "1"    ;
3190        JSR &FFEE       ;
3200 .CARRY
3210        LDA BUFFER+15 ;
3220        AND #1          ;Now the carry
3230        BNE PRNT3       ;flag.
3240        LDA #ASC "0"    ;
3250 .PRNT4
3260        JSR &FFEE       ;
3270        JMP START       ;
3280 .PRNT3
3290        LDA #ASC "1"    ;
3300        JMP PRNT4       ;
3310 .TEXT
3320 EQUB 13 : EQUB 10
3330 EQUS "X   Y   A   NZC"
3340 EQUB 13
3350 EQUB 0
3360 ]
3370 NEXT
3380 M% = MONITOR
```

Type RUN to assemble the program and CALL M% to call up the Monitor. Type G4000 to run the program. The letters A to K should be printed on the screen and the program should then jump back to the Monitor. Because of a 'deliberate' bug in the program, what actually happens is that it jumps straight to the Monitor without printing the letters. The program is obviously jumping to 'finish' too soon. Type B4007 to insert a Breakpoint at address &4007 then G4000 to run the program again. The screen will display:

```
X       Y       A       N Z C
0 0     0 0     4 1     1 0 0
P   G   H   B   Q   ?   _
```

This shows that the Y register contains 0 (we loaded it with 0 in line 50 of the assembler listing) and that the A register contains 41, which is the Hex value of the letter 'A'. The first letter of the text has, therefore, been loaded into the A register but we can also see from the register dump that the 'carry flag' (nzC) is zero, that is *not* set. Therefore the instruction at line 80 — Branch if Carry Clear — is being obeyed immediately so the subroutine at line 90 never prints the letter. Type Q, LIST the BASIC program and change line 80 to:

BCS finish

Type RUN to re-assemble the program, CALL M% to call up the Monitor and G4000 to run the program again. This time the screen will display:

```
ABCDEFGHIJ
PGHBQ?_
```

Better but still not quite right. The final 'K' is missing. This time, insert a breakpoint at &4010 by typing B4010 and run the program again with G4000. Now the display will be:

```
ABCDEFGHIJ
X       Y       A       N Z C
0 0     0 A     4 B     0 1 1
P   G   H   B   Q   ?   _
```

The Y register contains 0A which, if you count along the text, is pointing to the letter 'K' and the A register in fact contains the letter K (&4B). The carry flag now contains '1' so our test at line 80 — Branch if Carry Set — is being obeyed one letter too soon. So, type Q, LIST and change line 70 to:

CMP #ASC "K" +1

Re-assemble the program, type CALL M% and G4000. This time, the program will run correctly.

So, we have seen the Monitor's G B and Q in action; what about P and H? Type P4000 to examine the demonstration program. Step through the listing by simply typing 'return' or change specific locations by entering the appropriate data followed by 'return'. Jump back to the Monitor by typing just *one* character (for example 'A') then 'return'.

Type H followed by a four-digit address to display a Hex and ASCII dump of memory. Sixteen lines are displayed at a time — hit the space bar for the next sixteen or any other key to return to the Monitor.

THE MONITOR PROGRAM

There follows some brief notes about the Monitor's assembler listing, which was written on a BBC-B with the 1·2 Operating System and Level 2 BASIC.

Lines 100 to 130 set up the value of M% to the start address of the Monitor itself. The value of M% is always stored at &434/&435 so this is easy to do.

Lines 140 to 160 disable the 'Escape' key. This facility is not available with Operating Systems before 1·0 so these lines would have to be left out.

Lines 220 to 400 are the 'main loop' which prints the 'P G H B ?' prompt and waits for your reply before jumping to the appropriate routine. Early versions of the assembler do not have the 'EQU' operator so you will have to leave the assembler at line 640 and use the indirection operators query, pling and dollar to set up the 'prompt' and 'buffer' areas.

Lines 710 to 1550 are subroutines to get two and four digit Hex values from the keyboard and to print them either as Hex values or as ASCII characters.

Lines 1570 to 1840 are the routine for the 'P' command.

Lines 1860 to 1920 are for the 'G' command.

Lines 1940 to 2400 are for the 'H' command.

Lines 2420 to 2660 are for the 'B' command.

Lines 2680 to 3350 are for the register dump routine which is entered when a 'breakpoint' is encountered. Once again, early assembler users will have to use the query, pling and dollar operators for the text between lines 3320 and 3350.

PRINTOUT

Your opportunity to ask questions, put us straight, seek advice, or to just do some good, old-fashioned moaning.

SUBMISSIONS

Sir,
I recently submitted an article to Computing Today in response to your invitation which appears at the foot of the Contents page each issue. The article was based around a sophisticated screen dump utility program which I had written for the BBC micro linked to a Nagasaki Z2000 printer. After waiting for some considerable length of time, I received a letter of refusal which I felt was grossly unfair, given that (in my opinion) the article met and probably surpassed your 'standards'. Admittedly, the article, or at least the program, was not written with the magazine in mind. but if readers and prospective contributors had more rigid guidelines to follow, it would be both useful and informative.

Yours sincerely,
Guy Allen,
Plumstead.

The Editor Replies:
Thank you for your letter, Guy, as you have given me an excellent opportunity to suggest some general lines that potential Computing Today *contributors may wish to follow. But first, allow me to apologise to you and other readers who have experienced that most protracted of delays, the 'long wait'. The reasons for this are too numerous (ie too controversial) but readers may rest assured that henceforth, all correspondence will be answered promptly (weather permitting) and without delay.*

Your main point regarding the rejection of material is an important one. In your specific case, the article was rejected NOT because it was substandard, but because it failed to take into account the simple fact that it would appeal to very few readers ie those with BBC micros and Nagasaki printers (the Z2000 to boot!). Briefly, the kind of article that is likely to 'snapped-up' by us are those which discuss general principles at length, backed up with example programs which demonstrate the principles 'at work'. Similarly, the ratio of listings to descriptive copy is of concern to us. Most of our competitors use a program as the starting point for an article, hence the trend is for 3 or 4 pages of listing backed up with one page of descriptive text. Our aim in Computing Today *is for a reversal of this trend, such that the final article would contain 2 or 3 pages (say) of descriptive text, and a page or so of listings, if necessary. In an ideal world, the listings would be dispensed with completely, and principles would ultimately be described in algorithmic form, or in 'pseudo-code' such that the reader could digest and understand the sequence of operations necessary for him to perform the final coding himself. However, readers are invited to write to me to express their feelings on this topic.*

DODGY ON DRAGON

Dear Sir,
I was very happy to read the articles on the Dragon in your October 1984 issue, especially the one submitted by M.A. Seymour ("Negative Thoughts"). I have had very little experience using machine code, and have been baffled and frustrated about how to change the text display on the Dragon.

According to the Motorola spec's, the 6847 video chip should be able to produce FOUR text displays: black on green, black on orange, light green on dark green and light orange on dark orange. On the other hand, the Dragon user manual says, describing the SCREEN command (SCREEN type, colour set):

"TYPE is 0 for the text screen . . . The COLOUR SET is also either 0 or 1. The default . . . is SCREEN 0,0. This sets the text screen with black on green colour set. (It is possible to use SCREEN 0,1, which gives black text on an orange background, but every time the computer prints it will revert to black on green.)"

I would like to comment that, in fact, the only way to hold the orange text screen is by initiating a continuous, or very long, loop, to keep the computer busy. As soon as you press BREAK, the green screen returns, so it's not just PRINT that causes this.

Returning to the very welcome article by M.A. Seymour, I was absolutely thrilled to find that IT WORKS! At last, green on black! However, the article states (first paragraph) ". . . green characters on a dark screen background or orange equivalents using SCREEN 0,1." I humbly ask, can this really be done? If so, I would be extremely grateful to know HOW?!

Sincerely yours,
(Ms) M.S. Greer
Republic of Transkei

We would be pleased to hear from anyone who knows the answer to this.

AMSTRAD HASSLES

Dear Sir,
I am having difficulties in using an Epson printer with the AMSTRAD CPC464: No matter where I put the DIP-switches, I get two line feeds at a time. How can I get only one feed per line?
Name and address supplied.

The Editor Replies:
If you look in your User Instructions, you will see a list of printer port connections. It shows pin 14 earthed. Now, on an MX80 printer earthing this line, either externally or by means of a DIP-switch, produces an automatic line feed. This is believed to be the case with other Epson models, as well.

It is clearly necessary to break this line somewhere in order to make the printer compatible with the CPC464, which always puts out CR and then LF. AMSTRAD, on the other hand, would probably advise you to buy one of their printers. . .

CLUB AFRICA

Dear Sir,
The Dumont and Syndercombe Amateur Computer Club of South Africa was formed in 1983. It is open to people worldwide with any microcomputer. Members receive monthly newsletters, brochures, computer courses and more. Membership in South Africa is R1,20 per annum and International membership is free. Anyone who is interested should contact Jean-Pierre Dumont (International Affairs) at 8 Kipling Road, Farrarmere Benoni 1500, Transvaal, South Africa.

Yours sincerely,
Jean-Pierre Dumont.

SPECTRUM ERROR HANDLING

B. Twissell

An ON ERROR facility can be an invaluable aid when debugging a program. We show you how to add such a facility to your Spectrum.

This article describes a short machine code routine which adds an ON ERROR facility to Spectrum BASIC.

There are many ways in which a BASIC program can terminate:

1) legitimately, by reaching the end of the program or by a STOP statement.
2) By a bug in the program: NEXT without FOR, Out of DATA, etc.
3) By an unforeseen event: Out of memory, Number too big, etc.
4) By an I/O error: STOP in INPUT, Tape loading error, End of file, etc.
5) By the user typing BREAK.

When one of these events occurs, we may want the program to continue running, and possibly take some special action. There are many things that can be done:

a) terminate.
b) ignore the error and carry on regardless.
c) repeat the statement which caused the error.
d) jump to some other statement.
e) perform some statement (or routine) and then return to the interrupt statement.
f) hang up or self destruct.

We might want different actions for different errors; eg (b) is suitable for (5) but (d) is more suitable for (3). The only action provided by Spectrum BASIC is of course (a).

THE COMPETITION

Other microcomputers have ON ERROR and ON BREAK statements for specifying the actions to be taken when errors occur. In principle, Interface 1 owners could write a machine code routine to add such statements to the Spectrum. However, the facility described here is accessed via USR, PEEK and POKE; it can be used whether Interface 1 is present or not. The new facility will handle all the errors which have a report number — thus things like "L BREAK" are handled, but note that "Microdrive not present", etc are not.

Program 1 loads a machine code routine into memory. If the checksum is correct you should save the code using:

```
SAVE --- CODE 30000, 64
```

The dashes should be replaced with the filename of your choice. The routine is actually relocatable, which is a posh way of saying that it can be placed anywhere in memory. Thus you can load it again by:

```
CLEAR 59999
LOAD --- CODE 60000
```

or whatever.

To turn the error processing on, you simply call the routine, using:

```
RANDOMIZE USR 30000
```

30000 is of course replaced by the appropriate address if you've loaded it elsewhere. This would normally be the first statement in the program, for the obvious reason that error processing is turned on before any errors can occur. This implies that you should load the machine code routine before loading the BASIC program.

When error processing is on, the BASIC program is allowed to terminate 'normally', either with "OK" or by a STOP statement. The error processing will be turned off at this point, so if you re-enter the program you'lll have to turn the error processing on as before.

An attempt to terminate with any other error report will be caught, so that the BASIC program continues running.

DESIGN CONSIDERATIONS

The routine has been designed to keep the interface with BASIC simple, and also to keep the routine itself simple. It would have been possible — by some suitable POKE statements — to tell the routine such things as the line number to jump to, but this was rejected for simplicity's sake.

Perhaps the most obvious thing which the routine could do is an automatic CONTINUE whenever an error occurs. This would mean for example that BREAK would be ignored, but if you wanted to take some action you could detect that a BREAK had occurred either by looking at the keyboard or by PEEKing the variable ERRNR. However, in most cases, CONTINUE repeats the statement which caused the error. If the error was, say, Number too big, then the statement would fail again, and the program would hang in a tight loop for ever. And if a VERIFY fails, there's no point redoing the VERIFY unless you also redo the SAVE first.

Rather than have a lot of special cases to try to cope with all eventualities, the solution chosen has no special cases — after an error, execution is always resumed at the next statement. For example, in

```
100 LET n = 4
110 LET n = 1/0
120 PRINT n
```

the division will result in Number too big. The program will not crash, but will resume execution at the PRINT statement. Since statement 110 is not completed, n will still hold its old value, and so 4 will be printed.

WHAT NEXT?

After an error has occurred, the system variable ERRNR (one byte at 23610) will hold a value other than 255. ERRNR corresponds to error reports as follows:

ERRNR	REPORT
255	0 (OK)
0,1,...,8	1,2,....,9
9,10,....,26	A,B,....,r

ERRNR will hold such a value until the next error occurs, at which point it will be overwritten with the new error number.

If you want to detect when an error occurs, you should first clear any previous error condition by POKEing 255 in the ERRNR. Beware that executing the last statement does not set ERRNR to 255, because it is assumed to still contain the 255 which was put there when the program began execution. The upshot of this is that if some error has occurred and you haven't cleared ERRNR, the program may not be allowed to finish. You are advised to use a STOP statement whenever you want the program to finish, since this sets ERRNR to 8 and will always work.

Thus all error types (2) to (5) mentioned earlier are caught, and any of the actions (a) to (f) can be taken by appropriate code in BASIC.

Program 2 gives some examples.

Program 1

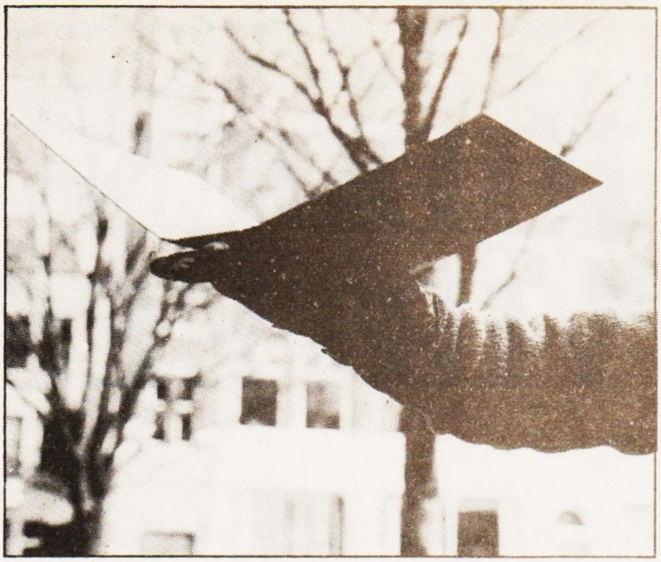
```
 10 REM
    ..................................................
    Implements an ON ERROR
    facility in Spectrum BASIC.
    See text.
    ..................................................

 20 CLEAR 29999
 30 address = 30000
 40 LET h$= "210f000922b05ceb2a
    3d5c732372c93a3a5c3c2802fe09ca03
    1321445ccb7e280b3a475c3c772a455c
    22425c2100007c32715c220b5c2ab05c
    e52a425cc39e1b"
 50 LET sum=0
 60 FOR i=1 TO LEN h$ STEP 2
 70    LET byte=16*FN d(h$(i))+
                   FN d(h$(i+1))
 80    POKE address+(i-1)/2,byte
 90    LET sum=sum+byte
100 NEXT i
110 IF sum<>5087 THEN
       PRINT FLASH 1 ' ' '
       "Line 40 is wrong!!!!"
120 DEF FN d(c$)=CODE c$-
    (CODE "0" AND c$<="9") —
    (CODE "a"-10 AND c$>="a")
```

Program 2

```
 10 REM
    ..................................................

    Example of ON ERROR
    facility. This assumes
    that the necessary machine
    code routine is in memory
    at address 30000.
    ..................................................

