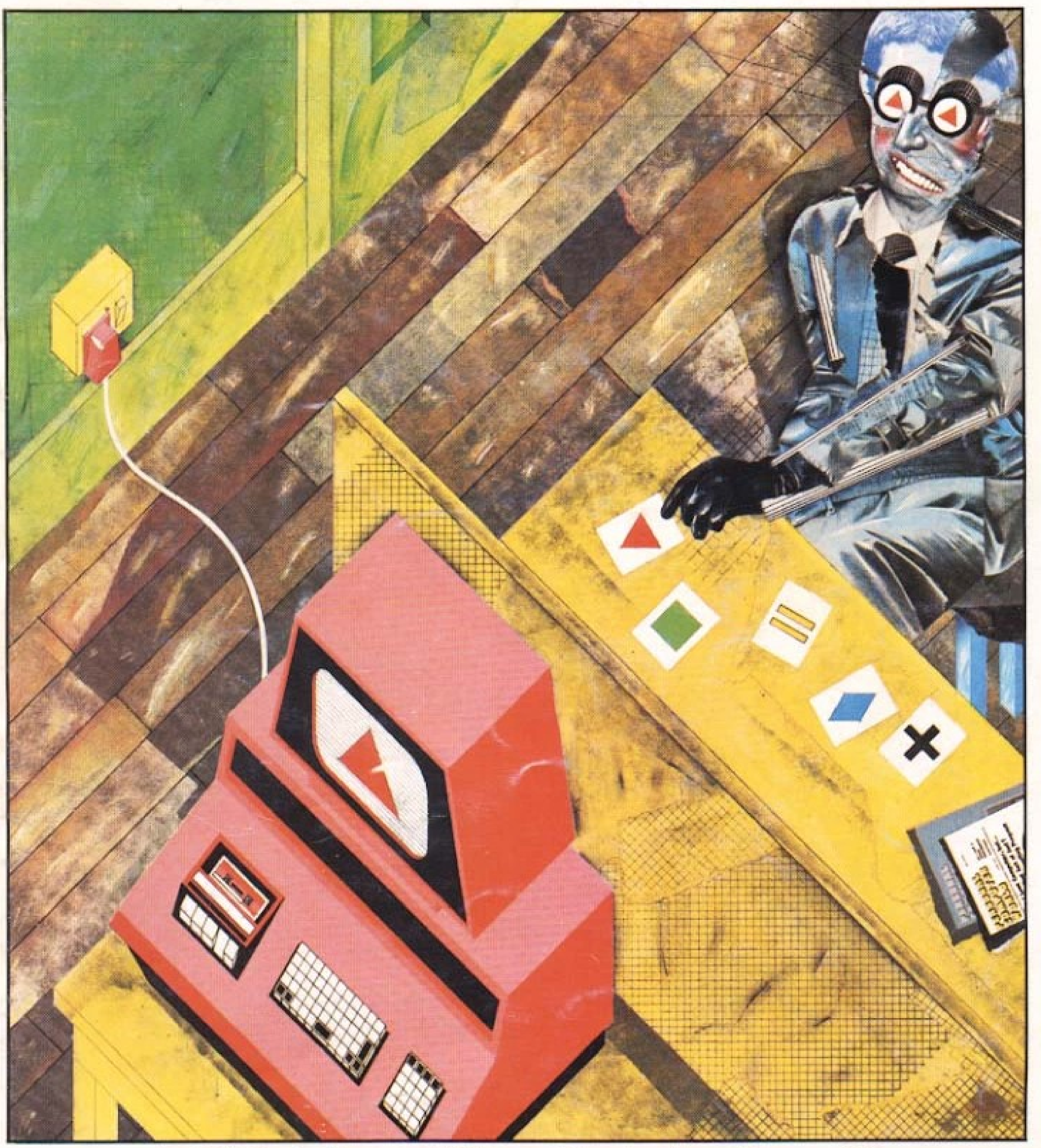
AUSTRALIAN Personal Computer

CROMEMCO SYSTEM THREE REVIEWED

AUSTRALIA'S LEADING MICRO MAGAZINE JUNE, 1980 $1.95

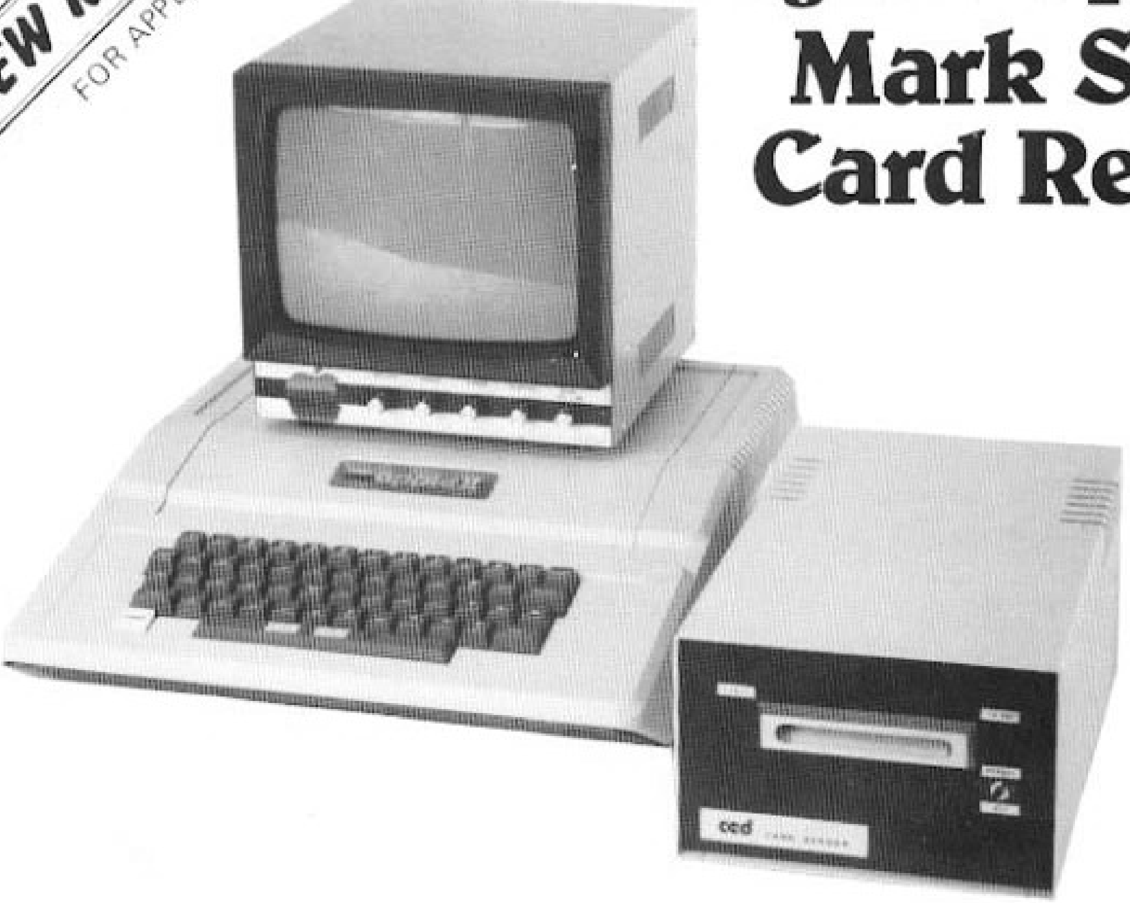


ESP~The ultimate interface?

NEW RELEASE

FOR APPLE II*

C.E.D. Model 1000 Intelligent Optical Mark Sense Card Reader



* DESIGNED FOR EDUCATIONAL USE
* CONNECTS DIRECTLY TO APPLE SERIAL INTERFACE
* APPLE BASIC CARDS GIVE EASE OF PROGRAMMING
* NO SOFTWARE REQUIRED – UTILIZES BUILT-IN MICROPROCESSOR & FIRMWARE
* RESIDENT SOFTWARE FOR APPLESOFT COMMANDS, MULTI-CHOICE QUESTIONS & "RAW DATA"
* CAN BE LOCATED AWAY FROM APPLE FOR REMOTE JOB ENTRY
* DIRECT CONNECTION TO PRINTER OR OTHER OUTPUT DEVICE
* EXECUTIVE SOFTWARE ON DISK GIVES HARD COPY LISTING & RUN OF PROGRAM ENTERED
* LOW COST
* HIGH SPEED
* RUGGED CONSTRUCTION
* HIGH RELIABILITY

Coming Soon!

AUTOMATIC CARD FEED ATTACHMENT

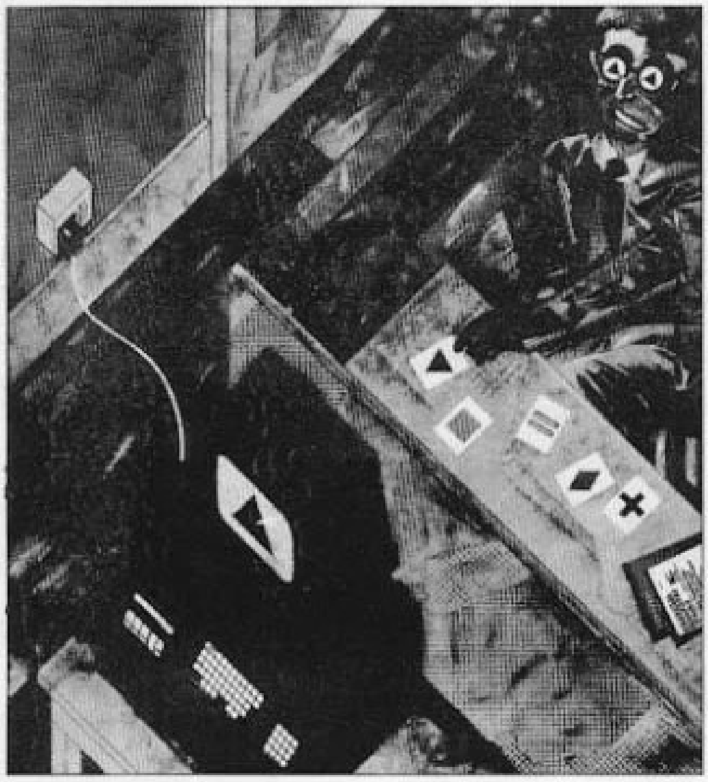
AUSTRALIAN DISTRIBUTORS FOR CED

ComputerLand® in Melbourne

Grd. Floor, 555 Collins Street, MELBOURNE. VIC. 3000.
Phone (03) 62 6737 (03) 62 5581 Telex AA37007.

CONTENTS

Volume 1 No. 2 June 1980



John Skelton's article cancelled due to illness.

Published by Australian Microcomputer Journal, P.O. Box 115, Carlton, Victoria, 3053. Telephone: (03) 82 5783. Telex: AA30333. P.O. Box 250, North Sydney, N.S.W. 2060.

Cover Illustration
George Snow.

Distributed by
Gordon & Gotch (A'asia) Ltd., MELBOURNE, SYDNEY.

Guidelines for Contributors
APC welcomes articles of interest. Don't be put off if your style of writing is "under developed" . . . true worth lies in the content, and shaping features comes naturally to us! Manuscripts should not exceed 3,000 words and authors are asked to use triple spaced lines with a wide left-hand margin; diagrams, listings and/or photographs should be included wherever possible. Please enclose a stamped, self-addressed envelope if you would like your article returned.

Because of the foregoing, it is necessary to add that the views expressed in articles we publish are not necessarily those of *Australian Personal Computer.* Overall, however, the magazine will try to represent a balanced, though independent viewpoint. Finally, before submitting an article, please check it through thoroughly for legibility and accuracy.

Subscription Rates:
Australia $24.00 for 12 issues, New Zealand $30.00 for 12 issues (surface mail). Prices include postage and packing. Supplies to specialist shops can be arranged by negotiation direct with the publishers.

TRS80 MEANS BUSINESS

We are legal agents for Taranto, Apparat, Acorn, Mumford, Shrayer, Archbold, Instant Software, Personal Software, Adventure International, Microsoft, Cottage (U.S.A.), Hayden, Butterworths, and many other fine firms. We always try to stock the best lines of each manufacturer and offer a full replacement service on any programmes bought from us that become faulty or damaged.
Due to extensive contacts in the U.S.A., we can import any item for you, from any manufacturer (from $5 to $10,000). Let us have the worries! The cost to you will be most reasonable. Leasing - lay by - or credit can also be arranged.

IMPROVE TRS—80 PERFORMANCE WITH NEW DOS+

INTRODUCTORY SPECIAL

NEW DOS+ and the very **Best of Apparat's** disk utility programs $99 (a $250 value). 40 Track version $110. **(NEW DOS 35 $49 — NEW DOS 40** ($60).