 20 RANDOMIZE USR 30000:
    LET errnr=23610:
    LET ok=255:
    LET number too big=5:
    LET stop in input=16:
    LET invalid file name=14:
    LET tape loading error=26
 30 PRINT "Setting up array."'
    "You can't BREAK this"'
    "(try it and see!!)"
 40 DIM n(300)
 50 FOR i=1 TO 300
 60 POKE errnr,ok
 70 LET n(i)=1/INT (10*RND)
 80 IF PEEK errnr=
    number too big THEN
    PRINT "overflow"
 90 NEXT i
100 REM LOOP
110 POKE errnr,ok
120 INPUT "Enter file name."'
    "STOP (shift-6) is"'
    "ineffective. . ."'
    LINE f$
130 IF PEEK errnr=
    stop in input THEN
    GO TO 110
140 REM END LOOP
150 REM LOOP
160 SAVE f$ DATA n()
170 IF PEEK errnr=
    invalid file name THEN
    STOP
180 PRINT "Now to verify."'
    "If it fails, the SAVE"'
    "will be repeated."'
    "BREAK will get you out."
190 POKE errnr, ok
200 VERIFY f$ DATA n()
210 IF PEEK errnr=
    tape loading error THEN
    GO TO 160
220 REM END LOOP
230 PRINT "Finished"
240 STOP
```

ON ERROR machine code

```
; ............................
; Adds ON ERROR facility to
; Spectrum BASIC. See text.
; ..............................
;
; First of all there is a short
; piece of code to turn on the
; error processing. This is a
; matter of working out the
; address of the <on> label -
; when this is invoked by USR in
; BASIC, BC will have the value
; of <turnon>, and <on> is 15
; bytes away - then the address
; of <on> is stored in two
; places: an unused location at
; 23728; and in to the location
; pointed to by ERRSP...
;
turnon          LD          HL,15
                ADD         HL,BC
                LD          (23728) ,HL
                EX          DE ,HL
                LD          HL, (ERRSP)
                LD          (HL) ,E
                INC         HL
                LD          (HL) ,D
                RET

;
; The BASIC interpreter may
; detect any error at all sorts
; of places. However, the action
; taken is always the same
; (except, that is, for errors
; from Interface 1). The
; interpreter clears all the junk
; from the Z80's stack. Since
; the length of the stack depends
; on where the error occurred,
; ERRSP always points to the 2nd
; value left on the top is an
; address. Normally it is 1303H;
; the code at this address prints
; an error message and prepares
; to receive a command. The
; error processing is turned on
; (above) by changinging this
; address on the stack. Thus
; when an error occurs, the stack
; is cleared and our code is
; invoked.
; For error numbers 0 (ok) and 9
; (STOP statement) we allow the
; program to terminate by jumping
; to the code which was supposed
; to be executed...
;
on          LD      A, (ERRNR)
            INC     A
            JR      Z, L1
            CP      9
L1          JP      Z, 1303H
;
; if a jump is about to take
; place, bit 7 of NSPPC will be
; 0 - we do nothing. Otherwise,
; we set up NEWPPC and NSPPC with
; the line number and statement
; which is to be executed next
; (this is one more than the
; current statement)...
;
            LD      HL, NSPPC
            BIT     7, (HL)
            JR      Z, L2
            LD      A, (SUBPPC)
            INC     A
            LD      (HL) ,A
            LD      HL, (PPC)
            LD      (NEWPPC),HL
;
; now we clear a couple of things
; which would ordinarily be
; cleared...
;
L2          LD      HL ,0
            LD      A ,H
            LD      (FLAGX) ,A
            LD      (DEFADD) ,HL
;
; finally, the address of our
; error will have been popped
; from the stack, so we put it
; back (remember, we stored it in
; an unused location)...
;
            LD      HL , (23728)
            PUSH    HL
;
; now we jump back in to the
; interpreter, as if CONTINUE had
; just been entered, but after
; the check for BREAK...
;
            LD      HL , (NEWPPC)
            JP      1B9EH
```



HIDDEN VIRTUES

Bill Horne

The Amstrad CPC464 has had nothing but favourable reviews. However, we've found out just a little bit more about the subtleties of this machine since we reviewed it in October '84.

When a new microcomputer appears on the market, everyone wants to get their sticky fingers on the keyboard and try it out. Is it just another run-of-the-mill device, or has it got some special virtue to commend it? Unfortunately, the answer may not be obvious at once. The apparent value of a new machine depends heavily on the nature of the documentation. If a given facility is not mentioned in the user guide, it may be difficult to guess that the facility exists.

A case in point is the AMSTRAD CPC464. In the October issue of Computing Today there was a thoroughgoing review of the machine as it then appeared. The conclusions were favourable, and no doubt encouraged a number of people to take a closer look.

Then further documentation surfaced, and that revealed the machine in an entirely new light. Yes, it was a nice machine for the BASIC user, but it was now seen to be much more than that. Some of its potential is yet to be realised in full, but there is quite enough to be going on with.

SIDEWAYS ROMs

Just for starters, there is the provision for attaching 'sideways ROMs' via the extension bus. The AMSTRAD CPC464 uses an ingenious system of memory expansion. All available memory addresses are used for 64K of RAM, but the top and bottom quarters of the address range are shared with 16K blocks of ROM, giving an apparent memory size of 96K. The top 16K of the RAM is devoted to screen memory, and an area below this holds code copied from ROM into RAM so that it is always available for immediate use. In the 0000-0040 address range ROM and RAM hold the same code, though some bytes in this area may be patched by the user to put special code in RAM.

With such an arrangement, it is essential to have means for switching banks and jumping to a given location simultaneously, or effectively so. The CPC464 provides this by creating what are really extensions to the Z80 instruction set. For example, a code sequence CF XX XX is given a special meaning analogous to the C3 XX XX format. The latter performs a simple jump to the location defined by XX XX, but the CF code — used in the CPC464 — does rather more.

Suppose the full call is CF 1B 88. This looks like an access to address 881B, but the two top bits of the address are stripped away, and access is to 081B. This is in the area occupied by the lower ROM and the bottom quarter of RAM, so the system needs to know which is required. Those two top bits give the answer. If bit 14 is a 1, the lower ROM is disabled, else enabled, and bit 15 similarly controls the upper ROM. The desired object is achieved: the jump address and the relevant ROM state are implemented simultaneously, to all intents and purposes.

There is no need, by the way, to worry about the ROMs when writing to memory, because written data always goes to RAM. It is therefore possible to write to screen memory at any time. To maintain a constant flow of data to the screen, the necessary accesses to screen memory are interlaced with ROM accesses.

So this sets up a nice system which achieves an apparent 50% increase in memory size with minimum hassle. But the system goes further. The lower ROM holds the operating system, while the upper ROM is the 'language' ROM, usually a BASIC interpreter. In theory, up to 251 alternative ROMs, each of 16K, can be switched in to replace the normal upper ROM. These sideways ROMs are not contained within the main unit, but are accessed through the extension bus. They can be entered by another 'pseudo op-code', known as FAR CALL. It takes the form DF XX XX YY. XX XX defines the entry address, YY can select a particular ROM or change the ROM state to enable or disable the upper and lower ROMs.

It would be impossible to go into all the aspects of this system here, but enough has been said to show that the CPC464 is something of a wolf in sheep's clothing.

PATCHABLE JUMPS

The extra documentation also revealed a massive list of more than 200 operating system calls, each one accessed by a link set in RAM. It is therefore possible to patch these calls to introduce user code. This has many uses. For example, it is sometimes desirable to vet the character codes passed to a printer, eliminating any that might produce undesirable effects. This can arise, for example, when a data stream contains VDU control codes that would mean something different to the printer in use. All that is needed is a change of the entry jump so that it accesses user code, that code performing the necessary vetting process, and then calling the original routine. Small wonder that those in the know have been calling the CPC464 'Hacker's Delight'!

Once again, it would not be feasible to go into full detail here, but the temptation to cite some examples is too strong to be resisted;

A call to B900 will enable the upper ROM, returning the previous ROM state in the A register. A call to B906 will perform a similar service for the lower ROM.

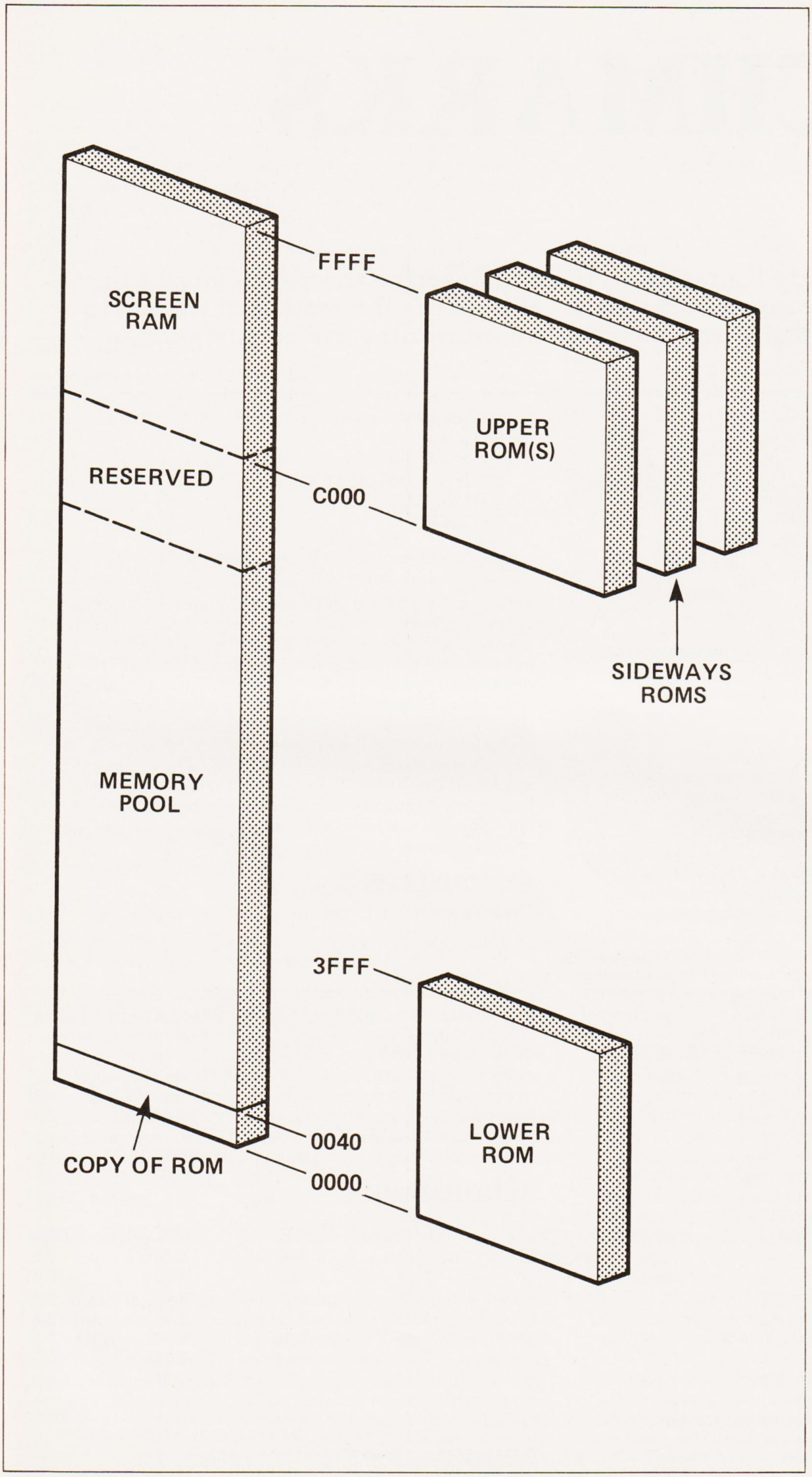
A call to B903 will similarly disable the upper ROM, while a call to B909 will disable the lower ROM. Again, the previous ROM state is returned in A.

A call to B90C, with A set to the value produced by the calls described above, will restore the previous ROM state.

A call to BC7A will reset the Sound Manager and the sound chip. That can be very handy on occasion! If you find the BASIC sound control system tedious, and you know how to control the AY-3-8910 sound chip, you need a call to BD34 with A holding the number of the register you want to set and C holding the data you want to send.

DEVPAC

To make full use of these facilities, you really need an assembler, a disassembler and a monitor. Lo and behold, AMSOFT can supply your needs, with a Hisoft DEVPAC adapted to the CPC464. We have not been able to try this out in full, but the associated manual makes us impatient to do so.



Amstrad CPC464 memory map

It will be evident by now that the CPC464 is really a machine for the machine code enthusiast, while remaining quite docile enough for a beginner. This highlights a difficult problem facing manufacturers. Should they aim their design at one market or the other? This is perhaps more a question of documentation and advertisement than of the actual hardware. Many apparently quite simple machines have a lot of hidden virtues, which are sometimes put to good use by professional programmers, while the actual users remain blissfully unaware of the complications they are calling up. Those who sell the machines may well have even less understanding of the hidden parts, and the machines must first be sold to people like that, before the general public have a chance to make a decision.

Had the CPC464 been offered as a 'Hacker's Special', it might never have reached the public at all. Yet it is being snapped up mainly by experienced users, rather than first-timers. Printer cables have been more in demand than printers, suggesting that many buyers have printers already.

The 'Concise Firmware Specification', an AMSOFT publication, will also be in great demand once its existence becomes known. Of daunting size, it lists all the operating system facilities and explains the working of various parts of the system, using a professional format and a presentation that would by no means disgrace a minicomputer manufacturer.

There is also the BASIC Reference Manual, which describes and specifies the BASIC system in more formal terms — and more completely — than does the User Instructions book.

Even more documentation is on the way, but that is still in the pipeline, and not for the public gaze as yet.

As its full capabilities emerge, the CPC464 looks more and more like the prototype of the micro of the future. The hardware is incredibly economical, and the main virtues stem from the firmware. That is more likely to produce good results than complicated hardware systems backed by slapdash software.

BENCHMARKS

Don Thomasson

Speed is one feature that is often used to describe the performance of a particular microcomputer. But how reliable are the tests that we use to measure this quantity, and how can we guarantee the consistency of such tests?

In 1977, the Kilobaud magazine introduced a set of benchmark tests intended to suit microcomputers. Of late, the results given by different machines in execution of these tests has sometimes been a little confusing, and there have been suggestions that a revised set of tests might be considered. It would be interesting to learn what our readers think about this, but perhaps it would do no harm to take a closer look at the existing benchmark tests to see whether we understand what they tell us.

BENCHMARK 1

The first test is essentially a FOR loop of 1000 iterations and no internal content;

```
100 PRINT "S"
200 FOR K = 1 to 1000
300 NEXT K
400 PRINT "E"
500 END
```

Execution times for the interval between the display of 'S' and the display of 'E' range from five seconds to less than a second. The execution times for a single iteration of the loop thus amount to similar numbers of milliseconds, it being assumed that the surrounding actions contribute a negligible proportion of the total time.

Looked at dispassionately, these figures are a trifle surprising. What is involved in executing the loop? Line 300 establishes the initial, current and final values of the loop index, and stores them away, with the address of the end of the line (or the beginning of the next line). Line 200 adds the step value, in this case a default of unity, to the index variable, compares the result with the end value, and if that end value has not been reached action returns to the point defined by the stored address — the start of line 300.

We are thus concerned solely with the actions of line 300, which entail an addition, a comparison, and a jump. And that can take five milliseconds, perhaps five thousand machine cycles? Evidently it does. After all, even simple operations on floating point numbers are tedious and complex. But who said anything about floating point numbers? Couldn't integers be used? And some systems run faster if the variable name is not appended to NEXT.

It begins to look as if there are a number of factors which the simple definition of the test fails to take into account. In 1977, it was reasonable to assume that floating point numbers would be involved, and that the full form of the NEXT statement would be used. That is no longer valid.

However, we have noted that the test establishes execution time for an addition, a comparison and a jump, plus perhaps some time used in scanning back through the stored loop data.

BENCHMARK 2

The second test uses a different form of loop;

```
100 PRINT "S"
200 K = 0
300 K = K + 1
400 IF K<1000 THEN 300
500 PRINT "E"
600 END
```

This time, the loop involves two lines. Line 300 performs an increment, and line 400 checks the result against a constant value, jumping back if that value has not yet been reached. There is, as before, an addition, a comparison and a jump, but the execution times are much longer, between 3 and 9 milliseconds per iteration. Now why should that be? The difference is that the location of variable K has to be determined three times, whereas with the FOR loop it can be found more directly by reference to stacked values. And in some systems the numeric value 1000 has to be converted to binary, whereas in other systems the conversion is performed when the program line is entered. There is not enough evidence to show how much these factors contribute individually to the increase in execution time, but it appears that identifying and fetching a variable may take around a millisecond, certainly no more.

BENCHMARK 3

The third test inserts a new line into the previous listing;