TRS-80 SOLUTION

HARDWARE

3 speed modification — a very simple kit; only 4 connections, easily reversible	$29.95
Blank computer cassettes — leadless guaranteed no dropout	$1.80
5—$8.80 10 — $16.00 100 — $100.00	
Reverse Video	$31.00
Archbold speed up	$35.00
Electric Pencil L/case kit with instructions	$18.00
Modem "CAT"	$195.00
Green Screen	$35.00

SOFTWARE

Sargon 1	$25.00
Sargon (version 2)	$37.50
Microchess "still the best seller" Level 1 and Level 2 4K	$24.00
Golf and Crossout "2 programmes"	$10.00
Game Playing with Basic "TRS Apple Pet" Tape 1, Tape 2, Tape 3	(each) $12.50
Santa Paravia "Become the Ruler of a Medieval City" Graphics TRS-80 L 1 and 2	$10.00
General Mathematics TRS-80 L1 and L2 Apple 2	$12.50
Space Trek IV "Trade or Wage War"	$9.00
Oil Tycoon	$9.00
Packer	$29.95
Gammon Challenger	$17.40
Codebreaker "Sound"	$14.00
Stock Market	$12.00
Special Delivery (Mail List)	$99.00
Complex Matrix Maths, PET TRS-80 1 or 2 — Apple 2	$12.50
Checker King	$27.00
Flight	$10.00
Airmail Pilot	$10.00
Utility 1 and 2	(each) $10.00
Backgammon / Keno	$10.00
Teacher	$12.00
Ham Package	$10.00
Azimuth Finder	$14.00
Music Tutor	$12.00
Star Trek "Acorn"	$12.00
Bandito	$12.00
German "Disk"	$23.00
French	$23.00
Italian	$23.00
CCA DMS "disk 32K"	$95.00
Stimulating simulations	$21.00
Electric Paint Brush	$21.00
Time Trek "W/Sound"	$21.00
Level 3 Basic	$52.00
Typing Tutor	$17.00
Editor / Assembler	$34.00
Adventure "Original 32K"	$35.00
Fortran	$102.00
Spooler	$21.00
mu MATH	$77.00
Basic Compiler	$202.00
Taipan	$12.00
Alien Invasion	$14.00
System Savers	$17.40
Aterm "Terminal"	$35.00
Print to L Print etc.	$9.00
Structured Basic Translater	$35.00
Cassette Label Maker	$16.00
Disassembler	$19.95
Engineering — Beam Analysis	$15.00
Builders Construction Ledger 32K with 3 disks	$200.00
Investment Analysis — Real Estate Cassette	$15.00
Disk	$20.00
Amateur Radio — Beam Direction Finder Cassette	$14.00
Disk Index	$19.95
Business System complete "even prints its own invoice form"	$210.00
Cribbage / Checkers	$12.50
Chess	$12.50
Star Trek 2	$10.00
Cash Journal "Interacts with Taranto General Ledger"	$50.00

PROGRAMMING AIDS

Combination Coding/CRT forms in 50's — 2 pads	$8.50
Flow Chart sheets 280mm x 406mm in 50's — 2 pads	$10.00
Giant Video Display pads — 5 pads	$37.50

SUNDRIES

Diskettes "Scotch" 5¼ s.s.	1 $7.00
The Best — Don't risk your data	10 for $60.00
with cheaper disks	100 for $550.00
House brand 1 — $5.00	10 — $45.00
Combination QSL album and Log Book	$9.95
Binders for CB Action	$6.50
Binders for Amateur Radio Action	$6.50

MODIFICATIONS

Installations (send keyboard by Comet, Ipec etc.)	
Lower Case (electric pencil/scripsit)	$40.00
16K upgrade kit (Memory)	$85.00
Installation of Memory	$20.00
Installation of Reverse Video	$45.00
Installation of Speed Up Board	$45.00

TRS-80 BUSINESS SYSTEMS (Model 1 — Level 2 Osborne)

General Ledger
Accounts Payable
Accounts Receivable
Invoicing and Inventory

$99.00 each on DISKETTE

DEFOREST SOFTWARE

QUALITY QSL PTY. LTD.
26 Station Street, Nunawading Vic. Australia (03) 877 6946
Post included at no charge, however we suggest certification or registration.
Trade Enquiries welcome.

THE TRS-80 MODEL II PROGRAMS

General Ledger/Cash Journal: handles up to 7000 transactions on 500 different user-defined accounts. It keeps track of them by month, quarter and year, makes comparisons to the prior year, and does departmentalization.
Accounts Payable/Purchase Order: generates the purchase order and posts the item to payable when the goods are received. Invoice-linked, it calculates and prints checks and aged ledger reports and links fully to the general ledger.
Accounts Receivable/Invoicing: keeps track of billed and unbilled invoices, open and closed items, aging and service charge calculation. It prints statements, links to the general ledger, and can work within either an invoice-linked or balance-forward accounting system.
Payroll/Job Costing: computes regular, overtime and piecework pay, keeps employee files, figures taxes and deductions, prints checks, journal, 941-A and W-2 forms, and breaks out individual job costs.
These Model II programs are completely customtailored, which explains their $249.95 price. Before we'll send you a disk, you have to fill out a detailed questionnaire that tells us your precise business requirements. Then we send you the disk, all the instructions you need.

Please send me the custom questionnaires for the following $249.95 Model II Programs.

☐ General Ledger/Cash Journal
☐ Accounts Payable/Purchase Order
☐ Payroll/Job Costing
☐ Accounts Receivable/Invoicing

☐ Please send me information on the TRS-80 Model I programs at $99.95 each

Your name . . . .
Company name . . . .
Address . . . .
City/State/Postcode . . . .

SPECIAL! SUPER "NEW DOS/80" now available $149.00

ADVENTURES

1—8 on Tape $16.00 each, 1 on DISK $19.00
2 on DISK $29.00, 3 on DISK $39.00

LIGHT PEN TRS-80 LEVEL II

BYPASS THE KEYBOARD
INTERACT WITH THE SCREEN DIRECTLY

APPLICATIONS
- o Education
- o Business
- o Games
- o Home

NO ASSEMBLY NECESSARY
READY TO PLUG IN
COMPLETE INSTRUCTIONS AND DEMONSTRATION PROGRAM PROVIDED
TRS-80 PEN Includes Demo-Game Cassette $19.00, Kit Form — $14.00

DUPLICATE SYSTEM TAPES WITH "CLONE"

This machine language program makes duplicate copies of ANY tape written for Level II. They may be SYSTEM tapes (continuous or not) or data lists. It is not necessary to know the file name or where it loads in memory, and there is no chance of system co-residency. The file name, entry point, and every byte (in ASCII format) are displayed on the video screen. Data may be modified before copy is produced.
CLONE. $18.00

*TRS80 is a registered Trade Mark of the Tandy Corporation and products advertised are in no way connected with or guaranteed by the Tandy Corporation.

Editor
Sean Howard

Editor (U.K.)
Bruce Sawford

Technical Editor
David Tebbutt

Consultants
John Coll,
Mike Dennis,
Miriam Cosic,
Michael James,
David Hebditch,
Sheridan Williams,
Dr Adrian Stokes,
Dr Stephen Castell.

Promotions Manager
Marie Pirotta

Promotions Assistant
Josephine McKenna

Production Manager
Michael Thomas

Typesetting & Artwork
Pam Typesetting

Printed by
Nunawading Press,
Victoria.

NEWSPRINT

Beware the Japanese

Watch for new computers from Japan. We just saw an item produced by the Nippon Electric Company that combined in a small keyboard-cabinet a computer with 16K RAM, Microsoft BASIC, colour graphics, disc controller, cassette interface, and motor control for $750 US. For another $750, NEC sells a colour monitor (that's monitor, not TV) which will handle the high resolution graphics and an 80 character display.

Console configuration can be set in BASIC, and there are "Motor" commands to control external devices through the connector at the back of the unit. NEC is not planning to market this model abroad, however. It's holding back in favour of a better, cheaper model to be introduced soon.

Moving Up

B.S. Microcomp has moved from Burwood to the bright lights of 561 Bourke St., Melbourne (4th floor), on the strength of their Commodore Microcomputer.

They have the following disc based software usable on the Commodore CBM System.

The WordPro 111 word-processor offers instant editing, global search and replace, document retention up to 170 pages, etc., etc.

The COMBIS System can store 10 sub-records in each record and 650 records per disc. Info can be extracted from any combination of sub-records and can be directly used by WordPro111.

Australian designed, the General/Creditors/Debtors Ledger can handle oodles of transactions, produces all the standard reports and allows for random enquiries.

B.S. Microcomp also sells games software from Commodore's "Arcade" series. They all include sound effects for use with a loudspeaker unit. Aw, Beryl, it's not the same without the noises!

Tel. 614 1433, 614 1551

IPS–100

The Information Processing System-100 uses a 10 slot S100 bus for standardisation so you can adapt it to other products. With an 8085 processor board, quad density dual floppy discs, 32K RAM and two independent RS-232 I/O ports, it's only 12" wide and can accommodate up to four users on uninterrupted time sharing. The Micropolis operating system, editor, assembler and extended BASIC are included. Other packages are available.

At $3,750 (tax ex.), if the IPS-100 is what you want contact Microprocessor Applications, Maskell's Hill Road, Selby, Vic., Tel. (03) 754 5108 or A.H. 211 8484.

Micro Conference

The National Microcomputer Conference on Personal Computing in the Eighties will be held at the A.N.U. in Canberra. Running over three days, July 9th to 11th there will be four sessions a day on various topics of software, programming and hardware.

Registration fee is $25.00.

Information from
MICSIG - Canberra,
P.O. Box 446,
Canberra City, A.C.T. 2601.

Relieves Headaches

Disc drive head cleaners for the TRS 80 and the Apple II are available from Computerware. In the form of a convenient mini diskette, they are reusable and come with an automating program.

The head cleaner allows more reliable disc drive operation and will save you the cost of conventional head cleaning maintenance. A cleaning solution is included in the price of $20.

Local Suppliers

For those in the Camden/Campbelltown area who don't want to journey to the Big Smoke for their software requirements, Seahorse Computer Services are just a local phone call away. They are agents for Commodore business machines and home computers and supply the full range of Commodore software and peripherals.

Software is also held for Apple, Pet, TRS 80, and Sorcerer; and Lifeboat Associate's professional software can be bought formatted to your needs.

They have a mail order service and a comprehensive catalogue which is available from Seahorse Computer Services, P.O. Box 47, Camden; Tel.(046) 366 131.

Apple Change of Heart

The heart of the Apple II computer has been transplanted – an astonishing bit of surgery that has drastically changed its character. The standard Apple II computer uses the 6502 micro and therefore cannot run any of the software available to users of systems with Z80 or 8080 micros (which have standard CP/M operating systems). This gap in the Apple's abilities has prompted one of the two best-known suppliers of computer control software – Microsoft – to produce a card which gives the Apple II a Z80 processor.

This unexpected grafting allows the Apple II to run the piece of software produced by the other best known systems software company – Digital Research – that is, the CP/M operating systems. The cost of the new processor board and two diskettes (with the CP/M software) is $349.

Microsoft BASIC, included in the package price,"has all the features not found in Applesoft, plus exclusive new features added to take advantage of the Apple II's special capabilities."

And adding the Z80 software card doesn't stop the Apple being an Apple. "It allows you to use either the 6502 or the Z80 processor – whichever is needed to run a particular program," says Microsoft, "and switching between the two processors is as easy as typing in a particular command. All the features that you love about your Apple stay intact."

The only caveat is that programs and data generated on the Apple in CP/M are not transferable to the normal Apple. It's either one or the other . . . never both.

Hard Disc

D.D. Webster Electronics have introduced hard disc versions of the Spectrum II. These new free-standing machines are available singly as a hard disc 10 or 20 M/B unit, or incorporating any of the Spectrum II's single or double floppy disc minicomputers. Three of these machines are already in use in Victoria.

Prices range from $16,077 for a 32K Spectrum II Z10 without floppies, to $23,195 for a 64K Spectrum II D20 using a 20 M/B storage disc with a double-sided double density floppy disc storing 2.52 M/B's on-line.

Contact D.D. Webster Electronics Pty. Ltd. at 17 Malvern St, Bayswater, Vic. Tel. (03) 729 8444.

Double Dutch

ADE have a new two-headed bidirectional character printer called the Qume TwinTrack. So you can now have a word processor which prints text in Swahili, to use their example, or Dutch, to use ours (just dying to use that heading), while simultaneously printing out an English translation. You can print in many colours. You can scrap hand insertion of special characters.

All this and more, with the TwinTrack which was designed to solve complicated printing problems. It uses two independently operating daisy wheel print heads. This allows for 192 on-line characters, with any combination of type fonts.

With two different wheels and a total of 192 characters, the printer is capable of up to 45 characters per second. Two identical wheels can generate up to 75 characters per second, with a usable print width of up to 26.3 inches.

Ten- or twelve-pitch or proportional spacing daisy wheels can be used. For the standard width of 15.3 inches, 183 columns of 12 characters per inch are possible. Using maximum width, 264 columns of ten characters per inch, or 315 columns at twelve characters per inch are possible.

So, if you have an African connection, or need multicolour printing or text involving algebraic symbols, subscripts, superscripts, etc., contact Anderson Digital Equipment Pty. Ltd., P.O. Box 322, Mt. Waverley, Tel. (03) 543 2077; or Box 341, Pennant Hills, N.S.W. Tel. (02) 848 8533.

More from ADE

ADE also have a new family of plotting systems – the CPS 14/15. They offer four pen plotting under program control and will produce four colour drawings on paper, mylar or vellum.

We are told that the digital plotters are designed to provide "draftsman like" quality, in a variety of applications.

The CPS 14 and 15 systems have up to 172 firmware generated symbols, circular buffer memory, and protocol for detection and correction of errors. A thumb wheel selector gives scaling up to nine times the original size.