```
310 A=K/K*K+K-K
```

This adds between 5 and 13 milliseconds per iteration, the time taken to perform an addition, a subtraction, a multiplication and a division. Six references to variables are needed, which accounts for a good deal of that time, and we can dimly begin to see how the extra time is split up. Remembering that multiplication and division in floating point are inherently faster than addition and subtraction, we can start a tentative allocation of individual times. By leaving out one term or another in line 310, we could obtain some fairly precise figures.

BENCHMARK 4

Next, a different line 310 is used;

```
310 A=K/2*3+4-5
```

Two references to variables are needed, the other factors come as constants from the line itself. The execution times do not differ greatly from those of the third test, but the difference is not consistent. Some machines are slower, others faster. Those which are faster usually convert numerics to floating point at the time of entry. They therefore have no need to convert from decimal at run time.

BENCHMARK 5

The fifth test uses a similar structure, but introduces a dummy subroutine call, the added lines being;

```
320 GOSUB 700
700 RETURN
```

The previous line 310 is retained.

This is where the differences really begin to show. The added time for each iteration ranges from 400 microseconds to 4 milliseconds. Yet the actions needed are superficially the same; the end of the GOSUB statement must be marked and stored, the address of the target line must be found, and the scanning pointer must be set to point to that line. Then the RETURN command involves resetting the scanning pointer to the end of the GOSUB statement. So wide a range of times suggests that the detail of these actions is implemented in very different ways in different machines.

BENCHMARK 6

So far, the tests have proceeded rationally, provided it is clear that the significant figures are the differences between one test and the next, but we now stray into confusion. Test six adds the following lines;

```
250 DIM M(5)
330 FOR L=1 TO 5
340 NEXT L
```

The dimensioning statement is irrelevant, since it lies outside the main loop, and is not used until the next test. The added FOR loop increases the time by between 4.6 and 43 seconds. This is more than might be expected from benchtest 1, but it must be remembered that in this case the loop is being set up a thousand times. If we care to work it out, we can separate the times for setting up and executing the loop. For a **Spectrum,** the iteration time is about 5 mS, so executing the loop in this benchmark should take 25 mS. The increase in time is about 43 mS, so setting up the loop takes about 18 mS. The **AMSTRAD CPC464** takes about 1.14 mS to execute an iteration of the loop, or 5.7 mS for five iterations. The added time is 8.9 mS, so setting up takes 3.2 mS. The information is there if you look for it.

BENCHMARK 7 & 8

Test 7 inserts one further line;

```
335 M(L) = A
```

This adds between 7.6 and 18.5 seconds overall, or 1.5 to 3.7 milliseconds to execute line 335 once.

Benchmark 8 goes off on a different track;