Contact as above.

ADE yet again

You've got to hand it to them – they won't be going broke from lack of publicity.

ADE have also recently announced the introduction of 80 and 300 megabyte disc subsystems to suit any DEC PDP-11 system as well as Data General, Nova and Eclipse minicomputers.

These subsystems will enable any of the above computers to access up to 1.2 billion bytes of on-line storage in a single configuration. With the use of CARDS (Continuous access reliable disc storage) systems, ADE can extend storage capabilities to a maximum of "2.4 billion bytes of on-line storage". And up to four CPU's can be interfaced to a common drive through the use of CARDS.

The 80 M/B subsystem will sell for about $25,000 and the 300M/B for about $40,000.

You know where they can be contacted.

New at GFS

GFS Electronic Imports, agents for South West Technical Products Corporation (SWTPC), have announced the release of two products to supplement their range of SWTPC/Motorola 6809 based microcomputer systems.

Firstly, a new Central Processor, the Model S/09, features a Motorola 6809 microprocessor, and is supplied with 128K of dynamic RAM which is easily expanded to 768 K.

It will support up to eight I/O ports, programmable for baud rates up to 38,400. Both multiuser and multitasking/multiuser operating systems are available, with Pascal, BASIC, extended business BASIC, assembler, text-editor, text processor, sort-merge, plus a range of business software.

The price of the S/09 is $62,000 (incl. sales tax).

The other newie from SWTPC is the Model CDS-1 "Winchester" technology hard disc drive unit, with a capacity of 16 M/B, for $8,200. Sounds like a weapon!

Both available from GFS Electronic Imports, 15 McKeon Rd., Mitcham, Vic., Tel. (03) 873 3939.

Lisp for the Big Question

The words "Artificial Intelligence" are equated in too many minds with some vague picture of a lunatic scientist trying to create a computer that lives.

A man called Aaron Sloman describes the actual benefits of artificial intelligence much more accurately. In a grotesque summary of his superb book "The Computer Revolution in Philosophy" it could be said that Sloman has found a real use for both philosophy and artificial intelligence by observing how well they solve the limitations of each. In particular, he accuses philosophy of building theories without the barest models of Mind to test them on. By testing concepts of Mind in artificial intelligence one can quickly find out if it is possible to create a set of rules that express the concepts; and from that, whether the rules (simulated on a computer) actually produce "behaviour" that parallels observable intelligent behaviour.

All this is by way of saying that the lauguage Lisp, used by artificial intelligence researchers, is

SOFTWARE

AUSTRALIA'S MOST COMPREHENSIVE CATALOGUE

SEAHORSE
supplies more than
470 programmes from:
MUSE / PROGRAMMA / MICROSOFT
MICROSENSE / LIFEBOAT
GEMSOFT
for the
APPLE
CBM 'PET'
TRS-80
SORCERER
$1.00 inc. p & p refundable with order

SEAHORSE COMPUTER SERVICES
P.O. Box 47, Camden, N.S.W. 2570
Old S. Rd. Razorback, N.S.W.
Telephone: (046) 366 131

eventually going to be the sort of tool that enables someone to define "sense of humour" accurately, and freeing philosophers from protesting that, sure, they're concerned with the problems of real life – like, what do we mean by "real" and what is an empirical definition of "life"? (with acknowledgements to The Hitch Hiker's Guide to the Galaxy) And Lisp is now available from Microsoft, together with another product written by the Soft Warehouse, for micros. The package is called muLisp (microlisp) and costs $200, to run under CP/M. The other package is a maths package, muMath at $250.

The muLisp package "offers all of Lisp's unique programming features," says Microsoft: "including 83 Lisp functions, flexible program control structures, and infinite precision integer arithmetic in any desired radix from base 2 (binary) to base 36." Dealers, contact Microsoft on (206) 455 8080 in the USA.

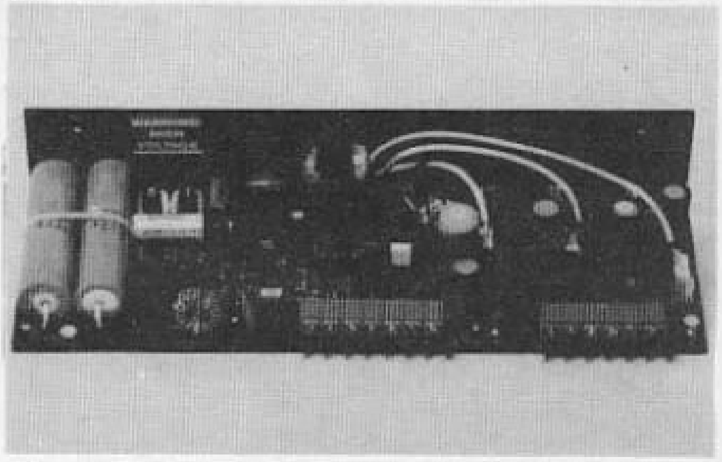
In Australia, LISP is available for the Cromemco system. At $295 for either a 5" or 8" diskette, documentation includes a detailed manual and a copy of "Artificial Intelligence Programming". From Informative Systems, 3 Bank St. Sth. Melbourne Tel. (03) 690 2284.

Beatitude

The heart warms to a company which can provide clear pictures and polite correspondence. MicroPro Design have advised us of a new product – the California DC range of switching regulated power supplies.

The LR series of open frame supplies provide size and weight savings over linear, series regulated supplies, we are told. They provide power for floppies, tape drives, memories and microprocessors, and include versions with up to four independent output voltages.

If their machines are as efficient as their photographers and reps, they're onto a winner. Enquiries to MicroPro Design, P.O. Box 153, Nth. Sydney, Tel. (02) 438 1220.



Hop on the TRS-80 bus

When Tandy launched its packaged microsystem, everybody pointed out that it would never have the benefit of being a standard add-on bus machine, like Altair, and that the job of expanding it would, therefore, be costly. Ingenuity will always find a way to prove these predictions wrong. In New Jersey, a company called Hartmann-Lang is releasing add-on-hardware for the TRS-80, by the devious method of using the official connector socket at the back of the computer, and expanding it to what Hartmann calls the STD bus – a 56 pin 8-bit bus supported by "products from numerous manufacturers".

Hartmann-Lang now has available the following products for the TRS 80/STD bus: the bus interface, a 6 slot STD bus mother board, a 16 channel ten and twelve bit analog to digital converter board, a four channel 12-bit digital to analog board, an eight channel reed relay output board, an eight channel triac board – useful for varying the speed of electric motors or the intensity of lights – plus a range of memory boards, input output ports, and printer interfaces.

Software available includes an intriguing emulator of the 6502 which, if it works, will allow the TRS-80 to run any program written for the Kim or PET family in machine code. There is also a collection of operating utilities, to make loading information from cassette easier, and to help with machine code programming.

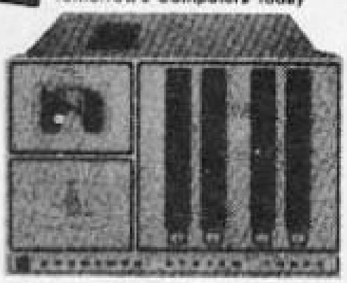
For the freaky, there is a monitor program – written in BASIC – which "makes an interesting case study in interaction between machine code and BASIC", comments the company. At least it isn't claimed to be any kind of useful breakthrough.

From the Horses Mouth

Mr Gower Smith of Computerland, whose family have a Romny Stud in NZ, informs us that the set of programs called the Stud Book, can be of great benefit to livestock breeders. Using an Apple II micro, the programs have been developed to handle selection and breeding requirements in Australian conditions.

That last one was tempting but Mr Smith is too nice to be cruel to. So we'll leave EWE to flex your RAMS, read past the BULL and HEIFER nice month!

Cromemco
Incorporated
Tomorrow's Computers Today



In the microcomputer field, the Cromemco System Three and Z-2H Winchester Hard Disc Systems stand alone in the range of features and capabilities offered. These systems are based on the Z-80A chip, and have from 1-4 mbytes of diskette storage, and from 10-80 mbytes of hard disc storage, combined with the widest range of software available in the industry, including Multi-user, Multi-tasking operation.

The computers have a large S100 motherboard and the operating system is a Superset of CP/M, thus allowing a wide range of non-cromemco hardware and software to be used. This also provides "obsolescence insurance". Some of these features include high resolution colour graphics, Eprom programmers, remote terminal emulation, and card reader interfaces.

Cromemco Basic, available in 3K, 16K, and 32K structured/KSAM versions, is fast, efficient, and ideal for teaching purposes because of it's dynamic error trapping on entry, and easy file handling. Cromemco Fortran IV and Cobol are equal in power to those found on mainframes, and of course, Pascal, C, and other high level languages are also available.

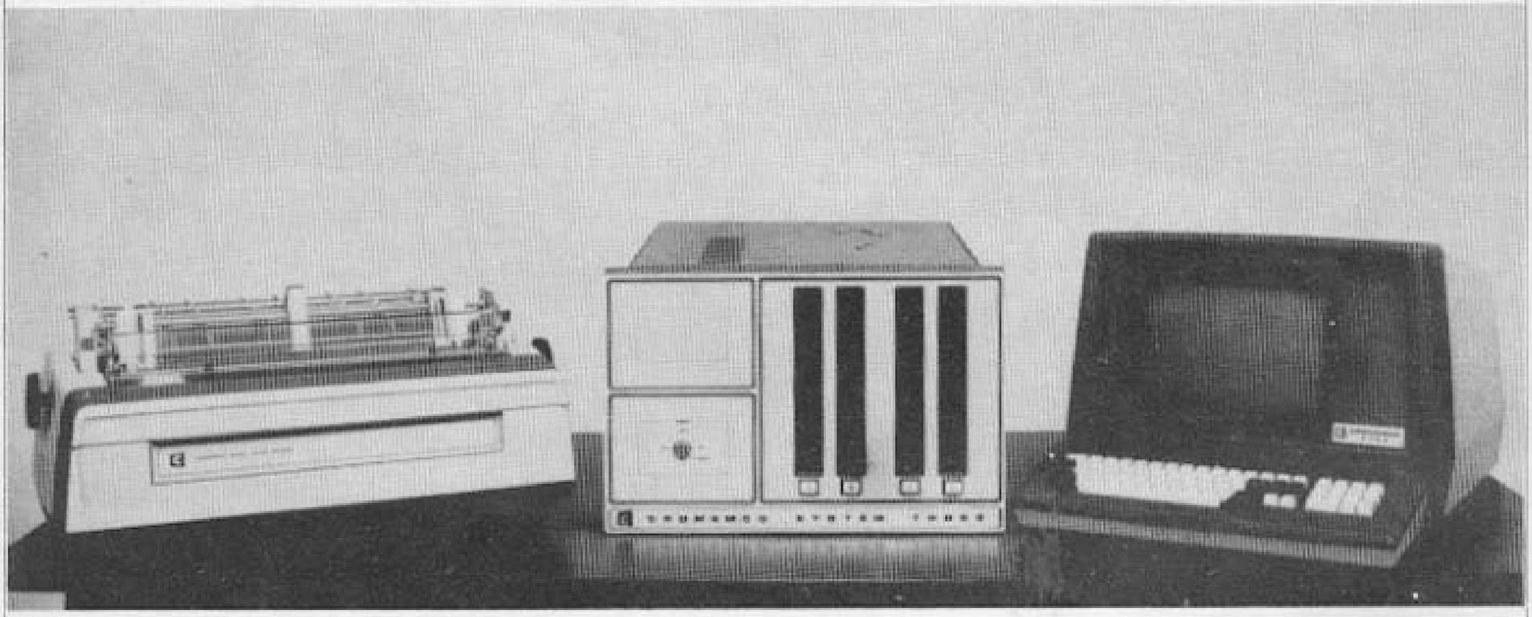
Informative Systems, Cromemco's authorised centre for sales and service, have installed many systems throughout Australia, backed by Cromemco trained technical staff offering maintenance, support and user training.

INFORMATIVE SYSTEMS Pty Ltd
Specialists in professional microcomputers and high performance computer support products.
3 Bank Street, South Melbourne, Vic 3205

Sydney (02) 680 2161
Telephone (03) 690 2284 Telex 30458 INSYST

CROMEMCO

For some time now anyone with a bulky wallet and a strong friend could walk into one of several shops and emerge with the top of the market microcomputer, Cromemco's System Three. As a 'black-box' machine running purchased software the System Three is less appropriate than some of its more compact, less expensive alternatives. But, for someone looking for a microcomputer, the System Three is a thoroughly professional machine. There is nothing gimmicky about the hardware or the software. The hardware is solid and based on the S100 bus; the operating system is more extensive than CP/M but will run most CP/M system software. The supporting software is reliable, comprehensive and well documented.



Hardware

The System Three is contained in one large, heavy box (all-up weight around 35 kg). To open it up one simply presses a button and the front swings open. Immediately visible are the card cage, the two drives and the power supply; with the cage slid forward, every card is completely accessible. The hardware layout has been carefully designed so that everything can be reached with complete ease. The machine that I was lent ran cool even over extended periods of time. I'm sure that this was aided by the emtptiness of the box and the size of the power supply.

The CPU is Cromemco's 4MHz, S100 compatible, Z-80A CPU – called ZPU (Z rhymes with C in American). It has power-on jump circuitry so that it can force an automatic jump to 1 of 16 memory locations. The clock can be set to 2 or 4 MHz (set at 4 on the System Three). There is a Wait State generator allowing use with slow memory.

The memory consists of four Cromemco 16KZ dynamic RAM cards. They can operate at 2 or 4MHz. A Bank Select feature allows memory to be organised into up to 8 banks of 64K, each selected by software. Cromemco also produce a memory card called the 64KZ, which can be used with their Extended Bank Select feature and external motherboard to expand the memory to 4 megabytes.

The disc controller card is Cromemco's Model 4FDC which interfaces with both mini and full size floppy discs. The card contains a serial I/O RS232 port with software-selectable baud rates (from 110 to 76,800). After powering-up one hits the return key a few times so that the System Three can set the baud rate. Also on the card is a 1K, 2708 PROM with system software and several switches for both enabling and disabling initialisation of discs and for booting specific software.

The disc drives are Persci's model 299B. The discs are in IBM 3740 Diskette Format. Data capacity of 0.5 megabyte per disc, located on 77 tracks Reading and writing transfer rate is 250 kilobits per second.

For the Benchtest, a Centronics 779 printer came connected to a Cromemco PRI parallel printer interface board. This board provides two parallel interfaces for simultaneous operation of a dot matrix printer and daisywheel printer.

Overall, I found the hardware well designed and substantial. For the two weeks that I had the system at no time did it show any sign of distress.

System Software

There are two operating systems supplied with the System Three. On powering-up, first of all the Resident Disk Operating System (RDOS) is booted in. Located in 1K of ROM on the disc controller card, RDOS contains a rather primitive monitor with fourteen commands (for reading, writing and displaying data on discs and in memory). More importantly, RDOS contains a bootstrap loader for the Cromemco Disk Operating System (CDOS), if you don't want to use RDOS, there is a switch on the disc controller board that can be flipped so that CDOS will automatically boot in on powering-up. Additionally, if a System Three is going to be used by an untrained operator, there is also, when powering-up, a facility to enter directly into an application program.

CDOS will appear familiar to CP/M users because although it's written in Z80 machine code (instead of CP/M's 8080 code) by Cromemco, they use the CP/M data structures and user interface under licence from Digital Research. Cromemco claim that most CP/M programs will run under CDOS but that programs developed under CDOS probably won't run under CP/M. I assume they are referring to system programs such as compilers and interpreters and not BASIC programs, since Microsoft Extended BASIC and Cromemco Extended BASIC have incompatibilities.

Under CDOS, both main memory and disc memory are divided into two parts . . . one for the operating system, the other for the user. CDOS takes up the lowest 256 words, the highest 8K of RAM and the first two tracks of a system disc. When booting in CDOS the system locks on drive A. This is in fact the default drive and the system prompts with 'A'. A user can move to another drive by specifying it (B, C, or D) and CDOS then prompts with the appropriate letter. Any command typed in response to the prompt is interpreted by a system program called CONsole and PROCessor. If it is an internal function (part of CDOS), a utility or user COMmand file name, then the command is executed. Otherwise the message, "program not found" is displayed, followed by CDOS prompt.

In any computer system there are tasks that users need to execute regularly in order to keep the system running smoothly. As is common practice, Cromemco provides a set of utility programs for some of these tasks.

Unlike the intrinsic commands, running a utility program takes up user space, as do user written COMmand files. The language translators (BASIC, COBOL, FORTRAN and Macro Assembler) are also supplied as COMmand disc files and therefore accessed by typing the appropriate name.

The feature that Cromemco appear most proud of (and which they believe makes CDOS superior to CP/M) is the power of their system calls. These are instructions which are passed directly from program to CDOS without any modification from the language translator. They are designed to handle I/O to and from specific peripherals, load subroutines from a library (such as multiplication and division), etc. CDOS has 151 of these making it possible to transfer programs to a wide variety of CDOS configurations; unfortunately this makes it more difficult to run the same programs under other operating systems.

The major design fault (which it shares with CP/M) concerns the allocation and deallocation of file space. When a file is ERAsed, no file rearrangement occurs and therefore space on a disc becomes increasingly fragmented. If records are not completely filled, the "holes" in the records will not be empty but rather contain whatever was previously on the disc. Also disc space is allocated in 1K "clusters" so that a 13 byte datafile takes up 1K.

Overall, I found CDOS easy to use, if a little intolerant (ERASE AFILE produces the message "program not found" since one must enter ERA AFILE). However, for people familiar with CP/M or DEC software, it should cause few difficulties. In fact, I inserted a CP/M disc in Drive B, read its directory, transferred a COMmand file onto (CDOS) disc A and then executed this program, without experiencing any problems. Even so, I'm not convinced that Cromemco made the right decision when they decided to bypass the industry standard CP/M. It means that users are far more dependant on Cromemco for software, although, from what I've seen, their standard is high. On the other hand, by writing their own operating system, Cromemco have fully utilized the Z80 processor, while retaining CP/M compatibility.

Text Editor

I found the text editor comprehensive, although awkward to handle. It can be used to create, edit and save text on program files. The list of single letter commands that are available is impressive – in fact, only the letter M is not a command. There are line, character and word orientated commands, as well as several commands for moving text between buffer and disc. There is a macro facility for sequences of commands that are needed repeatedly as well as conditional commands that allow the user to execute part or all of a macro.

Although quite powerful, I have certainly come across more convenient editors. When used as a character or word orientated editor the position of the cursor is crucial . . . and yet there is no method for seeing it.

Cromemco have a screen orientated editor that bears a remarkable resemblance to the American UCSD screen editor, used in one of their expensive VDU's.

BASIC

The most significant fact about Cromemco's 16K Extended BASIC is that it was not written by Microsoft. As system programmers can only get so many facilities into a given size memory, this means that some of Cromemco's unique features preclude their BASIC from having all of Microsoft's capabilities. The two most outstanding features missing are variable names longer than two characters and IF . . THEN . . ELSE. These omissions alone should ensure that Microsoft BASIC programs will need line by line revision before they can be executed on a System Three.

Having pointed out that this isn't the best BASIC for running "off the shelf" software, it's still worth looking at the language itself. The lack of IF. . THEN. . ELSE. . and decent size variable names make it quite difficult to produce readable programs. On a more positive note, one of the first things I noticed was the speed with which BASIC programs executed and also the accuracy of data.

The speed is partially due to the nature of the interpreter itself. Rather than using a pure interpreter as do most microcomputers, after entry, each line is translated into machine code. If a line cannot be translated, an error message with a $ under the offending characters appears. The advantages of separating translation from execution are several. Firstly, it is faster to execute a program in machine code than in BASIC. Secondly, a program is more likely to run because at least there are no syntax errors. And thirdly, it takes up less space and hence can be transferred to and from the disc more rapidly. Since a machine code program cannot be edited with the text editor and users may want to edit long BASIC programs, Cromemco offers two ways of filing BASIC programs on disc – SAVE-LOAD for machine code and LIST-ENTER for ASCII BASIC listings. When listing a program the system produces a reconstruction from machine code which is unlikely to be spaced as originally entered (FOR-NEXT loops are indented one space; everything else is left-justified).

The other unusual feature, which accounts for the accuracy as well as influencing speed, is the method of representing numbers. The user can specify if numbers are to be held as 14 digit (8 byte) or 6 digit (4 byte) reals or as integers (2 bytes). When running Benchmark 7 – using integers – it took 73% of the time needed for short reals, showing the kind of savings that are achievable if reals are not essential in processing. Reals are represented in BCD format, fully utilizing the Z80's BCD instructions. This allows the System Three to execute BASIC programs faster than less accurate machines. Unfortunately mathematical functions are slow in BCD, although they are usually accurate. (BM.8, which tests them, shows it also doesn't help to turn numbers into short form).

The list of reserved words in the BASIC Box indicates most of the strengths and weaknesses of Cromemco's Extended BASIC. In particular I noted the following features:

1. TRACE and NTRACE are for de-bugging.
2. RENUMBER leaves gaps in the numbering if lines have been deleted.
3. The initialization instructions are used to get numbers and angles into their most suitable form.
4. CON continues execution after stop.
5. For files (both sequential and random) PRINT and INPUT are ASCII, PUT and GET are machine code form.
6. BINs perform logical operations on 16 bit operands.
7. FRA gives the fractional part of a number.
8. NO ECHO can be used to prevent user's input from appearing on the screen (passwords).
9. ESC is used to break into an executing program and return to the BASIC system. NO ESC allows the programmer to disable ESC.
10. ON ERROR allows a programmer to trap non fatal errors.

I think from the list of reserved words it can be seen that Cromemco 16K Extended BASIC (which actually takes up 19K) is quite powerful. I preferred the Multi User BASIC which was 16K Extended BASIC without machine level instructions, plus 31 character variable names, IF . . THEN DO. . ELSE. . END. 0 and COMMON. Multi user BASIC also now has a Multi tasking facility.

Benchmark

I timed the Benchmark programs in the modes that would be appropriate for execution. . . and then applied some new tests to evaluate the disc file accessing facilities. All the files in these tests are 100 record files with 256 character records. These sizes have been chosen, not only because they are realistic, but also because they are large enough to allow for the significant improvements in access times which new technology promises.

Disc Tests

Test 1	Create "Datafile", open it, close it.
Test 2	Using a FOR-NEXT loop, put 256 'A's into A$ create "datafile", open it, using a FOR-NEXT loop write A$ to records Ø to 99, close the file.
Test 3	As test 2 but writing the records to the file starting with the last record, that is the FOR-NEXT loop's step is -1.
Test 4	Open "Datafile" using a FOR-NEXT loop, read each record out of the file, close the file.
Test 5	As test 4 but reading from the file starting with the last record.

Test 3 and Test 5 are designed to evaluate the "directness" of the file's random access facilities.

Disc Test 1	2.9	Disc Test 4	25
Disc Test 2	115.4	Disc Test 5	40.4
Disc Test 3	115.4		

Other Languages

Besides the Extended and Structured BASIC interpreters, Cromemco supply a Z80 Macro Assembler and FORTRAN, COBOL. PASCAL, ALGOL, RGP2, LISP, RATFOR and PILOT compilers. All come with comprehensive de-bugging packages. A program can be saved in machine code form as a COMmand file and executed just by typing its name.

The compilers and assembler give the System Three great flexibility. I would be surprised, though, if they get used anything like as often as the powerful, fast, easy-to-handle BASIC.

Business Potential

How one judges a business system depends on the use that's intended for it. If what is wanted is a black box that will run ready made packages, there are certainly cheaper, more compact microcomputers on the market. . . eg.Tandy's TRS80 or the Apple II. Also, although Cromemco say that software produced under the CP/M operating system will run on the System Three, their BASIC is sufficiently different from the industry standard Microsoft Extended BASIC that packages produced to run in this BASIC under CP/M will need customizing.

If on the other hand, the intention is to use a microcomputer for a variety of tasks (that include a substantial amount of development work) it is difficult to fault the System Three. The documentation is comprehensive and clearly written; most of the available software is accessible to both the novice who is prepared to devote some time as well as to the more experienced user. There is a wide selection of software enabling development of good business programs. These include a COBOL compiler, a FORTRAN compiler with file handling facilities, a powerful, fast Extended BASIC interpreter, a Structured BASIC interpreter, a macro assembler and a comprehensive text editor.

I was also given documentation for the following packages designed to be used in a business environment: Data Base Management System, Text Formatter, Multi-User BASIC, and several general business packages. The DBMS is an indexed sequential filing system that can be utilized without any programming knowledge. By following the manual, users can create a file, examine and alter the layout of its fields, enter new data, sort on multiple keys, display, insert, delete and query records and print out mailing labels. In addition, the files created by the DBMS can be accessed in BASIC.

The Text Formatter appears to be a comprehensive Word Processing System. Unfortunately it is not memory mapped so it is not as easy to use as those systems now on the market with screen editing. Although available, I was unable to evaluate the Multi-User BASIC; the system I had was not configured for it. From the documentation, it seems that if several users need to access the discs simultaneously, due to the protection system (that is designed to prevent users from corrupting each others files) they would find it painfully slow. If Multi User BASIC is going to be used for development work, rather than file accessing, it probably has a reasonable response time and it definitely has a very powerful instruction set.

Moving on to the hardware, the large box with places for four disc drives and twenty one S100 boards (the industry hardware standard) means that the System Three can be easily expanded without the added expense of boxes, cables and power supplies. The discs are double sided single density and therefore only two drives are needed to put 1 Megabyte on line. If 2M are not sufficient, Cromemco provide the HDD – an add-on system which offers one or two 11 Megabyte Winchester Discs. If the thought of a System Three sitting on top of a desk is unappealing, a custom desk will soon be available to support it; more reasonably, a standard 19" rack mounting will do the job.

In conclusion, for a business that wants a microcomputer in order to develop and run programs, the Cromemco System Three has much to recommend it. Firstly, it is available, by which I mean you can actually go into a shop and walk out with the hardware and software described; secondly it is an extremely expandable system and the add-ons are also easily available; thirdly Cromemco is a financially secure company and therefore there is no reason why they shouldn't continue to produce quality hardware and software.

Educational

How one judges a microcomputer as an educational machine depends on the functions it has to perform. The design of the System Three with its substantial box to cope with expansion, could be either desirable or undesirable. If you want something to move easily from room to room, then go for a smaller system. If, on the other hand you'll want to be plugging in all sorts of unusual boards, then the System Three is ideal; it has fourteen free S100 slots.