```
100 PRINT "S"
200 K=0
300 K=K+1
330 A = K 2
340 B = LOG(K)
350 C = SIN(K)
400 IF K 1000 THEN 300
500 PRINT "E"
600 END
```

This clearly needs to be compated with benchmark 2 to make sense, since lines 330 - 350 are added to the earlier benchmark. The added times range from about 1.6 seconds to 30.9 seconds, which could be interpreted in two different ways; either the slower machines are making a meal of the job, or the faster machines are skimping it. The three added commands all involve the use of a mathematical series. The more terms there are in the series, the more accurate will be the result, assuming that the terms are correctly proportioned. A very short series may be accurate over a given range, but will deviate in extreme cases. That may satisfy those who only care about speed, but not those who want accurate results. To be meaningful, this test should provide an indication of accuracy as well as speed.

It is not unusual to complete a set of bench test results by giving an overall average figure. I've done it myself, but I realise now that it makes a nonsense. The absolute test figures are not what is important. It is necessary to look at differences between one test and another.

All this could be taken to mean that the value of the existing tests is at least dubious. That is not so, providing they are properly understood and interpreted. However, it does seem that improvement should be possible, if only to separate out the different individual functions more clearly.

There are those who have expressed the view that such tests are all nonsense, anyway. If you are concerned about the speed of BASIC, you shouldn't be using it. There are a number of other factors which are more significant in assessing the value of a machine, such as available store space and screen control flexibility. These would be difficult to codify into anything approaching a figure of merit, and there would probably be endless arguments about the relevance of each and every factor.

What do you think? How do you judge one machine against another? Let us have your views, and I will try to analyse them and combine the ideas which seem most productive. If you think bench tests are a load of nonsense, say so. That's the kind of thing we need to know.

BENCHMARKS ANALYSED

	BM1	BM2	BM3	BM3−BM2	BM4	BM4−BM2	BM5	BM5−BM4	BM6	BM6−BM5	BM7	BM7−BM6	BM8	BM8−BM2
ZX Spectrum	4.9	9.0	21.9	12.9	20.7	11.7	25.2	4.5	68.2	43.0	86.7	18.5	25.1	16.1
Dragon 32	1.2	9.1	17.7	8.6	19.2	10.1	22.2	3.0	31.1	8.9	44.7	13.6	10.8	1.7
Commodore 64	1.2	9.3	17.6	8.3	19.5	10.2	21.0	1.5	29.5	8.5	47.5	18.0	11.3	2.0
Osborne I	1.5	4.6	12.1	7.5	11.9	7.3	12.9	1.0	36.1	12.2	49.6	13.5	6.2	1.6
Sirius I	1.8	5.3	10.6	5.3	10.9	5.6	12.6	1.7	23.4	10.8	35.9	12.5	11.3	6.0
Amstrad CPC464	1.1	3.3	9.2	5.9	9.6	6.3	10.2	0.6	19.2	9.0	30.3	11.1	34.2	30.9
BBC Micro	0.8	3.1	8.3	5.2	8.7	5.6	9.1	0.4	13.7	4.6	21.3	7.6	5.3	2.2

The benchmark times quoted were taken from the table published in Microchoice, less the Newbrain and with the addition of the AMSTRAD CPC464. Note that BM8 for the CPC464 does not agree with the figure published in a recent CT: This has been re-checked, as it was clearly too small to be true, no larger than the time for BM2. This alone shows the value of taking differences between benchmarks. All Computing Today benchmarks are performed using 1000 iterations.

CT INDEX '84

Due to circumstances beyond our control (the Editor forgot it) here is the slightly late index for 1984, lovingly word-processed into alphabetical order and subject.

AN ARGUS SPECIALIST PUBLICATION
computing today
DECEMBER 1984
90p
GRAPHICS GALORE!
Bits and blobs: drawing with Acorn's Bitstik
Taking the tablets: Koalapad reviewed
Art and the Amstrad: simple drawing program
IO's Pluto for out of this world graphics
PLUS:
GRiD lines — Compass computer
Computing today
APRIL 1984
MEMOTECH'S MTX500 ON THE TEST BENCH:
A black pearl from the peripheral people
Learn how to pull machine code apart with our BBC OS12 ROM disassembly feature
Have a Brain Storming session on your CP/M system
More of the same — Recursion on four Apple languages

GENERAL FEATURES	MONTH	PAGE
A Fast Sort	Jul	72
All Sorts of Sort	Sep	36
Atari ANTICs Part 1	Oct	39
Atari ANTICs Part 2	Nov	66
Business Matters — supplement	Nov	35
Computers and the Commons	May	72
Debugging	May	25
Disassembly Techniques	Apr	68
Electronic Mail	May	45
Hard and Fast (Winchester discs)	Jun	32
Lights, Computer, Action	Dec	42
Mind and Movement	Jul	32
Non-random Random Numbers	Feb	44
Out-of-this-World Pluto	Dec	37
PCW Show	Dec	51
Random Thoughts	Jul	79
Real-time Clock (project)	Sep	28
Recursion	Apr	72
Some Notes on the Apple Part 1	Jan	29
Some Notes on the Apple Part 2	Feb	51
The Art of Islam	Jul	52
The Biomechanics of Computing	Jul	24
The Leech (Spectrum data transfer project)	Jun	46
The Legible Spectrum (Centronics project)	Mar	59
Using Epson MX-80 Graphics	Feb	15
Viewpoint: Computer Manuals	Jan	72

REGULAR FEATURES

CLUB CALL	Jul	76
	Mar	64
	Sep	66
BOOK PAGE		
Smalltalk, Fifth generation	Jan	14
Beyond BASIC	Feb	29
Introductory books and IT	Mar	25
Children, graphics, current scene	Apr	38
Books on the Commodore 64	May	80
Robots, electronics, FORTH	Jun	56
Word processing, AI	Jul	64
Programming, cybernetics, modems	Aug	52
'How to' books, Logo	Sep	14
IBM PC, computer studies	Oct	30
Industry matters, MSX, measurement	Dec	34
PROCopinion		
New machines	Mar	56
Amstrad, QL, copyright	Sep	50

SERIALISED ARTICLES

Easycode	Mar	17
	Apr	26
	May	16
	Jun	24
Extending the 64's BASIC	May	54
	Jun	37
	Jul	41
	Aug	67
	Sep	47
Getting More from The 64 (Part 2)	Jan	18
	Feb	24
Learning FORTH (Part 3)	Jan	66
	Feb	37
	Mar	48
	Apr	31
	Jul	67
Problem Page (Part 6)	Jan	52

HARDWARE REVIEWS	MONTH	PAGE
Advance 86B	Oct	34
Alphatronic PC	Jun	14
Amstrad CPC464	Oct	17
Ape-face Atari Centronics Interface	May	84
Apple IIc	Nov	16
Apple Macintosh	Nov	60
Apricot xi	Aug	15
Atmos	May	59
Atmos Discs and Plotter	Jul	36
BBC 6502 Second Processor/Bitstik	Dec	15
Brother HR-5 thermal printer	Jul	29
CUBE's 6809 Second Processor	Sep	19
Commodore SX-64	Sep	32
Dragon 64	May	37
Electron	Jan	37
Epson PX-8	Nov	29
GRiD Compass	Dec	28
Hitachi MBE 16002	Jun	60
KoalaPad Graphics Tablet	Dec	56
Memotech MTX500	Apr	19
MicroSight Video Digitiser	Jun	18
Psion Organiser	Dec	46
Sinclair QL	Nov	51
Spectravideo SV318	May	33
Tandy Model 4P	Jul	18

Computing today
MAY 1984
Three of a kind
We look at low-cost computing with reviews of the Atmos, Spectravideo and Dragon 64 systems
Customize your Commodore 64 with our new series on adding BASIC keywords
Programs don't work? All the secrets of de-bugging techniques are revealed inside
Format your text in BBC Electron programs with our word-wrapping procedure
PLUS: The winners of our Adventure competitions

Computing today
BUSINESS MATTERS
PORTABLE POWER The Epson PX-8 and Apple IIc on test!
AT LAST THE QL an undongled review!
MTX Toolbox
A Macreview of the amazing Macintosh
DREAM HOLIDAY
GREAT COMPETITION

SOFTWARE REVIEWS		
Adventures	Jun	68
	Aug	20
	Sep	54
	Oct	44
Beebugsoft's BBC Micro Toolkit	Jun	28
Brainstorm Ideas Processor	Apr	64
Commodore BASIC Extensions	Jul	14
Data Genie	Jul	82
Dragon Sprite Magic	Oct	25
Electron graphics packages	Sep	16
Lostock Apple Screen Editor	Feb	48
Machine Code Tutor	Sep	52
MicroWriter BBC keyboard	Jun	42
SCOPE Graphics Language	Mar	38
Spectrum Machine Code (books and assemblers)	Feb	20
ZX81 FORTH	Feb	56

SOFTWARE		MONTH	PAGE
Art and the Amstrad Part 1	Amstrad BASIC	Dec	41
Atari Disc Library	Atari BASIC	Jan	55
BBC Disassembler	BBC BASIC	Sep	41
BBC Draughts	BBC BASIC	Oct	46
BBC Draw	BBC BASIC	Dec	38
BBC Poker Part 1	BBC BASIC	Mar	31
BBC Poker Part 2	BBC BASIC	Apr	44
CBM64 Program Recovery	BASIC/Assembler	Feb	64
Colour Genie Monitor	BASIC loader	Feb	33
Commodore Search and Replace	6502 Assembler	Jun	64
Dragon Hints on Tints	Dragon BASIC	Jul	75
Dragon inverse printing	6809 Assembler	Oct	51
Faster FORTH	TRS-80 assembler	Jun	57
Finding and Keeping	BBC BASIC	Aug	60
Genie Commands	Z80 Assembler	Apr	51
Genie Sprites	Z80 Assembler	Aug	28
Genie Utilities	Z80 Assembler	Mar	43
Interrupts	NAS-SYS	Jun	51
MBASIC Data Entry	BASIC	Sep	60
MTX Toolbox	Memotech BASIC	Nov	73
Make A Date (Calendar)	BASIC	Jan	76
Memory Moves	TRS-80 BASIC/ assembler	Nov	21
Microdrive File Line Editor	Spectrum BASIC	Aug	36
Mode 7 Screen Editor	BBC BASIC	Aug	23
NAS-SYS Printing	Z80 Assembler	Oct	55
Named Files for NAS-SYS	Z80 Assembler	Jan	44
Programmer's Aid	TRS-80 BASIC	Jan	24
Proportional printing	BBC Assembler	Aug	33
Screen Scroller	Z80 (BASIC loader)	Aug	41
TRS-80 Arraysave	Z80 Assembler	Sep	56
TRS-80 Screen Editor	BASIC/Assembler	Jan	62
Two PET Utilities	PET BASIC	Feb	59
Upgrading Your WP	Z80 Assembler	May	63
Word-wrapping	6502 Assembler	May	93
Z80 Text Compactor	Z80 Assembler	May	87

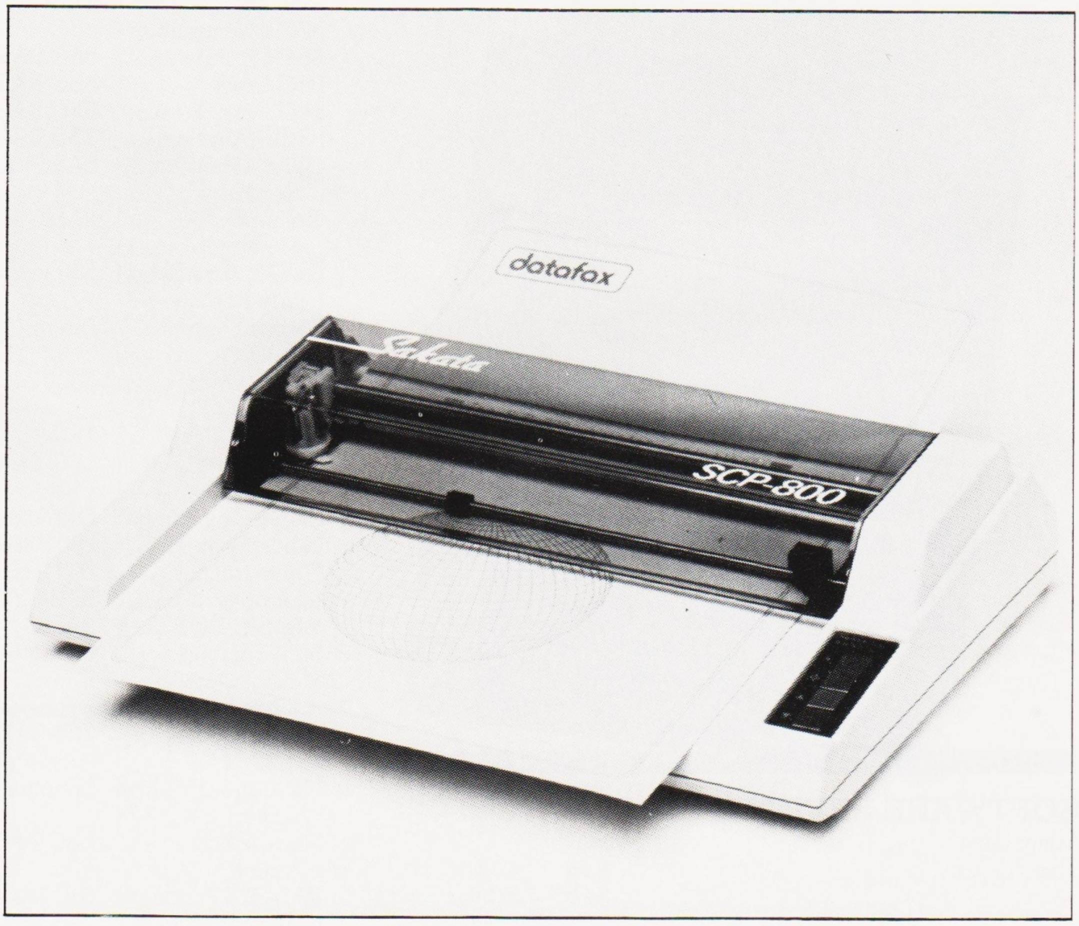
PRINT OR PLOT

Don Thomasson

Printer-plotters are usually out of reach for the small computer user, by reason of price or interface requirements. Here is an exception, low in cost and directly interchangeable with most printers where the interface is concerned.

It was just as well that we had stocked up with paper the day before the Sakata SCP-800 came into our lives. It's an engaging little beast, and we were soon knee-deep in discarded sheets produced by enthusiastic experimentation. It would print — by drawing the character shapes in a very neat fashion — and it would run several familiar screen graphics routines, given some simple adaptations.

The most surprising thing about the device on initial examination was that neither the hardware nor the supporting documentation bore a name or type number. That came from a covering letter. Recognition should be simple, however, as the shape and size are distinctive, even if there should be a change of number or name. As the photographs show, the unit is small and neat, the mechanism delicate, almost fragile in appearance, though probably no more so than similar devices which hide their working parts more completely.



GETTING TO WORK

The driving interface is identical with the 'Centronics' form, eight bits being required for full capability, though limitation to seven bits only appeared to rule out italic print characters. An RS232C serial interface is also provided as an alternative, working at 1200 baud clock rate. For convenience, we used the parallel version, which offered us a choice of computers. Our initial tests were based on a BBC Model B, but similar results were obtained with an AMSTRAD CPC464 and a Spectrum fitted with a Kempston interface.

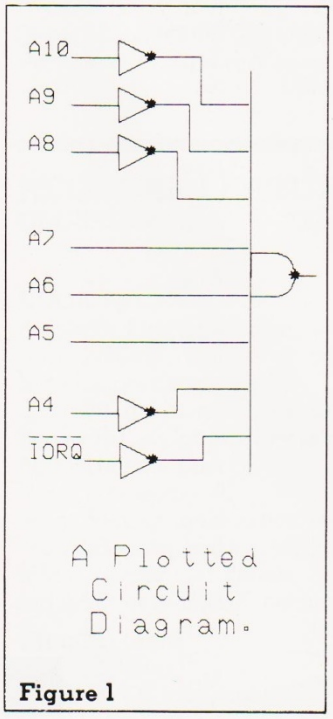


Figure 1

A scan through the manual revealed that output data was essentially in text form. Where a graphics display on the screen required the format 'MOVE X,Y'', the Sakata required 'PRINT ''M'';X;'','';Y'. The example programs used 'STR$(X)' in place of 'X', but that, perhaps, was made necessary by the (unidentified) machine for which they were written.

Table 1 shows the available command letters and their equivalents — where appropriate — in screen graphics commands.

Now, demonstrations programs are all very well, but producing an actual drawing is a more complex business,

FACTSHEET	Sakata SCP-800 Printer Plotter
Dimensions	292mm wide, 190mm deep, 71mm high.
Printing Speed	12cps (text mode)
Interface	Centronics or RS232C
Cost	£179 + VAT (£205)
Supplier	Datafax Ltd., Datafax House, Bounty Road, Basingstoke

because the data has to be provided, rather than being generated by calculation. Nevertheless, it was found possible to generate a respectable circuit diagram, using procedures or subroutines to handle the symbols (Fig. 1).

A SLIGHT SNAG

At this point, a slight snag emerged. Some printouts were perfect, but others showed quite distinct blank patches. These were more noticeable with the black pen, but use of the automatic check facility, which draws a line with each of the four pens, showed no hint that the black pen was drying out.

Up to that point, we had used single sheets of A4 paper, of the 70 gm/m^2 weight. When we tried the much thinner continuous roll paper that came with the machine, the bald patches were more numerous. However, we suspected that paper thickness was not the sole reason, as the roll paper still gave trouble, even when backed by an A4 sheet. It seems possible that the patches were due to grease from sticky fingers, and this idea gained support as we found that care in paper handling gave better results.

With both kinds of paper the process of feeding in a new piece called for care. Movement is controlled by tiny rollers at the edges of the paper, and it was only too easy to feed the paper in on the skew, but this problem diminished as we gained experience.

Another slightly disconcerting characteristic was that after a new sheet of paper or section of the roll had been moved into position it was necessary to re-initialise. Otherwise, the system remembered where it was on the last sheet and moved to that position.

One final niggle; some of the commands had separate meanings for the display system, and on machines where they were sent to both the plotter and the screen it was necessary to take special precautions. The command CHR$(18) was sent on the BBC Computer by the form VDU1,18, for example. This ensured that it went only to the printer, not to the screen control.

BROADER USAGE

It is evident that the control system is primarily intended to be used with BASIC, which is adapted to the output of decimal data to the printer, using ASCII codes. Machine code users will find this less convenient. They may be able to tap the decimal output routine in the BASIC interpreter, failing which they will have to write a routine of their own. This is not too difficult, but it is an unfortunate necessity.

Unless integers are used for the output data, there is a lot of redundant output. Coordinates are only recognised in integer form, and the output of several decimal places is a waste of effort. On the other hand, if a lot of successive relative moves are made, there could be a cumulative error.

PERFORMANCE

On the whole, the performance was adequate, considering the low cost (£205 inc. VAT). In text mode the rate of printing was no more than 12 characters per second, but the characters were clear and neat. Purists might say that the device was 'cheap and cheerful', but that would be unkind. It is aimed at a particular requirement, for flexible drawing and printing at low initial cost, and it is designed to suit users of BASIC rather than machine-code fans. These objectives are met with reasonable success.

The documentation was fully adequate, with only a few oddities due to translation, and the slightly specialised nature of the example listings. It was perhaps a pity that there were a couple of character shape oddities, a rather fat exclamation mark or 'pling', and a cross stroke on the diagonal of the 'Z', but neither is too serious.

One point which must be mentioned is that we examined only one version of the machine, the one with built-in power supply and RS232C interface. The data sheet lists RS232C as an optional adaptor, and lists an external 9V 1.2A power supply. Since these are associated with the SCP-800 type number, a certain amount of care may be needed in specifying the machine you want.

CONCLUSION

Were it not for the occasional bald patches and the rather odd outrigger used to support the paper roll, we would have been quite enthusiastic about this device. As it stands, it is an ideal way of getting involved in plotting for minimum cost, and could be used to produce business data in diagrammatic form.

On the whole, we must give it our support.

TABLE 1

Text Mode:	Selected by CHR$(17)
&08:	Non-deleting backspace. Allows underline, etc.
&0B;	Reverse line feed. For overline, superscript, etc.
&1D;	Change to next colour pen.
Graphic Mode:	Selected by CHR$(18)
"A"	Go to LH margin: Enter Text Mode
"Cn"	Select pen n.
"D";x;",";y	Draw absolute to x,y (DRAW X,Y)
"H"	Home to origin. (MOVE 0,0)
"I"	Set origin at present position.
"J";x;",";y	Draw relative to present position plus x,y (DRAWR X,Y)
"Ln"	Set type of line — continuous, dots, dashes.
"M";x;",";y	Move absolute to x,y (MOVE X,Y)
"P..."	Print subsequent characters in string.
"Qn"	Set print direction. (Up, down, left, right)
"R";x;",";y	Move relative (MOVER X,Y)
"Sn"	Set character size for printing. (n = 0 to 63)
"X";a;",";b;",";c	Draw a coordinate axis marked off in steps.

For commands D and J, further X/Y points can be appended.

MICRO VU

Geoff Parselle

Geoff Parselle describes a simple method to monitor the output from a standard cassette deck to ensure error-free loading.

One of the major problems associated with loading programs from cassette tape into many computers is the output volume level required to guarantee a good load.

Most cassette recorders use an automatic recording level when saving programs, so the problem of faulty saving does not usually occur. The main problem occurs when loading programs back into the computer from the cassette: program loss, blank screens, drop outs etc. are but a few of the results.

Commercial games and similar tapes often vary considerably in their output volume levels (ranging from very low level, unable to load without the volume control on the deck at maximum with any associated hum problems), to being so high that the volume control sometimes needs to be turned to such a low level that with cheap volume pots, holding a set level is difficult.

The need to hunt around to find the correct level can be most frustrating, particularly when it is found that the small paint mark made last week on the volume control is not correct after all!

The fairly cheap method I have devised to standardise volume loading levels is to use a VU meter on the output from tape to computer. After a little experimentation, a satisfactory level can be arrived at, such that it almost guarantees correct loading each time once set at the beginning of the load sequence during the auto levelling tone in front of the program. This only takes a second or so.

In my case I have used this system satisfactorily with both the BBC 'B' and the Spectrum computers.

For the BBC computer, the unit is as shown in photo 1 with associated wiring diagram at Fig. 1. It consists of two 3.5mm jack plugs for ear and mic connections on the cassette recorder, a 2.5mm jack plug for remote control and a DIN plug for the computer connection, together with the meter mounted in its plastic box.

The VU meter has been glued into a hole cut in the plastic box and screened lead used to make connections (soldered). As shown in photo 2 and Fig. 1, a diode has been added into the circuit to enable the meter to work off alternating current (AC), it being a direct current(DC) meter. Don't forget, diodes can only be fitted one way round, the larger coloured band encircling the diode being towards the meter positive (+) connection.

For the BBC Computer an 8 or 10 ohm resistor has been placed across the meter connections to create a load enabling the meter to work: see photo 2 and Fig. 1. The BBC Computer does not load the cassette deck with the equivalent of a speaker resistance, and because of this, the meter will not work. (This may happen with other computers as well). Because of this, fit the resistor across the meter pins to create the loading. You will not need this resistor for the Spectrum Computer which creates its own loading to the cassette deck. Of course, in all cases, the meter is only fitted into the ear lead, the mic lead being left untouched.

Slight problems exist where the Spectrum (and ZX 81) computer is concerned due to the need to remove the mic or ear connections when loading or saving. For this, a switch can be fitted into the same box as the meter. See wiring diagram of Fig. 2. Being double poled, the switch can be set for either load or save and, of course, when on load the meter also works.

There is no reason why this meter, or meter/switch system cannot be used with other computers unless they have dedicated cassette decks, the only alterations being the types of plugs used, the need for the load/save switch, and whether the meter needs loading by soldering in a resistor. Trial and error will decide whether the resistor is needed. Once built, try the unit without the resistor. Should no reading occur, then add the resistor — simple as that.

Since building this unit I can virtually guarantee safe loading of programs first time, and

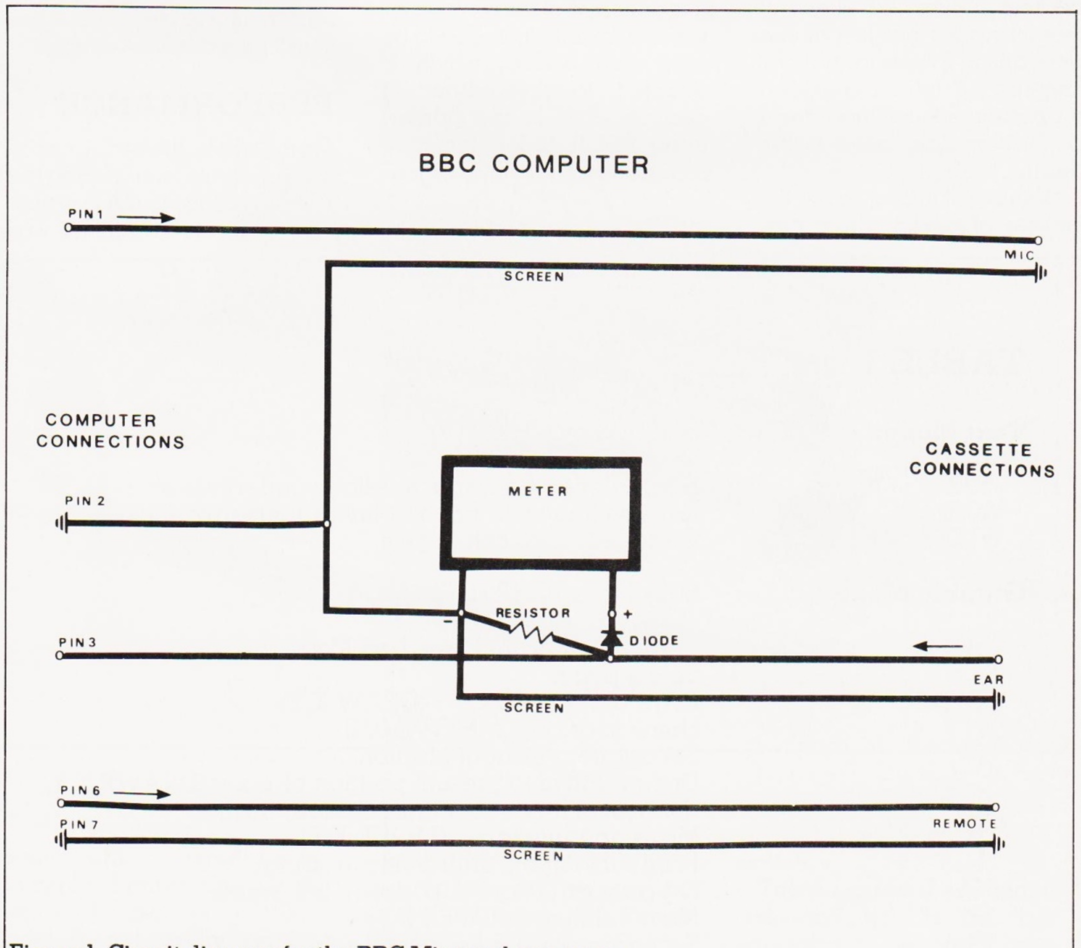


Figure 1. Circuit diagram for the BBC Micro volume meter.

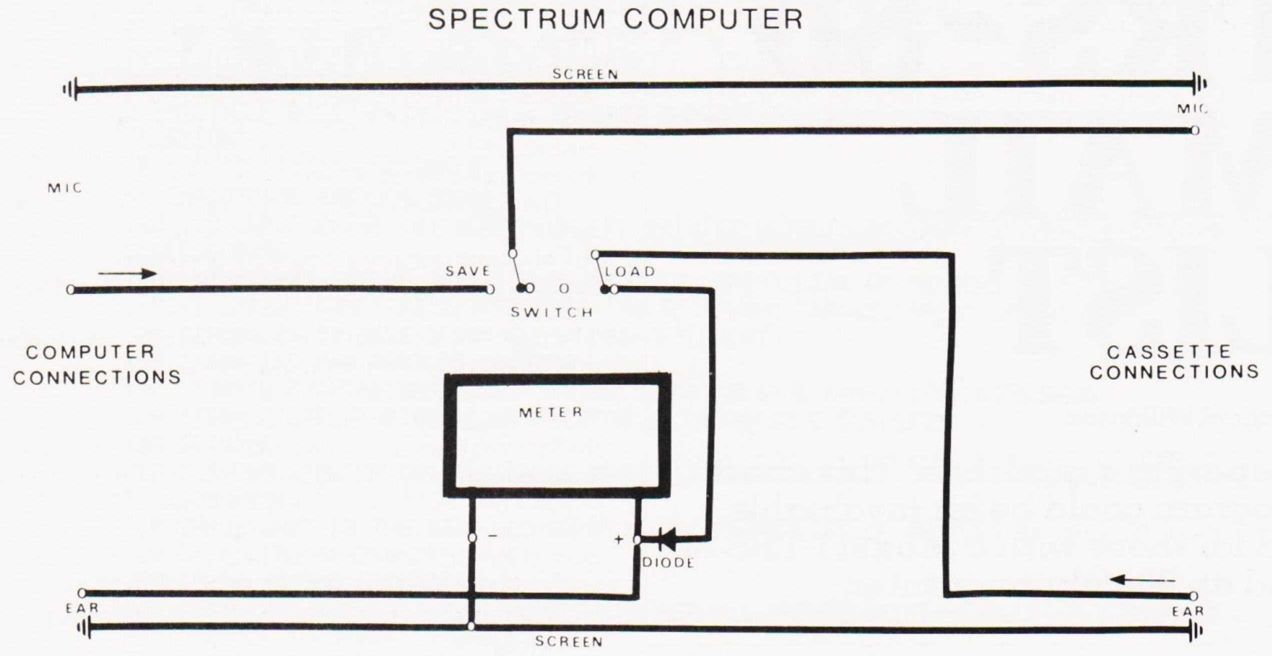


Figure 2. Circuit diagram for the Spectrum volume meter.

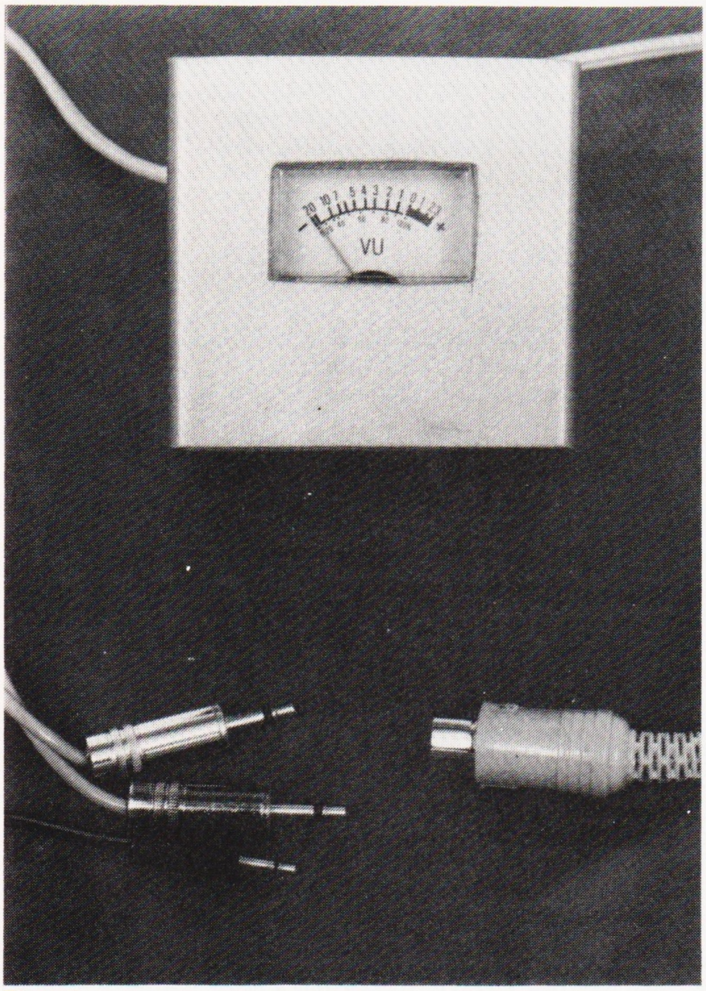


Photo 1.

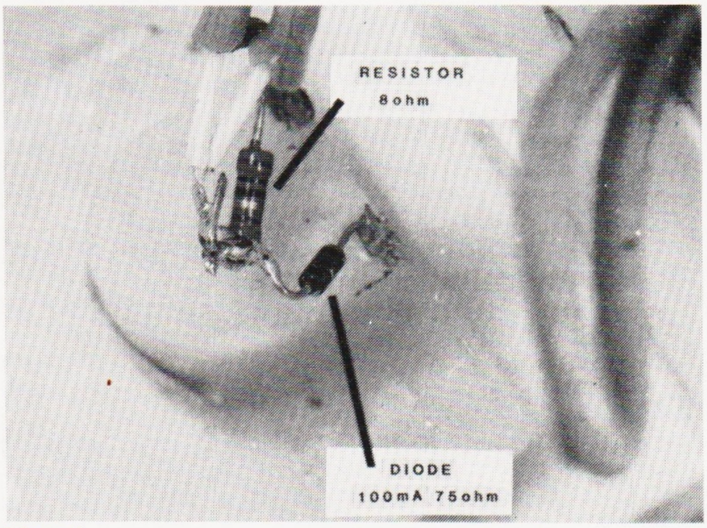


Photo 2.

now, note down the loading level required for each program on the cards used as my program index, eg −1 or 0 or +1 etc, this level being set immediately the program starts loading.

PARTS LIST

All items used are available from Maplin Electronics, PO Box 3, Rayleigh, Essex (Maplin Catalogues available from W.H. Smith).

Box Snap together Box B3. Ref. YK50E
Panel Meter VU Meter V41. Ref. RW73Q
Diode 100V 75mA. Ref. 1N4148
Resistor 8.2 ohm or 10 ohm. Ref. S8.2 ohm or 10 ohm
Switch Standard Toggle Switch SPDT. Ref. FH11M (double pole)
Leads a. Twin individually screened. Ref. XR21X Cable Twin, sold by the metre.
b. Single-core braided screen. Ref. XR13P single cable grey (for remote control connection), sold by the metre.
Plugs a. Screened plug 3.5mm Ref. HF81C Plug SCR 3.5
b. Screened plug 2.5mm (for remote control) Ref. HF77J Plug SCR 2.5
c. DIN Plug 7 pin (for BBC) Ref. HH30H DIN plug 7 pin
d. DIN Plug 5 pin (some cassette decks). Ref. HH27E DIN plug 5 pin A
c. DIN Plug 3 pin (some cassette decks). Ref. HH25C DIN plug

TRS-80 MAIL LIST

Michael Wilkinson

Preparing a mailshot? This short program could be an invaluable aid for those with a Model I TRS-80 and an 80-column printer.

This is a very useful yet very simple program for the Model I using an ordinary 80 column printer. You can use cassette, Exatron stringy floppy or disk, and it shows that the "old fashioned" Model I (or Video Genie) is still a useful workhorse. This program could be used by a small business or, as I do in conjunction with two other simple finance programs that I have written, by the treasurer of a club or society.

The program has been kept as simple as possible, so that as much memory as possible is reserved for storing names. The program itself takes only 1630 bytes, leaving room for 200 to 250 names and addresses on a 16K computer. It can be easily changed to suit different requirements, and should be easy to convert to other types of computers. The names and addresses are stored as DATA statements in the program itself, which means that you do not have to muck around with data cassettes and their slow loading.

EXECUTION

The program does not sort into alphabetic order, but by entering the names in order and numbering the data lines in increments of, say, 20 then there is room to add many more names in their correct position between existing lines.

When you run the program, you are presented with a menu giving four choices:

(1) **Listing** the names and addresses alphabetically (or in whatever order they were entered), either on the screen or printer.
(2) **Listing** the names and addresses by areas (or in some other grouping that you program), again on the screen or printer.
(3) **Printed** out on printer only, as address labels, three across the page.
(4) **A method** of saving two copies of the program on cassette.

Enter the program as listed, and get the feel of it by running it a few times until you are confident with using it. Then you can add your own name list, and adapt the program to suit your needs. I will describe some of the features and how easy it is to change it to your requirements.

When you run the program using menu selections (1) or (2), the subroutine at lines 600, 610 prevents the list of names from scrolling off the screen. When three asterisks *** appear at the lower right of the screen, the program halts until any key is hit. I prefer to use

X=PEEK(14463) : IFX=0... rather than **X=INKEY$: IFX=" "...**

as it is faster, and does not scroll past if you accidentally hit a key twice or suffer from key bounce.

If you select printer output without your printer connected, the subroutine at line 400 prints a message then returns you back to the menu. This not just a piece of fancy programming, but is important as the computer will lock up otherwise, waiting to send output to the printer. The only way out of such a situation is to push the RESET button — and if you have an expansion interface the program is instantly lost!

The subroutine at line 700 checks the number of lines printed by the printer. If it is over 61, a "top of page" command is given. If you are using single sheet feeding, or for some other reason do not want this feature, delete lines 700 and 710, and the GOSUB700 in lines 150 and 260.

Lines 50000 and 50010 is a useful feature which will allow you to save two copies of the program on cassette, with a gap of about two seconds between them.

OK, you have entered the program (and no doubt cursed at the fact that there are no spaces between BASIC words — but it does save memory!), and you feel that you have got the hang of things. Enter, DELETE 10000-10230. Type AUTO 10000,20. Now enter your name and address list as DATA statements. You must have a comma between each "field", which means four commas — if you have more or less, the program will not run! If one field is empty — such as no phone number at an address, you must have a blank space. For example:

```
10040 DATA CRABB MR B,21 SEASIDE RD,NAPIER, ,5
```

MINOR ADJUSTMENTS

Now before you run the program with your name list typed in, you must adjust it to suit the number of areas you have — or whatever other grouping you have. I have a friend in a harrier club who uses the program to print out lists of club members by their grades (1 = senior, 2 = junior man, 3 = colt, 4 = boy, 5 = woman, 6 = junior woman etc). Let us assume that you have 10 groups. Change lines 220 and 230 to

```
220 L=L+1:IFL>10... etc
230 IFL>10L=0... etc
```

Now try the program out with your name list. You might like to change the TAB positions to suit your longer addresses. That is easy — just alter the TAB numbers in lines 110, 150, and 260 to change printer layout, and the TAB numbers in lines 140 and 250 to change the screen layout. Remember though, that you can only TAB as far as TAB(63) using a printer. If you want to go higher use

```
STRING$(xx-PEEK(16539),32)B
```

(where xx = TAB number you want and B = what to print), instead of TAB(xx).

Perhaps you want a fancy title at the top of your list. Just add before the first LPRINT in lines 110 and 210

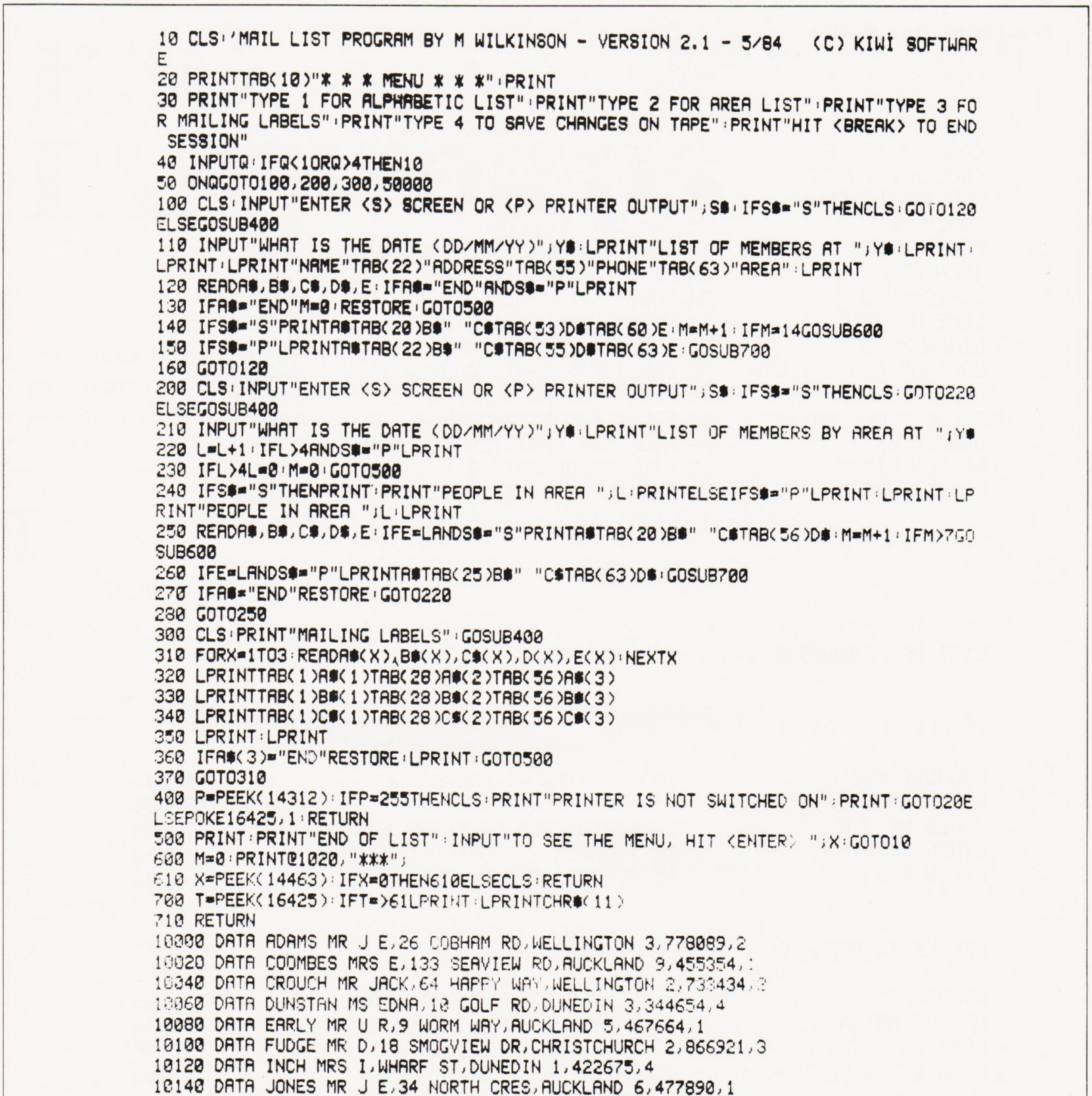
```
LPRINTCHR$(14),"T R S - 80 CLUB
(WELLINGTON)",CHR$(15)
```

or whatever your club name is.

Maybe you are using two-across self-adhesive labels. Make the following changes

```
310 FORX=1to2:... etc
320 LPRINTTAB(1)A$(1)TAB(41)A$(2)
330 LPRINTTAB(1)B$(1)TAB(41)B$(2)
340 LPRINTTAB(1)C$(1)TAB(41)C$(2)
```

If you want to use one-across labels delete from line 310 the FORX=1TO3. Delete from 360 the (3) after A$. Delete from lines 320, 330 and 340 everything after the A$, B$ and C$. If you are using a different size adhesive label, you may have to add an extra LPRINT to line 350, to make each subsequent label line up properly. I am a bit of a miser, and do not use self-adhesive labels

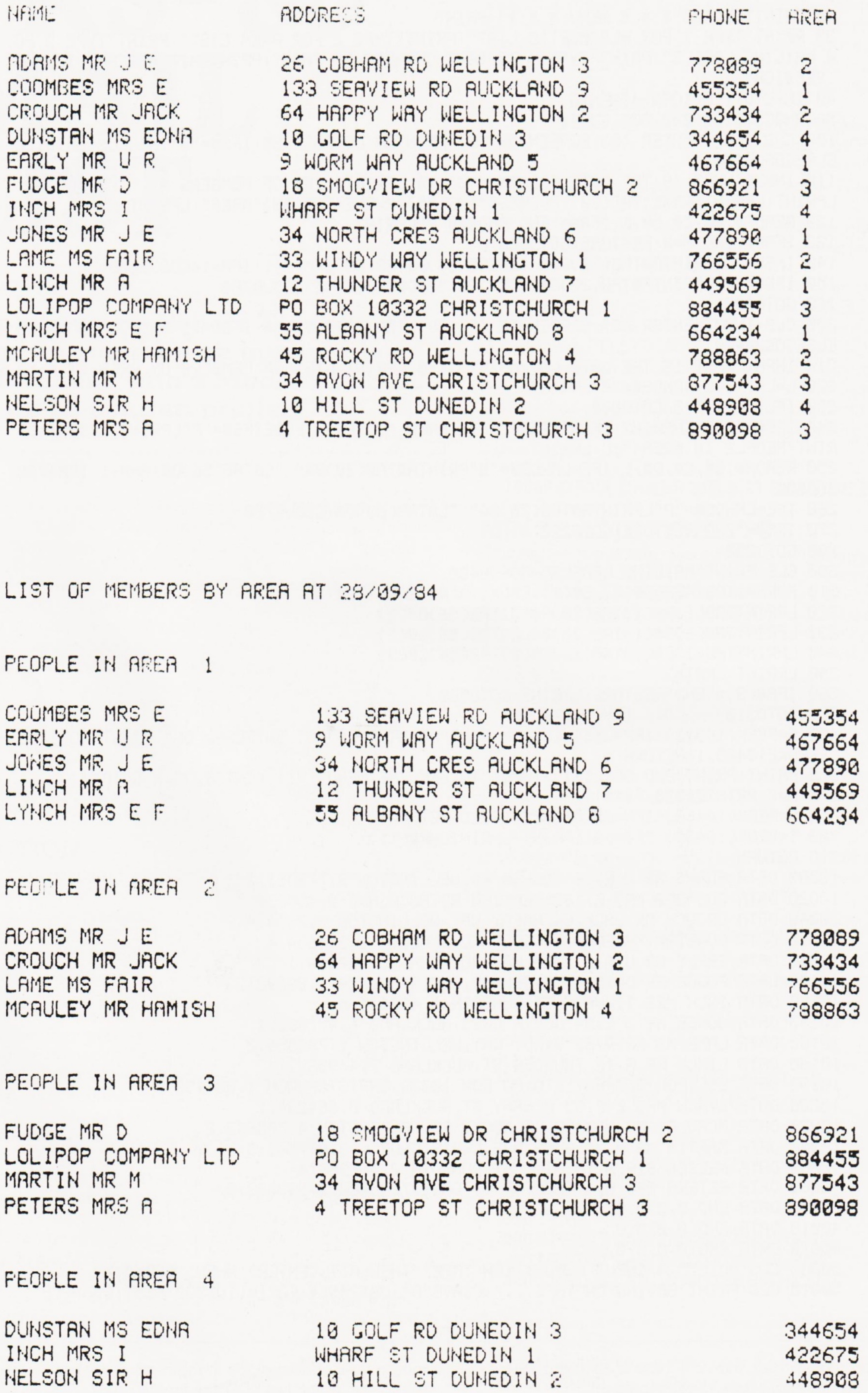