The software also has its pros and cons. On one hand there are the nine language translators available, so one isn't locked into BASIC. And, for teaching programming, their new Structured BASIC is an improvement over Extended BASIC. On the other hand, the System Three is not software compatible with the Apple II . . . which seems to be a popular educational machine. To compensate for this deficiency, fourteen diskettes are available each concentrating on aspects of secondary education.

I'm not convinced that any 8 bit micro with floppy discs can adequately stand simultaneous use. But a System Three with Winchester Discs should make a multi user system that can be compared with mini multi-user BASIC systems.

The System Three seems to be extremely robust in both hardware and software. Although too large to be carted around it's a reasonable choice for an institution that wants a mini but can't afford it. Several Cromemcos would probably be cheaper, more flexible and reliable than a single, larger machine.

Home and Games

This section of the Benchtest becomes more or less redundant. Even though Cromemco offer joysticks, they have not made any serious attempt to produce a hobby machine.

Documentation

Along with the System Three, printer and VDU, I was supplied with seventeen manuals. That included nine language manuals (Extended BASIC, Structured BASIC, PASCAL, ALGOL, PILOT, COBOL, FORTRAN, Trace Simulator for the Macro Assembler and Multi User BASIC), two text orientated manuals for editing and word processing, two operating system manuals (for the CPU board, the memory boards, printer interfaces, the disc controller and two on the disc drives) a graphics manual and a Data Base Management Systems manual. No one could accuse Cromemco of skimping on documentation!

Unless perhaps the sheer volume of paper proves rather too daunting, a novice could learn a fair amount about computers and programming from reading them. In particular, the manuals aimed at the new user (16K Extended BASIC, DBMS and Text Formatter) explain at some length the underlying theory of the software in question. The DBMS manual explains how files are organised, while the BASIC manual has an introductory chapter on languages, the types of problems that can be solved by programs and how the System Three responds to a BASIC program. This manual also includes a glossary of general computer terms.

For the more experienced user, these details can be easily skipped over because the manuals have comprehensive tables of contents, indices, appendices and user guides. The introductions also indicate for whom each chapter is aimed. In the programming and text manuals, each instruction and data type is clearly defined and provided with examples. There are also ample programs and text that describes a variety of features in a realistic context.

You can gather that I was most impressed with the overall standard of documentation. It's clearly written, well organised and informative.

Expandability

The System Three is designed with expansion in mind. The computer I was lent had 14 empty S100 slots in its back-plane. The disc-controller can be used with up to 4 drives (5¼ or 8 inch) and there is both the location and power available for the additional two drives. Regretfully, the discs must be single-sided, single-density. In addition, two 11M-Byte Winchester discs, with their controller, can be hung on a System Three. There are 10 places for peripheral sockets (VDU's, printers etc.).

Conclusion

The impression that the System Three gives is that it is an extremely professional machine. The hardware, software and documentation have all been systematically produced and Cromemco appear to be continuing their development work.

The machine is too expensive for the odd small business application . . . certainly for someone who does not know how to program and who is not planning on hiring a programmer. It does not even come with a set of games programs. But for the buyer who two years ago, would have bought a small mini, this machine offers a financially attractive micro alternative.

Technical Data

CPU:	Z80A, 4 MHz
Memory:	3K – 512 K
Keyboard:	Dynamic RAM
Screen:	Lear Seigler Adm3
Cassette:	N/A
Disc Drive:	Up to 4 drives, 2 heads per drive, 8" discs, single density
Bus:	S100
Ports:	1 serial, 1 parallel, expandable to 10
System Software:	CDOS
Languages:	Extended BASIC, Structured BASIC, FORTRAN, COBOL, Z80, Macro Assembler Multi User BASIC, LISP, RATFOR, RGP2 and PILOT.

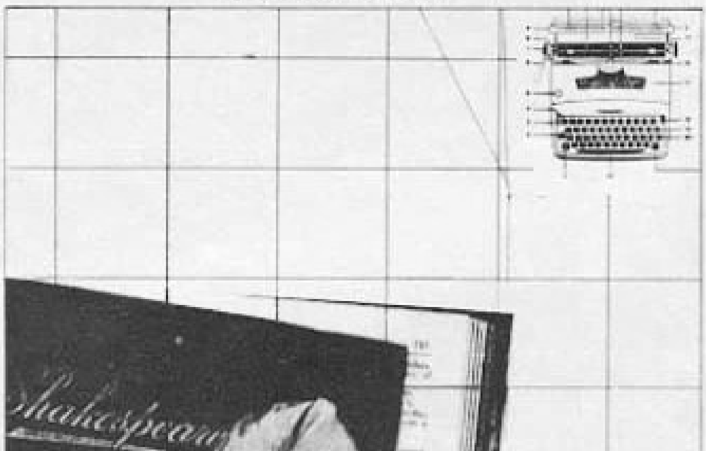
1.	CS3	64K Memory two disc drives	$6990
2.	CS3-002	Dual drive expansion (8")	$2495
3.	HDD-11	Hard disc subsystem	$6545
4.	HDD-22	Dual hard disc subsystem	$11500
5.	JS-1	Joystick Console	$95
6.	FOB-L	Extended BASIC	$95
7.		Structured BASIC	$295
8.	FDA-L	Z80 Macro Assembler	$95
9.	FDF-L	FORTRAN IV	$95
10.	FDC-L	COBOL	$95
11.	FDM-L	Multi User BASIC	$800
12.	DEM-L	Data Base Management System	$95
13.	WPS-L	Text Formatting System	$95
14.	RATFOR	(incl. FORTRAN IV)	$195

SHAKESPEARE, BASIC AND THE CIA

Fingerprinting sentence structure

A few miles out of Washington, approaching Langley, Virginia, there is a sign over the highway. It reads 'C.I.A. Turn Right'. Shortly after making that turn a security barrier is encountered, and behind it a chain link fence. Identity documents are painstakingly checked against a list held by the guard, and your physical details verified with a computer housed in a large complex of buildings within the compound. This is the first of a series of increasingly stringent checks that one meets on penetrating the heart of America's Intelligence machinery. And there, some six stories below ground, is a computer that plays with words. Exactly what this computer does, and indeed its very specification, remains a closely guarded secret. For at least part of the time, however, it is engaged in some fascinating literary detective work.

by Julian Allason.



Illustrations by Russell Mills

Literary detection is not a new science. Almost from the moment that Shakespeare was laid to rest, scholars have argued about the authenticity of various passages. In 1850 Spedding postulated that Shakespeare's disputed play, Henry VIII, was actually the result of a collaboration with Fletcher. This year Spedding's thesis was largely vindicated by Thomas Merriam — and a computer.

It was the Cold War, with its ceaseless propaganda battles, that generated the interest of the Intelligence community in computerised linguistics. Forgeries and plants abounded. They needed to know what was authentic — and what was not.

A celebrated case concerned the autobiography of Kim Philby, the Soviet double agent who had reached the top of the British Secret Intelligence Service. After his defection, a book entitled 'My Silent War' appeared, complete with foreword by Graham Greene. Philby claimed it was entirely his own work. SIS, knowing that it was a final attempt to smear them and damage Anglo-American relations, sent a copy to Langley. There, CIA specialists ran comparisons of 'My Silent War' with articles that Philby had written whilst operating as a Foreign Correspondent for the Observer. The tests showed that whole chapters had been written by others. The book is now regarded as highly suspect.

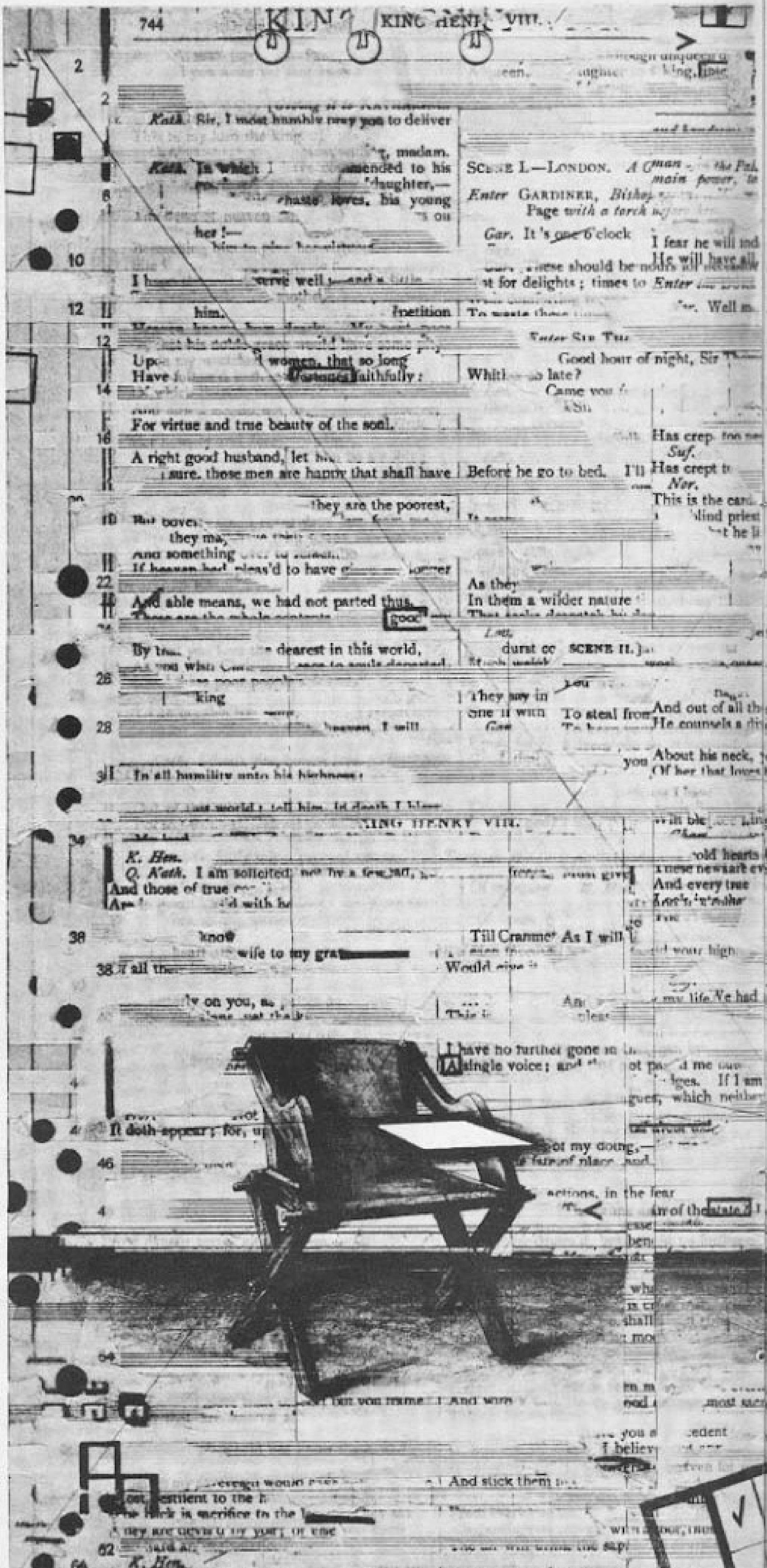
A similar thing happened when the memoirs of the West's top Kremlin agent appeared. Using similar methods, Soviet specialists swiftly "proved" the "Penkovsky Papers" a forgery. Penkovsky was shot and his book remaindered.

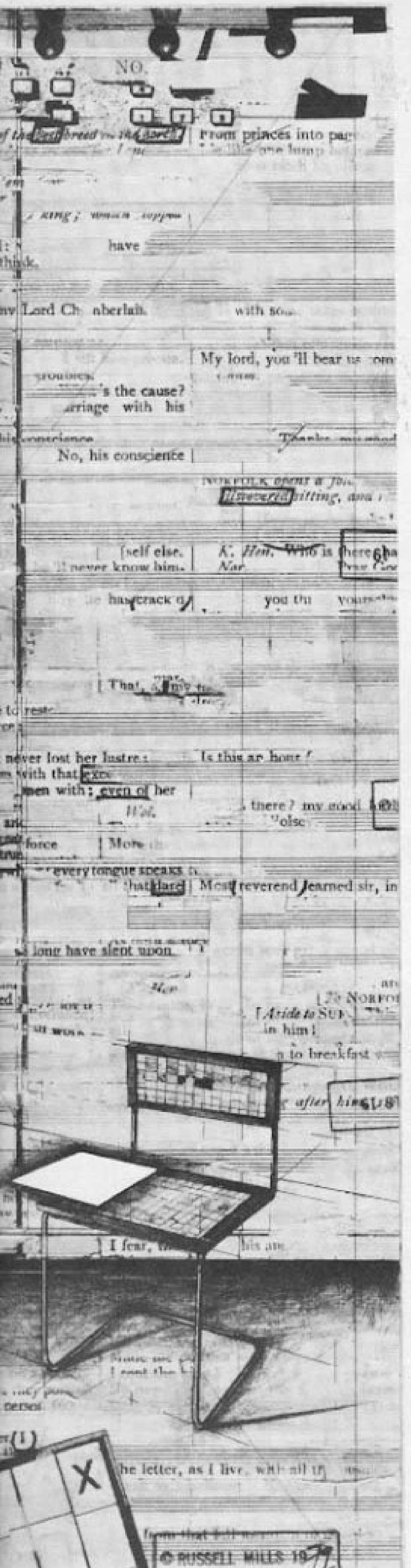
All methods of literary detection involve recognition techniques. The detective uses the computer to help establish an author's sub-conscious habits of speech or writing. Most Shakespearian scholars are capable of assembling a passable pastiche of the Bard's prose. To overcome vulnerability in this area, only very common "filler" words such as "and", or "it is" are tested. This is because use of these words is a matter of subconscious habit. Furthermore, they occur throughout written output whatever the mood or occasion. Position in the sentence is also held to be important.