```
10 CLS:'MAIL LIST PROGRAM BY M WILKINSON - VERSION 2.1 - 5/84   (C) KIWI SOFTWAR
E
20 PRINTTAB(10)"* * * MENU * * *":PRINT
30 PRINT"TYPE 1 FOR ALPHABETIC LIST":PRINT"TYPE 2 FOR AREA LIST":PRINT"TYPE 3 FO
R MAILING LABELS":PRINT"TYPE 4 TO SAVE CHANGES ON TAPE":PRINT"HIT <BREAK> TO END
 SESSION"
40 INPUTQ:IFQ<1ORQ>4THEN10
50 ONQGOTO100,200,300,50000
100 CLS:INPUT"ENTER <S> SCREEN OR <P> PRINTER OUTPUT";S$:IFS$="S"THENCLS:GOTO120
ELSEGOSUB400
110 INPUT"WHAT IS THE DATE (DD/MM/YY)";Y$:LPRINT"LIST OF MEMBERS AT ";Y$:LPRINT:
LPRINT:LPRINT"NAME"TAB(22)"ADDRESS"TAB(55)"PHONE"TAB(63)"AREA":LPRINT
120 READA$,B$,C$,D$,E:IFA$="END"ANDS$="P"LPRINT
130 IFA$="END"M=0:RESTORE:GOTO500
140 IFS$="S"PRINTA$TAB(20)B$" "C$TAB(53)D$TAB(60)E:M=M+1:IFM=14GOSUB600
150 IFS$="P"LPRINTA$TAB(22)B$" "C$TAB(55)D$TAB(63)E:GOSUB700
160 GOTO120
200 CLS:INPUT"ENTER <S> SCREEN OR <P> PRINTER OUTPUT";S$:IFS$="S"THENCLS:GOTO220
ELSEGOSUB400
210 INPUT"WHAT IS THE DATE (DD/MM/YY)";Y$:LPRINT"LIST OF MEMBERS BY AREA AT ";Y$
220 L=L+1:IFL>4ANDS$="P"LPRINT
230 IFL>4L=0:M=0:GOTO500
240 IFS$="S"THENPRINT:PRINT"PEOPLE IN AREA ";L:PRINTELSEIFS$="P"LPRINT:LPRINT:LP
RINT"PEOPLE IN AREA ";L:LPRINT
250 READA$,B$,C$,D$,E:IFE=LANDS$="S"PRINTA$TAB(20)B$" "C$TAB(56)D$:M=M+1:IFM>7GO
SUB600
260 IFE=LANDS$="P"LPRINTA$TAB(25)B$" "C$TAB(63)D$:GOSUB700
270 IFA$="END"RESTORE:GOTO220
280 GOTO250
300 CLS:PRINT"MAILING LABELS":GOSUB400
310 FORX=1TO3:READA$(X),B$(X),C$(X),D(X),E(X):NEXTX
320 LPRINTTAB(1)A$(1)TAB(28)A$(2)TAB(56)A$(3)
330 LPRINTTAB(1)B$(1)TAB(28)B$(2)TAB(56)B$(3)
340 LPRINTTAB(1)C$(1)TAB(28)C$(2)TAB(56)C$(3)
350 LPRINT:LPRINT
360 IFA$(3)="END"RESTORE:LPRINT:GOTO500
370 GOTO310
400 P=PEEK(14312):IFP=255THENCLS:PRINT"PRINTER IS NOT SWITCHED ON":PRINT:GOTO20E
LSEPOKE16425,1:RETURN
500 PRINT:PRINT"END OF LIST":INPUT"TO SEE THE MENU, HIT <ENTER> ";X:GOTO10
600 M=0:PRINT@1020,"***";
610 X=PEEK(14463):IFX=0THEN610ELSECLS:RETURN
700 T=PEEK(16425):IFT=>61LPRINT:LPRINTCHR$(11)
710 RETURN
10000 DATA ADAMS MR J E,26 COBHAM RD,WELLINGTON 3,778089,2
10020 DATA COOMBES MRS E,133 SEAVIEW RD,AUCKLAND 9,455354,1
10040 DATA CROUCH MR JACK,64 HAPPY WAY,WELLINGTON 2,733434,2
10060 DATA DUNSTAN MS EDNA,10 GOLF RD,DUNEDIN 3,344654,4
10080 DATA EARLY MR U R,9 WORM WAY,AUCKLAND 5,467664,1
10100 DATA FUDGE MR D,18 SMOGVIEW DR,CHRISTCHURCH 2,866921,3
10120 DATA INCH MRS I,WHARF ST,DUNEDIN 1,422675,4
10140 DATA JONES MR J E,34 NORTH CRES,AUCKLAND 6,477890,1
10160 DATA LAME MS FAIR,33 WINDY WAY,WELLINGTON 1,766556,2
10180 DATA LINCH MR A,12 THUNDER ST,AUCKLAND 7,449569,1
10190 DATA LOLIPOP COMPANY LTD,PO BOX 10332,CHRISTCHURCH 1,884455,3
10200 DATA LYNCH MRS E F,55 ALBANY ST,AUCKLAND 8,664234,1
10220 DATA MCAULEY MR HAMISH,45 ROCKY RD,WELLINGTON 4,788863,2
10240 DATA MARTIN MR M,34 AVON AVE,CHRISTCHURCH 3,877543,3
10260 DATA NELSON SIR H,10 HILL ST,DUNEDIN 2,448908,4
10280 DATA PETERS MRS A,4 TREETOP ST,CHRISTCHURCH 3,890098,3
40000 DATA END,0,0,0,0
40010 DATA END,0,0,0,0
40020 DATA END,0,0,0,0
50000 CLS:OUT255,4:INPUT"REPOSITION TAPE, THEN HIT <ENTER> WHEN READY";X
50010 CLS:PRINT"SAVING ON TAPE...":CSAVE"A":OUT255,4:FORI=1TO2000:NEXT:CSAVE"A"
```

— I print them on plain paper, then cut them to size with scissors and use wide clear adhesive tape to stick them on.

Adding extra names is easy if you keep a printout of the DATA lines. Just LLIST 10000 — so that you can add the new DATA lines in their correct place to maintain alphabetic order. To change an entry, just EDIT the line to be altered. Who said that cassette-based TRS-80 computers were not good for anything useful?

Example output from Mail List



```
NAME                   ADDRESS                            PHONE    AREA

ADAMS MR J E           26 COBHAM RD WELLINGTON 3          778089    2
COOMBES MRS E          133 SEAVIEW RD AUCKLAND 9          455354    1
CROUCH MR JACK         64 HAPPY WAY WELLINGTON 2          733434    2
DUNSTAN MS EDNA        10 GOLF RD DUNEDIN 3               344654    4
EARLY MR U R           9 WORM WAY AUCKLAND 5              467664    1
FUDGE MR D             18 SMOGVIEW DR CHRISTCHURCH 2      866921    3
INCH MRS I             WHARF ST DUNEDIN 1                 422675    4
JONES MR J E           34 NORTH CRES AUCKLAND 6           477890    1
LAME MS FAIR           33 WINDY WAY WELLINGTON 1          766556    2
LINCH MR A             12 THUNDER ST AUCKLAND 7           449569    1
LOLIPOP COMPANY LTD    PO BOX 10332 CHRISTCHURCH 1        884455    3
LYNCH MRS E F          55 ALBANY ST AUCKLAND 8            664234    1
MCAULEY MR HAMISH      45 ROCKY RD WELLINGTON 4           788863    2
MARTIN MR M            34 AVON AVE CHRISTCHURCH 3         877543    3
NELSON SIR H           10 HILL ST DUNEDIN 2               448908    4
PETERS MRS A           4 TREETOP ST CHRISTCHURCH 3        890098    3




LIST OF MEMBERS BY AREA AT 28/09/84


PEOPLE IN AREA  1

COOMBES MRS E             133 SEAVIEW RD AUCKLAND 9              455354
EARLY MR U R              9 WORM WAY AUCKLAND 5                  467664
JONES MR J E              34 NORTH CRES AUCKLAND 6               477890
LINCH MR A                12 THUNDER ST AUCKLAND 7               449569
LYNCH MRS E F             55 ALBANY ST AUCKLAND 8                664234

PEOPLE IN AREA  2

ADAMS MR J E              26 COBHAM RD WELLINGTON 3              778089
CROUCH MR JACK            64 HAPPY WAY WELLINGTON 2              733434
LAME MS FAIR              33 WINDY WAY WELLINGTON 1              766556
MCAULEY MR HAMISH         45 ROCKY RD WELLINGTON 4               788863

PEOPLE IN AREA  3

FUDGE MR D                18 SMOGVIEW DR CHRISTCHURCH 2          866921
LOLIPOP COMPANY LTD       PO BOX 10332 CHRISTCHURCH 1            884455
MARTIN MR M               34 AVON AVE CHRISTCHURCH 3             877543
PETERS MRS A              4 TREETOP ST CHRISTCHURCH 3            890098

PEOPLE IN AREA  4

DUNSTAN MS EDNA           10 GOLF RD DUNEDIN 3                   344654
INCH MRS I                WHARF ST DUNEDIN 1                     422675
NELSON SIR H              10 HILL ST DUNEDIN 2                   448908
```

BBC PASSWORDS

Passwords are often the key to accessing a particular program, but not remembering or concealing a password can often be the key to disaster!

The method of only permitting authorised users to gain access to particular programs, data bases or files on a computer system using passwords is a popular one, often incorporated into software packages, particularly those written for more serious business applications. Passwords are also useful in a similar manner to prevent unauthorised use of software which may be accessable to many users linked to a network filing system (such as Acorn's Econet system used to link up BBC Micros). Additionally, by incorporating password-requesting routines into the start of programs, the copying of such pieces of software by today's unpopular pirates may, in theory (!), be reduced to a pointless task if the offending pirate does not know the password required to spark his or her illegal copy into life. However, of course, the degree of protection afforded by such a method is only as good as the password routine employed in the program.

A number of password routines have been published before, most of which really just work by looking to see if the password entered by the user is the same as some previously defined sequence of characters. In order to conceal the required password from potential snoopers, various 'invisible' control codes may be placed in the program listing to hide any tell-tale lines of code, or the password may be stored elsewhere in memory.

It may be concluded that many of the different programs that request the entry of a correct password, all have one weakness in common — somewhere within their innards lies a copy of the correct password for comparison with user entries. Considering the vast numbers of supposedly 'protected' games programs that are copies and often modified by many people, I wouldn't imagine the task of extracting a password from a few thousand bytes of data poses too much of a daunting task for many people (with the help of a few PEEKs of course!).

A MORE EFFICIENT ANSWER?

The answer to the problem of using passwords but not storing the actual password required at the same time may be found if one imagines applying any password entered by the user as a 'key'. This key may be used in an attempt to decode or 'unscramble' the bytes which actually make up the main program in question. The program would have been encoded or 'scrambled' previously in an identical manner using the intended future password.

The result of using this technique of encoding is that unless the correct password is entered, the resulting program remains just a meaningless sequence of bytes stored in memory — no use to even the most experienced pirate! The real bonus, of course, is that no copy of the correct password actually exists within the program.

If that all seemed just a bit unclear, don't worry, all will become a lot more obvious — I hope!

THE NITTY GRITTY!

The heart of all this encoding and decoding business lies in the use of the logical operator known as Exclusive –OR (EOR). This enables all the theory to be put into practice from a programming point of view.

Firstly, it's important to understand what exclusive –OR does; this is best shown at a binary level:

Truth Table:

A	B	RESULT
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table above, it may be seen that the operation outputs a value of 1 only if *one* of the inputs is a value of 1. All other combinations of inputs result in 0. Consequently, the result of 5 EOR 3 is 6

ie

```
(5)   0 1 0 1
(3)   0 0 1 1

(6)   0 1 1 0
```

The full potential of the exclusive –OR function for the purposes in which we are interested is only realised when it may be shown how the value of one number will alternate between two values repeatedly when it is exclusive –OR'ed with a constant value. Using our previous example:

```
5 (THE CONSTANT) AND 3:
                        (5)  0 1 0 1     EOR
                        (3)  0 0 1 1
          RESULT :      (6)  0 1 1 0     EOR
                        (5)  0 1 0 1
          RESULT :      (3)  0 0 1 1     EOR
                        (5)  0 1 0 1
          RESULT :      (6)  0 1 1 0     EOR
                        (5)  0 1 0 1
          RESULT :      (3)  0 0 1
                                               etc., etc.
```

Above, using 5 as the key, 3 (the data) is encoded into a value of 6. It is important to note using the exclusive –OR, the original data (3) may only be obtained using the unique key of 5. No other value will work — try it if you don't believe me!

It may now be seen how a long sequence of numbers may be encoded using a shorter sequence of numbers as a key. For example, to encode 5,11,3,14,7,6,6 using 1,2,3 .