Surprisingly, our syntactic habits are so ingrained that they show through even when an attempt is made to mimic the literary style of another. Usage of certain "filler" words remains fairly constant throughout a writer's work.

At the simplest level a literary detection program involves a series of string searches on text samples of established authorship. The incidence of certain strings, for example "such a", are noted. A profile of the author's literary style is then constructed. Similar tests are carried out on suspect text, and finally both profiles are compared. It should then be readily apparent whether or not all the samples were written by the same person.

Although fairly long samples of text are required for a definitive evaluation, it is possible to obtain reasonably accurate results from a short BASIC program. The routine I am working on for Petsoft uses less than 8K. The following simplified example illustrates a string search for the word "and".





```
100 DATA"HANDSOME ANDREW AND HIS WIFE"   : REM Text Sample
110 READ T$                              : REM Read Sample
120 FOR C=1 TO LEN(T$)                   : REM Set counter to no. of
                                           characters in string
130 IF MID$(T$,C,5)="  AND  " THEN S=S+1  : REM Tests next five characters
                                           (including spaces)
140 NEXT C                               : REM Increments Character Counter
150 PRINT " 'AND'APPEARED"S"TIMES"
160 PRINT"IN A STRING OF"LEN(T$)"CHARACTERS"
170 PRINT"ITS INCIDENCE WAS" S/(LEN[T$])*100 "%"
```

Note that five spaces are allowed for the string "AND" to avoid acceptance of "HANDSOME", "ANDREW" etc. An additional statement would be required to accept "AND" as the first word in a string.

In practice an expanded algorithm tests a much longer sample of text for the incidence and position of a number of such "filler" words and phrases.

At about the same time as the C.I.A. was trying to catch up on computerised literary detection they faced another problem. They needed English translations of all the scientific and technical information being published abroad. Their linguists could not keep up. Computers, it was argued, could provide the answer.

Early efforts at machine translation met with little success. The problem was the inadequacy of available syntactical analysis. Linguistics, the scientific study of language, was still in its infancy.

But in 1957, Professor Chomsky of the Massachussetts Institute of Technology, published a book called 'Syntactic Structures'. In it, he argued the existence of underlying or Deep Structures beneath the surface structure of the sentence. These defined and inter-related all the factors determining structural interpretation.

It is fair to say that not all linguisticians accept Chomsky's thesis. But it has given the machine translation and literary detection specialists a good deal to think about.

Computerised linguistics is now finding a much wider, and academically more respectable range of applications. In 1974 Dr. Andrew Morton created a legal precedent with his evidence that only 7 of 11 police statements submitted in a case had been written by the defendant. The result was an acquittal.

In a recent book *(Literary Detection, Bowker, $22.00)* Dr. Morton examined the difference between Jane Austin and The Other Lady, who in 1965 completed the novel which had lain unfinished since Jane's death. Although in literary terms the imitation is quite good. Morton demonstrated that the probability of Jane Austin having penned the 4,000 words that were written by The Other Lady to be more than one thousand million against (see chart).

With the continuing evolution of linguistics and the rapid pace of micro-processor development, it is reasonable to project not only considerably more accurate machine translation than my pocket Craig translator offers, but the prospect of an infallible literary detective.

Post script: Having run this article through my PET, the computer confirms that it is almost certainly not written by Shakespeare. . .

A comparison of Jane Austen and The Other Lady

	Occurrences of the Habit in					
Habit	*Sense and Sensibility*	*Emma*	*Sandition* (Jane Austen)	*Sandition* (The Other Lady)	Chi squared (a)	(b)
an	25	26	11	29		
a + an	172	212	112	112	1.40	12.85
a	147	186	101	83		
P.B. *such*	14	16	8	2	0.20	3.92
and	253	299	151	154		
F.B.*I.*	12	14	12	1	2.45	6.84
the	270	271	229	221		
P.B. *on*	11	6	8	17	1.58	8.45
F.W.S.	22	26	19	8	0.43	6.34
this	32	39	15	15		
this + that	126	144	52	37	0.25	3.64
with	59	74	28	43		
with + without	77	84	38	47	5.02	3.71
very	37	68	26	27		
P.B. *the*	4	2	3	7	—	12.7

Notes: 1. The samples are: *Sense and Sensibility* — Chapters 1, 3, *Emma* — Chapters 1,2,3. *Sandition*, Jane Austen — Chapters 1,6. *Sandition*, The Other Lady — Chapters 12,24.
2. The figures for chi squared are for the comparison of the three genuine samples, (a), and then for the comparison of these samples taken together for the comparison with The Other Lady, (b).

Cottage Computers

a division of Embryonic Systems Pty. Ltd.

We are the southern agents for MicroByte

MicroByte

Software for the 2650

ACOS

BINBUG 4.4 (VDU)

BINBUG 5.2 (Terminal)

ASSEMBLER

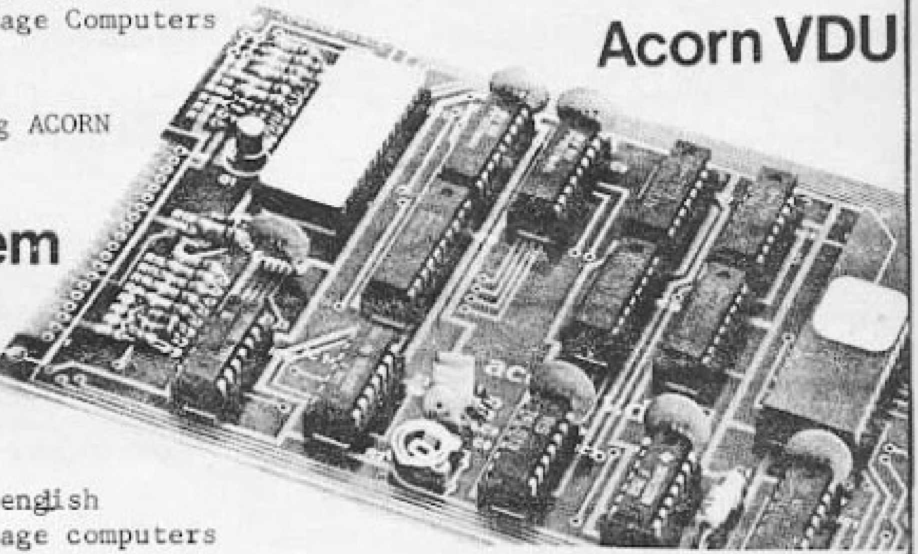
BASIC

SOURCE GENERATOR

This fully tested, well documented software will turn your 2650 based system into a powerful ,flexible microcomputer........
WE STOCK ALL ITEMS HERE at Cottage Computers

Full Product Info Sheets available

Acorn VDU



HERE IN AUSTRALIA!!!!The Amazing ACORN

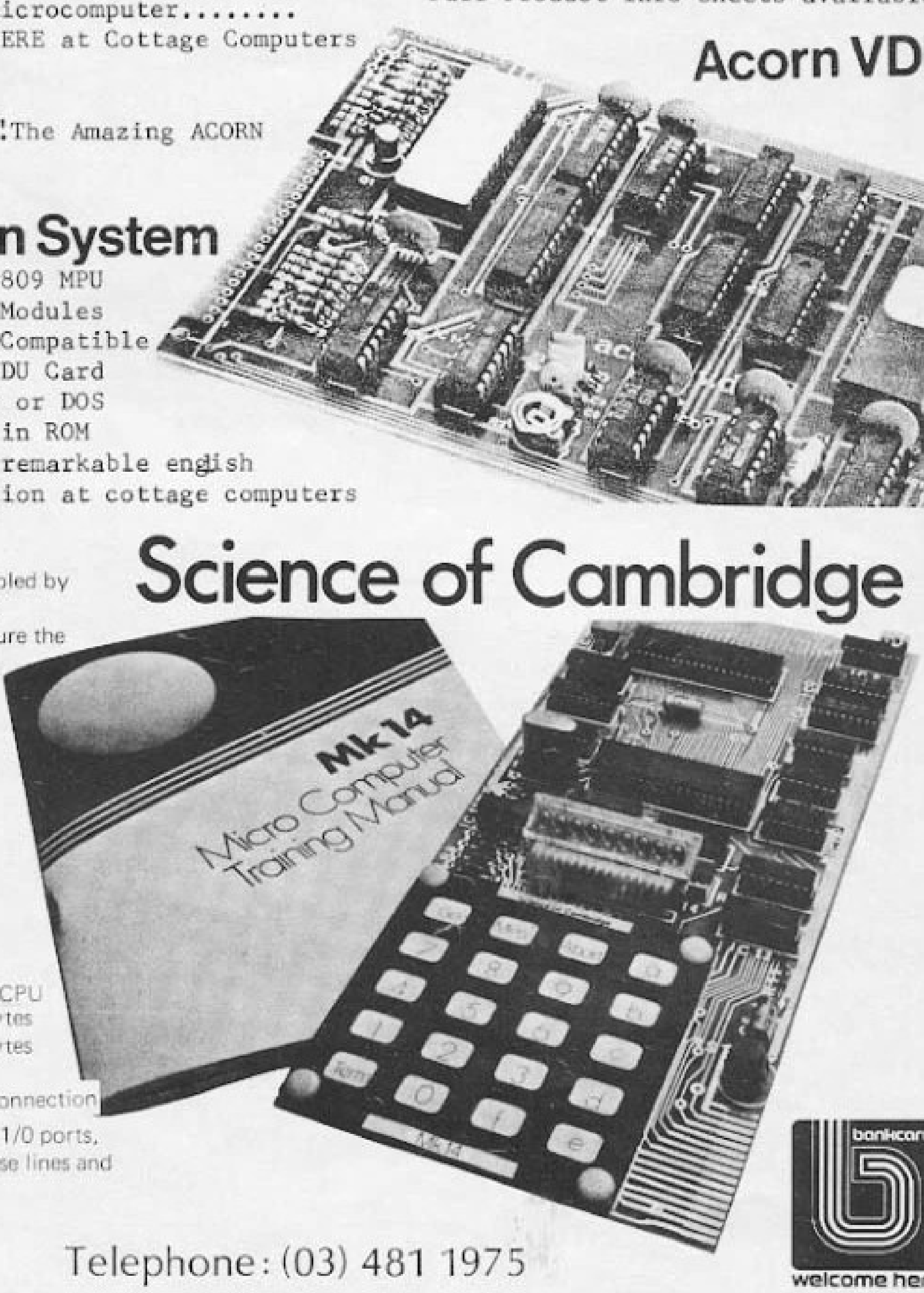
Acorn System

6502 or 6809 MPU
EUROCARD Modules
TELETEXT Compatible
Colour VDU Card
ACORN COS or DOS
4K BASIC in ROM

Come and see this remarkable english computer in action at cottage computers

Science of Cambridge

The MK14 is a kit of parts assembled by the user to form a minimum cost computer. It comprises in miniature the essential elements common to all computer systems.



**MK 14 Kit $93
**Cassette I/face $14
**VDU kit $69
**PROM Prog.$24

Extra Programs available soon

• Central Processor Unit	SCMP CPU
• Fixed Memory	512 bytes
• Read/Write Memory	256 bytes
• Users Terminal	
• External Input/Output	Edge connection for 16 1/0 ports, 8 data bus, flags and sense lines and supply

bankcard welcome here

Telephone: (03) 481 1975

386 Queens Parade, FITZROY NORTH, Victoria 3068

BUSINESS COMPUTING II

By Rodnay Zaks

The Mailing List

The mailing list is often neglected in a business system. This is because most business systems provide a complete record of sales (the sales list). In such a sales record, the name of the customer, as well as all information relevant to the transaction, has been entered. At first sight, it might therefore look as if the system has all the information available that it needs in order to list customers for a promotional action. Technically this is correct. However, in practice, such a sales list is not usable as a mailing list for immediate use.

For short mailings, the sales list can indeed be processed by a specialised program which will generate the names and addresses and print them. For long lists (such as several thousand names) direct processing of the sales list is completely impractical. First, it would be slow, as the information in the sales list is block structured and only a few entries must be retrieved from every block. Second, the information in the sales list will normally be bulky, so that relatively few names will reside on a single physical medium such as a diskette. Processing the sales list will therefore require significant time and frequent manual changing of diskettes. In short, it is generally unacceptable.