```
        (5) 0101    (11) 1101    (3) 0011    (14) 1110
Key:    (1) 0001     (2) 0010    (3) 0011     (1) 0001

        (4) 0100    (15) 1111    (0) 0000    (15) 1111

        (7) 0111     (6) 0110    (6) 0110     EOR
        (2) 0010     (3) 0011    (1) 0001

        (5) 0101     (5) 0101    (7) 0111

4,15, 0,15,5,7 :     'scrambled sequence'
```

The 'scrambled' sequence of numbers 4,15,0,15,5,5,7 may only be 'unscrambled' back to the original sequence of 5,11,3,14,7,6,6 by applying the key of 1,2,3.

From the example shown above, it may be noted that when the last number of the key is reached, the key sequence is merely repeated if numbers still remain to be encoded.

The value of this technique may now be appreciated; the original sequence of numbers (5,11,3,14,7,6,6) may be visualised to represent the bytes constituting an imaginary BASIC program, data file or machine code program, etc. Similarly the key (1,2,3) represents the ASCII codes of the letters in an entered password.

A FINAL TOUCH

Using the method of encoding data just described, we could now go ahead and write the software to do the job. However, a slight weakness is still present in the technique (though by no means a major one), and since the remedy of its elimination is a simple one, it is worth incorporating into the final algorithm.

The weakness is this : if a password entered by someone attempting to access data is the right length and correct except for, say, one character, then the data will be decoded correctly, except for every nth byte in the sequence. Where 'n' will be the number of characters in the password. For example, if, in an attempt to decode the sequence of numbers (4,15,0,15,5,5,7) to its correct form (5,11,3,14,7,6,6) an incorrect key (password) of 1,2,4 was applied instead of 1,2,3, then a partially correct result of 5,11,W,14,7,X,6 would be obtained. Only every third byte would be wrong (W and X). In a case like this, if the data decoded was, for example, a file of text, it may be relatively easy for someone to guess the incorrect letters.

A simple way to stop this happening is to calculate the sum of all the bytes in the password, hence obtaining a 'hash' value. The lower 8 bits of this value are then logically EOR'ed with each byte in the password, replacing the original value each time.

If a simple character in the password is now wrong, the resulting hash value will be different and, after EOR'ing, *every* byte in the password will be adversely affected.

IDEAS INTO PRACTICE

Now that we know the principle used in encoding data this way, we are in a position to put the ideas into practice and produce a program. The flowchart shown represents the main encoding/decoding algorithm previously described. Using the flowchart, it should be possible to write a suitable routine to run on any micro.

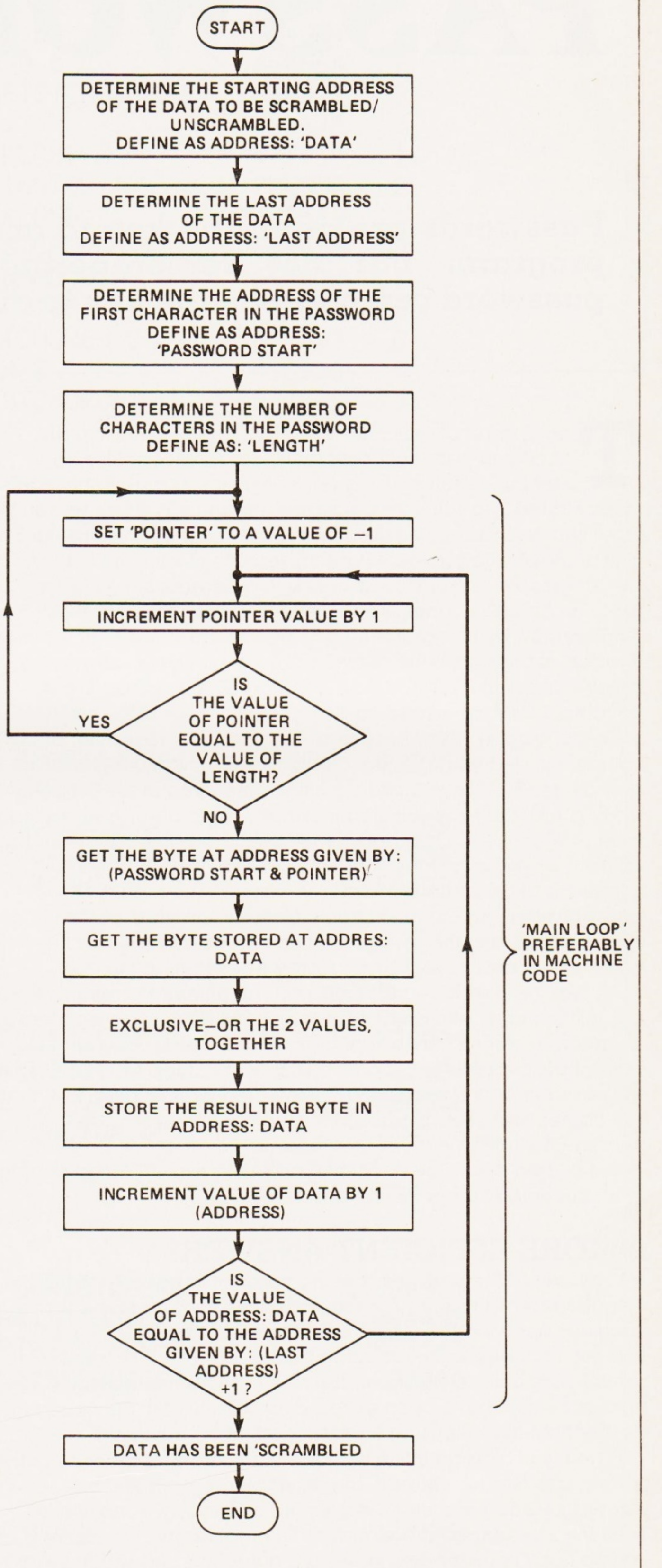
It would be preferable, though not essential, to write the main loop (see flowchart) of the routine in machine code; firstly, to increase the execution speed and secondly, because not all BASIC's include an Exclusive –OR function in their repertoire whereas all the common processor in use do possess such operations in their instruction set.

A PRACTICAL EXAMPLE

The following two programs which I have written provide a useful example of how BASIC programs may only be successfully used when a correct password is entered by a user.

The particular programs will run on a BBC Micro and have been written to suit a system equipped with at least one disc drive. However, the programs could be used by tape users with some minor modifications so don't be put off!

Flowchart describing main encoding/decoding algorithm.


START
DETERMINE THE STARTING ADDRESS OF THE DATA TO BE SCRAMBLED/ UNSCRAMBLED. DEFINE AS ADDRESS: 'DATA'
DETERMINE THE LAST ADDRESS OF THE DATA DEFINE AS ADDRESS: 'LAST ADDRESS'
DETERMINE THE ADDRESS OF THE FIRST CHARACTER IN THE PASSWORD DEFINE AS ADDRESS: 'PASSWORD START'
DETERMINE THE NUMBER OF CHARACTERS IN THE PASSWORD DEFINE AS: 'LENGTH'
SET 'POINTER' TO A VALUE OF –1
INCREMENT POINTER VALUE BY 1
IS THE VALUE OF POINTER EQUAL TO THE VALUE OF LENGTH?
YES
NO
GET THE BYTE AT ADDRESS GIVEN BY: (PASSWORD START & POINTER)
GET THE BYTE STORED AT ADDRES: DATA
EXCLUSIVE–OR THE 2 VALUES, TOGETHER
STORE THE RESULTING BYTE IN ADDRESS: DATA
INCREMENT VALUE OF DATA BY 1 (ADDRESS)
IS THE VALUE OF ADDRESS: DATA EQUAL TO THE ADDRESS GIVEN BY: (LAST ADDRESS) +1 ?
DATA HAS BEEN 'SCRAMBLED
END
'MAIN LOOP' PREFERABLY IN MACHINE CODE

The programs are particularly suited to circumstances as described earlier, where many users may have access to one another's software. An example being in a school environment where a single 'class disc' might be used to store individual pupils' computing projects or assignments. The use of passwords here would help prevent a pupil's ideas (or homework!) being copied.

THE PROGRAMS

Two listings are provided:
(i) The encoder — used to initially 'scramble' a BASIC program using the password entered. The user is then prompted to save the resulting data as if it were machine code.
(ii) The decoder — which requests the filename of the required (scrambled) program present on the same disc, and loads it. A password is then requested. If the password entered is correct, the program runs, otherwise the machine will just crash or report an error. After you have typed in each listing, save it immediately onto the disc to be used for later storage of encoded programs, before you run it. It is important that listings (i) and (ii) are saved under the filenames "ENCODE2" and "DECODE2" respectively.

After you have entered and saved both listings *BUILD the following two loader routines onto the same disc by entering the following (exactly):

```
*BUILD ENCODE       <RETURN>
T% = TOP            <RETURN>
PAGE = &1300        <RETURN>
CHAIN"ENCODE2"      <RETURN>

<ESCAPE>
```

```
*BUILD ENCODE       <RETURN>
PAGE = &1300        <RETURN>
CHAIN"DECODE2"      <RETURN>

<ESCAPE>
```

Both routines ensure that the main programs reside in memory below the actual program being encoded/decoded.

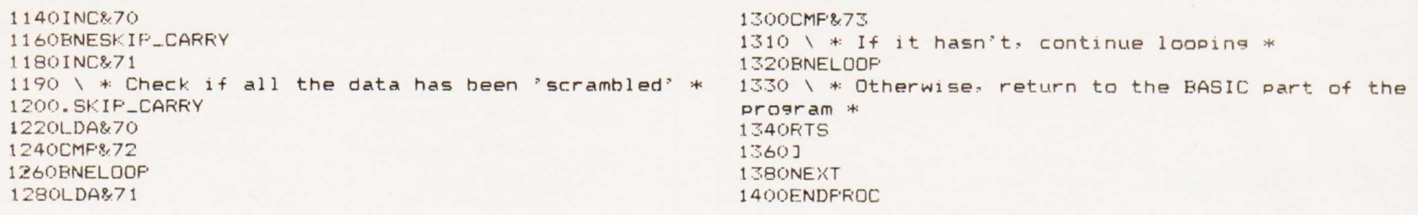
To run either program enter *EXEC followed by ENCODE or DECODE. To encode a new program, you must first load it or type it in (with PAGE at the default value of &1900). Then insert the correct disc and simply type *EXEC ENCODE. After that, just follow the instructions. In future, to load and run a previously scrambled program, insert the disc as before and enter *EXEC DECODE.

A FINAL NOTE

Before encoding your latest masterpiece, it might be well worth saving a NORMAL COPY of it on a back-up disc -- just in case you ever forget the password!

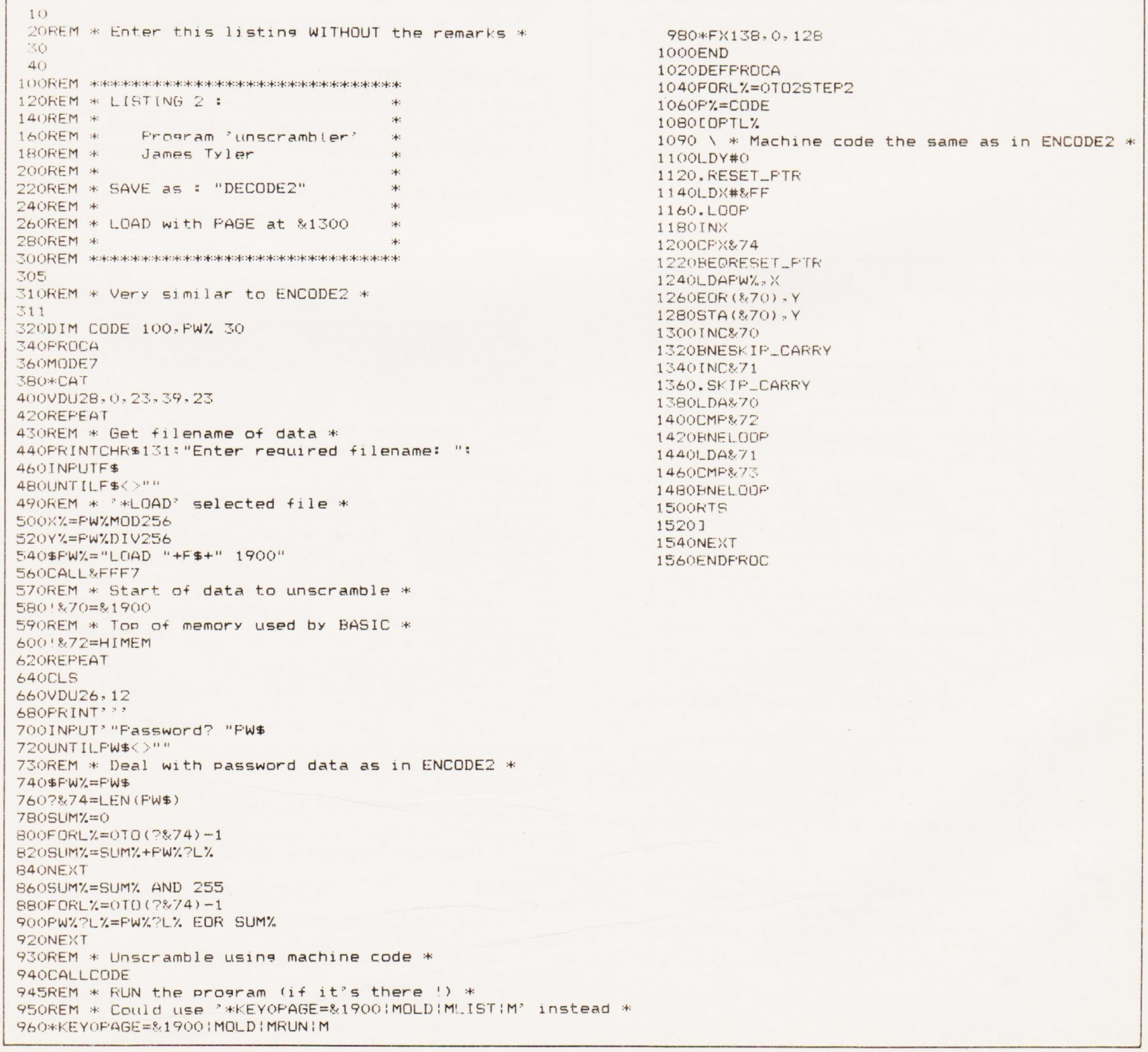
The ENCODER program

```
  10
  20REM * Enter this listing WITHOUT the remarks *
  30
  40
 100REM ******************************
 120REM * LISTING 1 :                *
 140REM *                            *
 160REM *    Program 'scrambler'     *
 170REM *                            *
 180REM *    JAMES TYLER             *
 200REM *                            *
 220REM * SAVE as : "ENCODE2"        *
 240REM *                            *
 260REM * LOAD with PAGE at &1300    *
 280REM *                            *
 300REM ******************************
 320MODE7
 330REM * Reserve space for machine code *
 340DIM CODE 100
 350REM * 30 bytes reserved for password storage *
 360DIM PW% 30
 370REM * Assemble machine code section *
 380PROCA
 390REM * Starting address of prog. to be scrambled
 395REM * Stored in locations &70 and &71
 400!&70=&1900
 410REM * End address of data(+1) in &72 and &73 *
 420!&72=T%+1
 440REPEAT
 460CLS
 480INPUT'''"Enter future password: "PW$
 500UNTILPW$<>"" AND LEN(PW$)<31
 510REM * Store number of characters in password *
 520?&74=LEN(PW$)
 530REM * Store password in memory reserved *
 540$PW%=PW$
 550REM * Calculate SUM of all bytes in password *
 560SUM%=0
 580FORL%=0TO(?&74)-1
 600SUM%=SUM%+PW%?L%
 620NEXT
 630REM * Just get the lower 8 bits *
 640SUM%=SUM% AND 255
 650REM * EOR this value with each password byte *
 660FORL%=0TO(?&74)-1
 680PW%?L%=PW%?L% EOR SUM%
 700NEXT
 710REM * Now enter the main 'scrambling' machine code
 loop
 720CALLCODE
 740CLS
 760PRINT'''"Program has been 'scrambled'."
 780PRINT'"Password is ";PW$'SPC9"(Don't forget !)"'''
 800PRINT"NOW save program, entering:"''
 810REM * Now you can *SAVE the data as machine code *
 820PRINT"*SAVE (name) 1900 ";~T%
 840END
 860DEFPROCA
 880FORL%=0TO2STEP2
 900P%=CODE
 920[OPTL%
 930 \ * Machine code routine *
 935 \ *  Y Register not used - remains at zero *


 940LDY#0
 950 \ * Start of 'main loop' *
 960.RESET_PTR
 970 \ *  X Register used as POINTER *
 980LDX#&FF
1000.LOOP
1010 \ * Increment POINTER *
1020INX
1040CPX&74
1060BEQRESET_PTR
1070 \ * Get byte of password *
1080LDAPW%,X
1090 \  * Exclusive-OR with data byte *
1100EOR(&70),Y
1110 \ * Store resulting byte in original address *
1120STA(&70),Y
1130 \ * Increment address to point to next data byte *
```



```
1140INC&70
1160BNESKIP_CARRY
1180INC&71
1190 \ * Check if all the data has been 'scrambled' *
1200.SKIP_CARRY
1220LDA&70
1240CMP&72
1260BNELOOP
1280LDA&71
1300CMP&73
1310 \ * If it hasn't, continue looping *
1320BNELOOP
1330 \ * Otherwise, return to the BASIC part of the
program *
1340RTS
1360]
1380NEXT
1400ENDPROC
```

The DECODER program