For efficient mail list management, it is also imperative that the user codes every transaction at the time it is performed. For example, the user might code the type of business, or the type of individual involved in the transaction. He might code the nature of the items purchased or the range of the sale. In this way, a compact code can be used which can be utilised as a selection criterion within the list. An efficient mailing list is created by processing the customer's list or else the sales list periodically. It must be kept sorted, either alphabetically, or by post code or by a special user code. In this way, whenever a new name is entered into the list, possible duplication can be checked immediately and efficiently. Even so, a typical mailing list, reduced to the name, address, and code, will contain a large number of characters for each entry. Let us look at it: it will contain the first name, middle initial, surname, title, company name, division or mail stop, street or PO Box, city, state, post code. In addition it will contain a 5 to 20 digit code for efficient retrieval of names. Result: perhaps 100 characters per entry. Let us assume that 80 characters are sufficient for our purposes. A typical block on a diskette contains 256 characters. In our example such a block would contain only 3 names. A typical diskette will provide about 1200 such blocks, and will therefore store about 3600 names.

This is not sufficient for most mailing lists. An efficient mailing list program should be capable of handling anywhere from 1,000 to 10,000 or 50,000 names. Clearly, such a system would be very cumbersome to use, as diskettes would have to be changed manually frequently until the complete list is processed. This may not be acceptable.

As a result of this constraint, microcomputers, restricted by their limited amount of disc memory, impose the necessity of a specially structured mailing list file. An index block is created, which contains only a few bytes per entry. It contains the customer name or code, and a customer pointer, usually a number. In this way it can be restricted to 20 to 30 characters maximum. A complete diskette will be typically allocated to this mailing list index. Whenever a selection is to be performed, only the index block diskette needs to be mounted initially. As an example, the mailing list selection is for all companies in the distribution field, which have ordered more than $200 worth of parts in the last three months. The selection program will derive all its information from the index file and will generate a list of customer numbers meeting the specified criteria. This list can then be used by a simple mailing list printing program which will print the addresses from the customers or sales file, as selected by the list of customer numbers. Whenever the selection has been performed, the program will start generating messages such as "please load diskette number 2-5". This diskette contains the first block of entries that this mailing list processor wants to print. A number of diskettes will still have to be successively mounted on the system, if the mailing list is long, in view of the limitations inherent to each diskette. However, the complete selection, which can be a lengthy process, has been performed efficiently in a very short amount of time using only the index block diskette. A good system will write on a diskette the names it will print and will signal the user on the CRT display that it wishes to have another specific diskette mounted next. The user will have the time to mount the next diskette while the system is busy printing labels. By the time the printer, which is a slow device, has finished printing the last name of the previous diskette, the next one will already have been mounted. The system will perform at the maximum mechanical speed possible.

The drawback of having to feed successive diskettes to a system can be eliminated by purchasing much larger hard discs or dual drive double density diskette drives. New low cost fixed head discs are beginning to become available.

Existing Software Facilities

Most microcomputer systems available today offer the hardware capabilities required to perform all of the operations that have been described at sufficient speed, provided no complex arithmetic is required. Unfortunately, no complete business programs have yet been developed, which completely automate all the tasks required. It can be seen easily from the above description that the task of the file management system and of the various processing programs is complex. Such programs have been created only for the larger computers.

The cost of developing such programs is much larger than the cost of developing the actual hardware on which it resides. For this reason, manufacturers are usually not anxious to develop such expensive programs when they are in the business of selling hardware. Consulting firms and software houses have traditionally had a good business selling specialised programs or packages to business users.

Most microcomputer systems available today will offer a few business programs. Typically they will offer a general purpose file system, which allows the user to create and manipulate files symbolically. This is totally insufficient. Specialised processing programs must exist, which allow the business user to gather data automatically, conveniently, to update it, and to have the computer perform automatically the necessary updates in other files. Most programs available solve only one of the problems. Usually, there is an accounts receivable program and a separate accounts payable, and a separate inventory program, and so on. These separate "packages" are useful to maintain independent files. Provided that the number of transactions is high for any one of these files, they provide a valuable service. However, they do only part of the job. The business user is still required to update manually all of the files that might be involved in a single transaction.

Micros in Business

Now that the limitations have been pointed out, is this a reason not to consider a microcomputer system in a business application? No.

The facilities that have been described are the ideal facilities for a business automation system. However, even solutions far from this ideal will provide benefits far in excess of the cost of the system. A useful system can be purchased for as little as 5 to 10,000 dollars. Such a system with minimal software packages will permit the automation of the inventory, the accounts payable, the accounts receivable, and often other functions such as mailing lists, back orders, payroll or tax computations. Such benefits by themselves usually far outweigh the initial cost of the system. For this reason microcomputer systems are a very valuable tool for limited business automation. However it should be kept in mind that an invisible price will be paid. If extensive files are created, or if specific programs are written, the so-called "software investment" will become dominant. After a period of time, the investment in structuring these files or developing new programs will become more significant than the initial purchase price of the system. After a few years, limitations will be felt and there will be a desire to move to a larger, more complete system. Naturally there is a possibility that the manufacturer of the original system will have expanded its resources. Typically this is unlikely in the long run. In view of the constantly decreasing price of hardware, it is likely that, within a few years, the hardware itself will be obsolete, and that the manufacturer will be marketing a new system, which will not be completely compatible with the first one. The business user will then have to move to the new system and reformat his files and redevelop new programs.

However, it is the author's feeling that even if the first system is essentially abandoned after a few years, it will provide an extremely valuable transition into the true computerised business management world. It is correct that the initial hardware investment will have been lost after a few years, and that a substantial software investment might also be lost. However, this will have been translated in a structuring of business procedures, computerised operation, the training of personnel, and an awareness by management of the additional capabilities which will be needed in future systems. Such a system will provide immediate initial benefits that will typically far outweigh its cost, and it will provide substantial educational benefits to its purchaser. After using such a system for one or two years, the intelligent purchaser who understands the nature of his needs, will be in a position to thoroughly understand the ideal system for his business needs. At this point, his second choice is likely to be an optimal one.

The comparison is analogous to the recommendation that a new driver should first drive a low cost car before purchasing a car of his dreams. The first machine may be inadequate, but provides educational value by expanding the skills of its user.

Customer Business Programs

Once the specific needs of a business have been identified, it becomes possible to evaluate whether a specific microcomputer system will meet its needs. If it does not, additional software programs or packages should be obtained. The essential question at this point is: is it worth writing a program or contracting outside to add specific programs required by the business? The usual answer is: no. Clearly, if the programming capabilities are available in house, or if the educational value of programming is significant to you, then it might be worth writing your own programs. The important point is that most business users simply thoroughly underestimate the amount of effort needed for a usable correct, and documented business program. It is indeed possible to develop quickly in a high level language such as BASIC a program which will appear to meet the main needs. However, unless such a program is well developed, well documented, well adapted to the needs of the business, and to the existing system, it might bring more harm than good. It might be found that after a certain number of items have been entered into a file management program, it becomes essentially useless, because its efficiency breaks down thoroughly, or simply because it is incapable of managing it. It might be found that when some types of data are used or entered, incomprehensible failures occur, which nobody in the company is able to correct. Creating a complete, debugged, truly usable system is typically a long and expensive task. Unless the benefits are clearly justified by the high development costs of such a program, it is usually best to stay away from any tailorised development. Programs which have already been in existence for some time and tested on other users are the most desirable ones to obtain. In addition, a system should be integrated, from a software standpoint. All programs should be capable of handling common file structures, and updating the required files. Additional programs simply added on a system might not be able to make use of its facilities or might create structures which other programs will not be able to use.

However, packaged programs might not be optimal for your specific business. In this case you will have to evaluate a trade-off:

- packaged programs can be modified or adapted, preferably by an in-house programmer. This will result in a better efficiency of the programs for your specific business procedures or requirements.
- customising, or adding programs will increase the overall cost of the system, and may detract from its reliability.

Summary

Most microcomputer systems are available today with simple business packages. These programs are designed to automate transactions. Any business with a large number of repetitive transactions will accrue benefits from computer automation.

However, this is true only if the type of "standard package" available with the system fits the business. For example, a sales entry program might ask so many questions as to require twice as long as manual processing (this is usual). In such a case, the automation is worthwhile only if additional benefits accrue. Such additional benefits are: data entry discipline enforcements, automatic processing or updating of other files, report generation.

If these additional programs or benefits exist, then even a somewhat unoptimal, cumbersome, "standardized" data entry program will yield benefits.

This is always the underlying assumption . Warning: if these additional facilities do not exist yet on the system you are considering you should evaluate trade-offs carefully. Many small businesses find manual order processing or manual bookkeeping more practical, and economical than a "standard" business package.

If a part-time programmer is available, or can be justified, many special purpose programs can be developed that exactly fit the business. If the programmer is good, all potential benefits of a computer system will accrue, resulting in a number of benefits, which have been described above:
– efficiency, lower cost, increased reliability, instantaneous management information, improved collecting of receivables, automation of new business procedures (automatic reorder, credit check, specialised mailings).

Microcomputers available today offer all the hardware resources required for less than $15,000. Software is the key.

This is an extract from "An introduction to Personal and Business Computing", published by Sybex of 313 rue Lecorbe, 75015 – Paris, France.

SUBSCRIPTIONS

I would like to subscribe to Australian Personal Computer for one year (12 issues)

beginning with the month of ....

My Name ....
(Block letters please)

My Company
(if applicable) ....

My Address ....

....

Date ....

Signature ....

☐ Australia $24.00

☐ Elsewhere A$30.00

☐ I enclose my cheque, made payable to Australian Personal Computer for $ ....

☐ Please invoice my company

Please send this entire order form, together with your remittance to Australian Personal Computer, Subscriptions Dept., P.O. Box 115, Carlton, 3053, Australia.

AUSTRALIAN
Personal Computer

APC is the pre-eminent publication in the field of microcomputers. Whatever your area of interest in microcomputing – be it business, home, educational – or simply as a leisure pursuit – APC is your passport to the future. Subscribe today, and sleep soundly at night, safe in the knowledge that your personal copy of APC will come thudding through your letterbox every month.

ADE

THE PROFESSIONAL'S CHOICE

COMPUCOLOR II

MICROLINE 80

SPECIFICATIONS
- Integral Mini Disk Drive Standard
- Color Graphics on 128 x 128 grid
- Extended Disk Basic
- Eight colours (foreground and background)
- Up to 32 K/B user memory
- 8080 assembler available

APPLICATIONS
Education: Programmed learning
Small accounting or business systems
Process or control functions
A multitude of games available.

SPECIFICATIONS
- 80 cps 9 x 7 dot matrix
- Program selectable fonts
- Graphics
- 80-132 column printing
- 200,000,000 ch. head life
- full 96 ch ASC II Set

DEALER ENQUIRIES WELCOME

ANDERSON DIGITAL EQUIPMENT PTY. LTD.

THE VIABLE ALTERNATIVE

P.O. Box 322, MT WAVERLEY, VIC. AUST. 3149 Phone (03) 543 2077, P.O. Box 341, Thornleigh, N.S.W. AUST. 2120 Phone (02) 848 8533. Adelaide: 79 9211. Perth: 325 5722. Hobart: 34 4522. Brisbane: 350 2611. Darwin: 81 5760. Canberra: 58 1811. Newcastle: 69 1625. Albury/Wodonga: (062) 2761. Barnawartha 129. N.Z. Wellington: 64 4585. Auckland: 87 6570. Christchurch: 79 6210. New Guinea Lae: 42 3924.

COMPUTER ANSWERS

Every month in APC, Sheridan Williams will assist readers with their hardware, software and systems difficulties. Some questions he will deal with himself, other enquiries will be directed towards members of his consultancy panel.

Fortran Only

I can program in BASIC but only have access to a computer that uses Fortran. Can you foresee any problems that I may encounter when writing data processing programs in this language?
D. Simpson,

I find it hard to believe that there is "only Fortran" on any computer system. Anyway, that said the language has some excellent input and output formatting procedures so you should have no problem reading in data and designing the layout of your output documents, provided that you master the FORMAT statement. Where you may have difficulty is in the actual handling of text, during the processing. There are differences between versions of Fortran so I'll stick to what is general and common to all the variations. You may use either REAL or INTEGER variables for storing strings but each variable can only hold 8 characters; by using an array it's possible to hold strings of any length — e.g. if the array variable TEXT(3) has been declared then you'll be able to hold 8x3=24 characters in it. By using the statement WRITE(n,m)TEXT and the associated FORMAT (1X,3A8) you'll get any text re-output. (Note that there is a difference between 3A8 and A24.) The next problem will be the comparison and maybe the sorting and swapping of these strings. These must be done using the Fortran utilities COMP, COPY, ICOMP and not by the usual methods. For example if variable A contains the word FRED and variable B contains the word BERT you will probably get an error if you try and compare them using IF(A.EQ.B); the reason will only be apparent if you are aware of the way in which text is stored. You'll have to compare them with COMP (K,A,1,B,1). Your Fortran manual should give all the details that you'll need when using these utilities.

The next problems will occur in filing. If you have extended Fortran this supports direct access files, whereas ordinary Fortran needs extra utilities. Look up the Direct Access Backing store package if you need it. If, however, you're not using direct access files this will not be relevant.

I'm afraid that's all that I can say in what must of necessity be a very brief reply; needless to say, write directly to me if you require any more details.
SW

Seeking To Justify

A very simple question which probably has no simple answer . . . how do I right-hand justify numbers that are printed in BASIC?
M. Carlyle.