```
 10
 20REM * Enter this listing WITHOUT the remarks *
 30
 40
100REM ********************************
120REM * LISTING 2 :                  *
140REM *                              *
160REM *    Program 'unscrambler'     *
180REM *    James Tyler               *
200REM *                              *
220REM * SAVE as : "DECODE2"          *
240REM *                              *
260REM * LOAD with PAGE at &1300      *
280REM *                              *
300REM ********************************
305
310REM * Very similar to ENCODE2 *
311
320DIM CODE 100,PW% 30
340PROCA
360MODE7
380*CAT
400VDU28,0,23,39,23
420REPEAT
430REM * Get filename of data *
440PRINTCHR$131;"Enter required filename: ";
460INPUTF$
480UNTILF$<>""
490REM * '*LOAD' selected file *
500X%=PW%MOD256
520Y%=PW%DIV256
540$PW%="LOAD "+F$+" 1900"
560CALL&FFF7
570REM * Start of data to unscramble *
580!&70=&1900
590REM * Top of memory used by BASIC *
600!&72=HIMEM
620REPEAT
640CLS
660VDU26,12
680PRINT''
700INPUT'"Password? "PW$
720UNTILPW$<>""
730REM * Deal with password data as in ENCODE2 *
740$PW%=PW$
760?&74=LEN(PW$)
780SUM%=0
800FORL%=0TO(?&74)-1
820SUM%=SUM%+PW%?L%
840NEXT
860SUM%=SUM% AND 255
880FORL%=0TO(?&74)-1
900PW%?L%=PW%?L% EOR SUM%
920NEXT
930REM * Unscramble using machine code *
940CALLCODE
945REM * RUN the program (if it's there !) *
950REM * Could use '*KEY0PAGE=&1900|MOLD|MLIST|M' instead *
960*KEY0PAGE=&1900|MOLD|MRUN|M
 980*FX138,0,128
1000END
1020DEFPROCA
1040FORL%=0TO2STEP2
1060P%=CODE
1080[OPTL%
1090 \ * Machine code the same as in ENCODE2 *
1100LDY#0
1120.RESET_PTR
1140LDX#&FF
1160.LOOP
1180INX
1200CPX&74
1220BEQRESET_PTR
1240LDAPW%,X
1260EOR(&70),Y
1280STA(&70),Y
1300INC&70
1320BNESKIP_CARRY
1340INC&71
1360.SKIP_CARRY
1380LDA&70
1400CMP&72
1420BNELOOP
1440LDA&71
1460CMP&73
1480BNELOOP
1500RTS
1520]
1540NEXT
1560ENDPROC
```

ACT

MICRODEALER

xi APRICOT

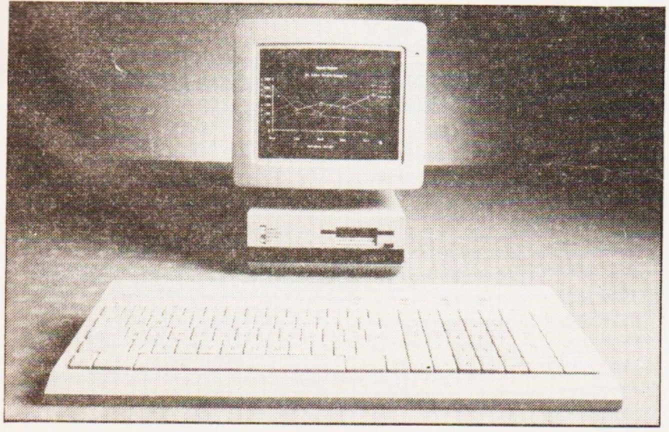
CPU	8086
MEMORY	256K RAM
LANGUAGES	Microsoft BASIC, Personal BASIC
MASS STORAGE	No cassette drive Integral Sony 3½" 315K microfloppy disk drive Integral 5 or 10 Mb hard disk
OS	MS-DOS 2.11 with GSX bundled CP/M-86 (not yet available) Concurrent CP/M-86 (not yet available)
KEYBOARD	QWERTY, cursor, numeric 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.

APRICOT F1

CPU	8086
MEMORY	256K RAM
LANGUAGES	MS-DOS, Concurrent DOS (Optional)
MASS STORAGE	No cassette drive One integral 3½" 720K Sony microfloppy disk drive
OS	MS-DOS 2.11, Concurrent DOS (optional)
KEYBOARD	QWERTY, cursor, numeric pad
INTERFACES	Infra-red link for keyboard or mouse, expansion slot, RS-232C, Centronics
DISPLAY	TV or optional monitor
GRAPHICS	80 by 24 text 640 by 256 four-colour, 320 by 256 16-colour maximum high resolution
SOUND	No

Notes. The Apricot F1 is designed as a low-cost entry-level machine for small businesses (a cheaper cut-down version, the F1 e, is for schools and colleges). It includes several bundled applications including SuperCalc, SuperWriter and SuperPlanner. An optional five-slot expansion bus may be added: also a 10 Mb Winchester unit. There is an optional infra-red mouse/trackball. RAM is expandable to 768K.



CBM MICRODEALER

COMMODORE 64 COMMODORE 715B

Notes: The Commodore 64 is a popular micro with a great deal of games software available. There is also some business software, such as spreadsheets and word processors, available, but this suffers from the lack of an 80-column screen. Graphics and sound have extensive capabilities, for example eight multi-colour sprites and three channels of sound covering nine octaves each.

The Commodore 715B is the top model in the 700 range of business machines. Although built round the 6509 processor, there is a second processor option (8088). The machine has been designed to meet IEC specifications. The black-and-white monitor screen is integral and features tilt and swivel. The keyboard may be detached.

NASCOM MICRODEALER

NASCOM 3

CPU	2 MHZ Z80	DISPLAY	40 or 80 column 25-line display
MEMORY	8K or 32K inbuilt RAM (expandable to 60K)	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)
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	SOUND	No
INTERFACES	RS232 and 16-bit parallel		

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 availab·lity of CP/M means a ready supply ot business software.

COMPUTAMART

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

BERKSHIRE

Micro General
PRINTER SPECIALISTS
Call for advice on printer selection.
Epson & Microline stockists. Interfaces & cables available for all popular micros.
6 THE BIRCHWOODS, TILEHURST, READING.
TEL: 0734 25226

CHESHIRE

northern computers
Churchfield Rd.,
Frodsham.
Tel: (0928) 35110

Open: 6 days 9-6. Retail and Wholesale. Apple II & III, Atom, BBC, VIC20/64, Newbrain, Dragon 32, Electron, Spectrum. All accessories. Easy parking off M56.

CORNWALL/DEVON

A. B. & C. COMPUTERS (CT)
Duchy House, 6 Lower Aylmer Sq., St. Austell.
Tel: 0726 64463/67337

Wide range of popular Micros, Printers, books and accessories. We stock all U need — try us first for service and competitive prices

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

A. J. Duchesne
(Computer Consultants) Limited
Specialists in Small Business Computing
10-12 Creechurch Lane, London EC3A 5AY
Telephone: 01-621 0433

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, Memotech, Oric, Psion, Sirius, Sanyo & Sinclair.
Open 6 days per week

NORFOLK

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

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.

LOOKING FOR MICRO-COMPUTER HARDWARE OR SOFTWARE, THEN LOOK NO FURTHER THAN COMPUTAMART

CLASSIFIED ORDER FORM
COMPUTAMART

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 £17.50!

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:
JASON INSKIP
ASP LTD, 1 GOLDEN SQUARE,
LONDON W1

COMPONENTS

68000 SYSTEM KITS

A range of high quality, double sided, 8 by 8 PCB's which form the core of a 68000 system.

CPU coritains 68000 L8 micro with a mHz crystal clock, dynamic refresh sequencer, bus buffering etc. PCB £80. Component set £110.

B10 provides 8/16 K Eprom, Ram, 20 I/O lines, RS232 interface, counter-timer, interrupt handling etc. PCB £80. Component set £98.

RAM 256K dynamic. Page/address selectable. PCB £85. Component set £245.

A boot loader/monitor is available in EPROMS at £30.

Add VAT to above prices. Delivery 14 to 28 days.

P. L. Watson, 101 Village Road,
Bromham, Bedford MK43 8HU.
Tel: 02302 2867

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.

HARDWARE

NEWBRAIN & SANYO

Professional Micro Computers for the price of hobby machines.

NEWBRAIN ON SPECIAL OFFER

With over £200.00 free software. (Accounts, databases, etc. etc)
Limited Offer-ring now!

SANYO 550/555 COMPUTERS

Micropro Wordstar, Calcstar etc at no extra cost! **Printers:** Epson, KDC, Juki, Shinwa, Daisystep 2000 etc. **Sanyo 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.

FOR SALE

BARGAIN OFFER: Commodore 64, Disk Drive, Printer MPS 801 with software (word processing, accounts, games). Phone 01-654 0808.

SOFTWARE

ATARI: INTERNATIONAL CLUB: Mail order only. You can have world famous cassettes for £6.00 maximum. Hurry! Sous-les-Vignes, CH-1264 Cheserex, Switzerland.

SOFTWARE BUSINESS

DERLITE PAYROLL

WEEKLY OR MONTHLY — BBC MODEL B

Covers all aspects of Tax, NIC, SSP etc.
Saves on up to 85% of your office time.

For up to 20 employees	**£35 + VAT**
For up to 45 employees	**£55 + VAT**

Payslips included. 40/80 track disc. Two discs Less 10%

Full details on request. Dealer Enquiries Invited.

Derlite Ltd., 31 Normandy Way, Bodmin, Cornwall
TEL: 0208 2565

SOFTWARE

COMPUTER SOFTWARE BUSINESS AND GAMES FOR MOST COMPUTERS

Commodore 64, VIC 20, BBC, Atari, Dragon, Spectrum MZ700, ZX81.

New releases for CBM 64, Spectrum, BBC, Atari, Dragon, Spectrum, MZ700, ZX81.

New releases for CBM 64, Spectrum, BBC. Every 2 weeks — just send SAE and we send you a new list every 2 weeks.

Altai C10 data cassettes 40p each. 5 C15 data cassettes £2.40 5¼" floppy discs D/sided — DL density. £2.30 each or 5 for £10.

Send to: **M. J. Seaward,**
St. Olaf's Road, Stratton,
Nr. Bude, Cornwall EX23 9AF
Tel: (0288) 4179

BOOKS AND PUBLICATIONS

PARAPHYSICS Journal (Russian Translations); Psychotronics, Kirlianography, Heliphonic Music, Telekinetics, Computer Software. SAE 4 x 9" Paralab, Downton, Wiltshire.

LIBRARIES

LOOK! Any program for any micro at 20% discount. Also the only tape exchange club specialising in today's TOP 20. (Spectrum, Commodore). Membership free. Write, stating micro, to Ham Software Library, Ham Lane, Elstead, Surrey GU8 6HQ. We've got the lot.

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.

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, software listings, 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 1984 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)

£15.00 for 12 issues UK □
£17.50 for 12 issues Overseas Surface □
£50.00 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)

□□□□□□□□□□□□□□□□□□

Please use BLOCK CAPITALS and include postcodes.

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

ADDRESS ....

....

.... POSTCODE ....

Signature ....

Date ....

CT Feb '85

Argus Tape Magazines add a new dimension to your micro.

Play one of these exciting bi-monthly Argus tape magazines and you'll see why they're Britain's biggest seller. Each issue gives you a variety of exciting and challenging games to play, reviews of other newly released software *plus* valuable utilities enabling you to write your own programs and games.

Stretch your imagination and skills with Argus tape magazines – available every other month from WH Smith, Menzies and other leading stores.

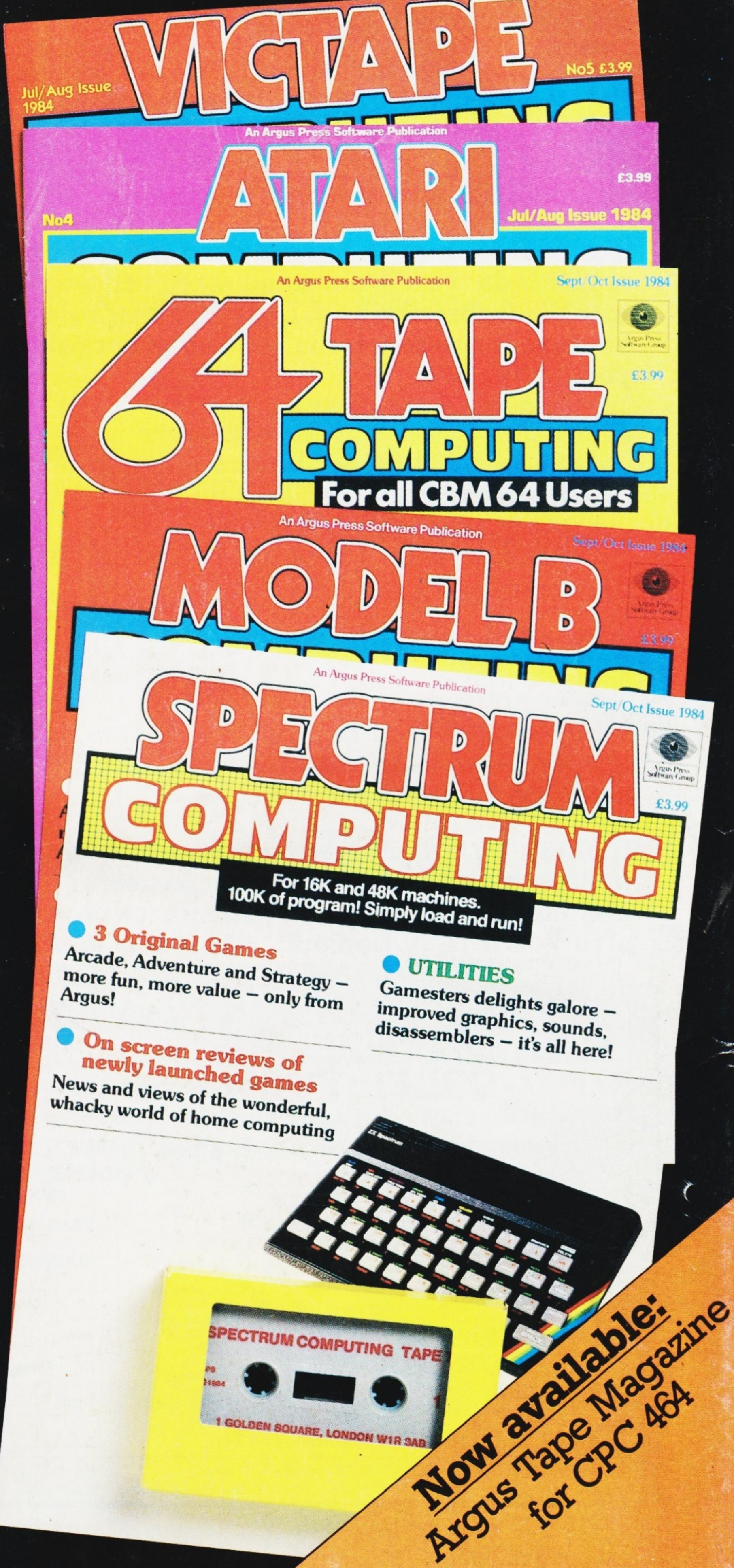
■ **You'll see them advertised on TV from September**

■ **There's already a big demand, so make sure you get your tape magazine today!**

Argus Tape Magazines produced by
ARGUS PRESS SOFTWARE
1 Golden Square, London W1R 3AB
Telephone: 01 437 0626

Argus Press Software Group

For Spectrum, Commodore 64, BBC, Vic, Atari (and from October CPC 464).


VICTAPE
Jul/Aug Issue 1984
No5 £3.99
An Argus Press Software Publication
ATARI
£3.99
No4
Jul/Aug Issue 1984
An Argus Press Software Publication
Sept/Oct Issue 1984
64 TAPE
COMPUTING
For all CBM 64 Users
Argus Press Software Group
£3.99
An Argus Press Software Publication
MODEL B
Sept/Oct Issue 1984
£3.99
An Argus Press Software Publication
SPECTRUM
COMPUTING
Sept/Oct Issue 1984
£3.99
For 16K and 48K machines.
100K of program! Simply load and run!
3 Original Games
Arcade, Adventure and Strategy – more fun, more value – only from Argus!
UTILITIES
Gamesters delights galore – improved graphics, sounds, disassemblers – it's all here!
On screen reviews of newly launched games
News and views of the wonderful, whacky world of home computing
SPECTRUM COMPUTING TAPE
1 GOLDEN SQUARE, LONDON W1R 3AB

Now available:
Argus Tape Magazine
for CPC 464