One of BASIC's many failings is the formatting of output. Perhaps the language's most frustrating feature is that all numbers are printed in left-justified form; almost certainly this is not what we require. There are BASICs that have the instruction PRINT USING. This allows fairly comprehensive formatting, but unless you are able to buy this version for your machine (Research Machine's 380Z will accept it for example) you are stuck with the problem. Incidentally if you have a PET printer this allows formatting of printout, not the screen display.

I've seen several solutions to this problem, all of which use a subroutine to do the formatting for you. I've devised the following solution which should prove instructive to those of you who have never seen this technique used before:

```
DEF FNA(X)=1—LEN(STR$
(X))
```

To implement this function and produce an output justified in the Yth position use PRINT TAB(Y+FNA(X));X

Probably of more use is a function that will allow you to align the decimal points; indeed it may also introduce a technique that's new to some of you:

```
DEF FNA(X)=-LEN(STR$(INT
(X)))-ABS(X)(1)+(X=0)
```

Implementing this function is the same — to align the decimal points in the Yth position use PRINT TAB(Y+FNA(X));X.

This latter function may require tailoring because some versions of BASIC implement logic in different ways to others. The function will work unaltered if you get the answer −1 (minus one) to the statement PRINT (2=2). If you get the answer 1 (one) then you will have to use this function:

```
DEF FNA(X)=—LEN(STR$(I
NT(X)))+(ABS(X)(1)—(X=0)
```

I'm sure that you will get many hours of pleasure (frustration?) out of untangling this function so I'll not explain how or why it works. If you do have any problems you're welcome to write to me direct (enclosing a stamped envelope, please).
SW

High or low level Chess?

When is it better to write programs in a high rather than a low-level language? I'm asking because I'd like to create my own chess program and I'm not sure whether to use BASIC or not.

First I'll start by giving you a definition. A high-level language is one where each source code statement is translated into many (sometimes hundreds) of machine code instructions.

Programming in a high-level language takes far less time than writing the same thing in a low level (assembly) language. Nowadays, cost of labour is one of the predominant factors in the production of a working computer system and, in general, the shorter the programming time the cheaper the program will be. However, a program written in a high level language will translate less efficiently, sometimes taking 50% more space and half as much time again to run than the machine code equivalent. So for programs where speed is important, or where the amount of available memory is low, it's better to write in machine code.

An example of an inefficient translation would be the machine code produced when the single BASIC statement LET A=X↑2 is translated; the machine code produced could be immensely long, and up to 50 instructions may be needed to enable the computer to find the log of X, multiply it by 2, and then antilog it. With very little thought this could have been coded in only a few instructions.

If ever you have used PEEK and POKE in a BASIC program, then you have effectively programmed in machine code.

Finally, to answer your question on whether to write chess in BASIC or machine code. If you have a BASIC compiler rather than an interpreter then go ahead in BASIC; the program won't be as efficient, but it will be far quicker to write, and easier to debug. If, however, you have a BASIC interpreter (which most people have), then I'm afraid that you will have to write it in a low level language or it will be so slow as to be almost useless. Chess playing algorithms, are very complicated and require huge amounts of searching; speed is of paramount importance. A BASIC interpreter will work 100-200 times slower than the equivalent machine code! The reason is that instead of translating the whole program into machine code right at the start of the run (as a compiler will do) an interpreter will re-translate every line into machine code as, and when it comes to it.
Sheridan Williams

"Not only does it tell you your weight, but also when your feet need washing."

Stock Discs

I have been told that I should *not* look at cassette based micros for business purposes. I do not see why, as even a C60 cassette should in my estimation be capable of holding more than 50,000 characters. My application is for stock control, and I would be unlikely to have more than 1000 items on file at any one time.

There are many reasons for rejecting cassettes in favour of discs.
1 Even the 5¼ inch mini-floppy discs hold more on each side than the average cassette. I have seen figures from 70K to 350K quoted. This would save you having to change cassettes in order to swap between programs and files.
2 You can hold many (usually up to 40) programs and files on a single disc, and they will be instantly retrievable.
3 Cassettes can generally be read at between 30 and 300 characters per second. Discs can be read at around 10,000 ch/s upwards. To read a 50K file from cassette could take up to 20 minutes, and yet only around 10 seconds from disc.
4 Discs tend to be more reliable. . .this is because of the nature of the Philips cassette format and the cassette drives used. If digital cassettes with full logic control were used, this statement would not be true and also search times would be significantly improved.
5 Discs are a 'direct access' medium, whereas cassettes are 'serial access'. The advantage of direct access is that *any* record on a file is available for immediate use; in order to access the 1000th record from tape the previous 999 must be read and discarded. With disc the read/write head can be moved directly to the relevant track.

When discs are used, program packages may be written as a 'suite' of programs — one program calling the other when required. It is preferable to write many small programs rather than one large one as each can be worked on and developed separately.

Even if discs are used in a serial access mode (and there are many applications suited to serial access), they are considerably faster than cassettes.

I suggest that you follow this by reading more on the subject of files. I have only just brushed the surface on one absorbing aspect of programming.
Sheridan Williams

Pet protection

I have had a lot of trouble with the cassette tapes on my PET 'losing' data. I use the tapes to hold lists of names and addresses, and I have to keep re-typing them. This is very time consuming, can you suggest a (cheap) cure?

The first obvious suggestion I can make is — have you read the PET manual; this gives a number of rules for using the tapes which help minimise the problems you are getting. As tape dropout occurs with many people I will summarise them here.
1 Use only C30 cassettes, the mylar tape backing is thicker, and runs past the heads better
2 Clean the heads frequently by, for example, using a conventional cassette cleaner; the recommended frequency is every five hours of use.
3 Don't leave cassettes in the halted position with the play button down for long periods of time. This causes the rubber drive capstan to become temporarily dented, thus leading to frequent errors.
4 Where possible use the verify command to check the tape is OK when initially written.

As two further suggestions of my own, don't try to cram too much data onto the one cassette — split your data over a number of tapes. Although it may take time to load and unload during a program run, it's much better than having to re-type all your data. Don't store your cassettes near a magnetic source (VDUs, televisions etc) as the magnetic flux can erase the information on the tapes. An advantage of using multiple tapes is that, should a tape dropout occur, then only one of your tapes need be re-written.

The real solution to your problem (unfortunately it's not cheap) would be to buy a second cassette recorder for your PET and "backup" (keep double or triple copies of your data). It is very unlikely that two tapes containing data would both go wrong at the same time.
Jon R. Malone

COMPUTERWARE for PROGRAMS

PET Disk

Job Costing for Engineers
Data Storage & Retrieval
Debtors
Creditors & General Ledger
Inventory
Word Processor
Assembler
. . . . prices on application.

TRS-80 Disk

Terminal Control.	$36.00
Line Printer	$30.00
Check Register	
Accounting System	$59.00
System Doctor	$47.00
Basic Toolkit	$36.00
Analysis Pad.	$59.60
Disk Head Cleaner.	$20.00

APPLE Disk

Typing Tutor	$30.20
Cookbook	$30.20
Mailing List	$32.00
Graphic Demo Disk . . . . for Trendcom 200	free with purchase of Trendcom 200
Disk Head Cleaner	$20.00

COMPUTERWARE
63 PAISLEY STREET, FOOTSCRAY VICTORIA 3011
TELEPHONE: 68 4200

ESP:PSYCHIC PERCEPTION ~OR DECEPTION?

TYRONE CRUDIS PEERS INTO THE UNKNOWN

Extra-Sensory Perception, abbreviated ESP, is the presumed ability to become aware of events external to one-self without the use of the known senses. Included in this definition are *precognition* or the knowledge of events before they happen; *mental telepathy* or the knowledge of the thoughts of another; and *clairvoyance* or the ability to perceive events or objects at a distance. Whether such abilities exist in actuality is still a subject for heated discussion among scientific investigators today, as it has been for many decades.

Some scientists discount any phenomena that do not occur regularly under conditions which are known and can be reproduced ("laboratory conditions"), but there is a whole alternate side of reality encompassing phenomena which are evidenced only by uncontrolled incidents appearing spontaneously. Examples are mutation in plants and animals, the effects of impact of subatomic particles and automobiles, fireballs, love at first sight, and, perhaps, ESP. This article gives the background for, and examines a BASIC program which will test an individual's ability to predict a coming event by ESP, and will evaluate performance in terms of the deviation from what could have been expected by chance.

George Snow

Testing problems

In tests of precognition, telepathy, and clairvoyance involving two persons, an *experimenter* and a *subject*, it is difficult to isolate one such ability from the others. For example, if an experimenter is shuffling and dealing cards while a subject in another room successfully calls out what he believes each card to be, how can we tell whether the subject is using precognition to predict the order of the cards, telepathy to read from the experimenter's mind the name of the card dealt, or clairvoyance to perceive the card through the intervening wall? Clearly the existence of the human experimenter causes the problem, and if he could be replaced by a machine capable of truly random shuffling and unimpeachable recording of the order of a deck of cards and of the guesses, telepathy at least could be ruled out. If the actual deck could then be replaced by an invisible symbolic deck, not shuffled or cut until the subject's choice was recorded, a good argument could be made that clairvoyance was also prevented, leaving pure precognition as the only possible mechanism to explain successful tests.

Prior to the advent of computers many complex electromechanical devices were developed to solve this problem and, incidentally, to prevent unconscious errors or conscious cheating on the part of the experimenter or subject from affecting the results. Your micro-computer offers you an ideal way to run such tests, isolating precognition from all other potential results. The accompanying short program, due to F. Chambers of Co. Mayo, Ireland, is designed to test precognition in this way. While it uses PET graphics to good effect, these can be easily modified to suit other machines' requirements.

Scoring ability

In order to evaluate performance it is necessary to know not only how many right guesses were made but also how much this number deviates from the number which might have been achieved by chance.

Common sense tells us that it is equally probable that a fairly-tossed coin will fall heads or tails. The statisticians say that the probability of heads is one in two or 0.5, the probability of tails is also 0.5, and the probability of either one or the other occurring is 1, or a certainty. On the other hand, experience tells us that when we flip a coin a number of times we do not always get heads half the time. The mathematician, Jacob Bernoulli, (1654-1705) is said to have cogitatted about this for twenty years before he came up with Bernoulli's Theorem: "If the probability of an event's occurrence on a single trial is P and if a number of trials are made independently and under the same conditions, the most probable proportion, X/N, of the event's occurrences, X, to the total number of trials, N, is also P; furthermore, the probability that X/N will differ from P by less than a given amount, however small, increases as N increases."(1)

Although this statement appears simple, it is a nest of subtleties and traps and can easily lead us into fallacious inferences such as the so-called "Law of Averages" which many gamblers insist guarantees that a run of bad luck must be followed by a run of good luck "to even things up". Not so! In infinite time the heads will equal the tails, but the pattern of events by which this is achieved cannot be predicted.

Binomial distribution

In the deck of cards used for ESP tests the familair suits and values are replaced by five symbols for a total of twenty-five cards. If the deck is shuffled and cut and a random guess made as to the value of the card thus exposed, the odds are five in twenty-five or 0.2 that the guess will be correct due to chance alone. Such a situation where there is one "right" category and one "wrong" category, is called by statisticians a *binomical distribution*. In a binomial distribution the probability that X out of N selections will be right by chance is:

$$\frac{N!}{X!\,(N-X)!}\,P^X Q^{(N-X)}$$

where N! (read "N factorial") is 1x2x3...x(N-2)x(N-1)xN. One can observe that the fractional term on the left is the number of combinations of N objects taken X at a time. This is also known as the *binomial coefficient* for N and X, and tables of its values can be found in many handbooks. On the other hand our computers can be put to work generating them. P is the probability of being right by chance, which is 0.2 in the ESP card guessing case, and Q is 1-P, the probability of being wrong, or 0.8. If this formula is worked out for each value of X from 1 to N, the result is a binomial distribution, a stepped, bell-shaped curve, shown in Fig. 1 for N=100. The plateaux of the curve represent the probability of guessing exactly X

cards correctly by chance. The value of X corresponding to the peak of the curve is PxN, the probability of a single trial times the number of trials; it is called the *mean*. A parameter called the *standard deviation*, symbolized by the Greek letter sigma, which represents the extent of scatter of points within the curve, has the value $(NxPxQ)^{1/2}$ or $0.4xN^{1/2}$ for this particular case. As the peak of the curve is made lower, and the skirts more extended, the standard deviation becomes greater. *Variance*, an alternate term, is simply the square of the standard deviation. It is interesting to note from Fig. 1 that the probability of getting less than 7 or more than 35 cards right out of 100 is so small as to be invisible on the graph; no less than $2x10^{-4}$. The probability of getting half the cards right is $1.6x10^{-11}$.

Appraising results

The standard deviation of the binomial distribution has been used since about 1885 as a basis for measuring the amount of deviation from chance of actual results in card-guessing experiments.(2) The quotient of the actual number of correct guesses less the expected chance number divided by the standard deviation is called the *critical ratio*. It is a measure of the unlikelihood of getting the results by chance:

$$CR=\frac{X-PN}{(NPQ)^{1/2}}$$

A ratio of zero indicates that chance alone is at work. A ratio greater or less than zero signifies that some factor other than chance is affecting the results. Whether this factor is ESP depends on how well the experiment is designed to exclude all other possible explanations. The larger the ratio, the greater the significance. A critical ratio of 2.7 sigma, which has a probability of only about 0.0035 occurring by chance, and corresponds to 31 guesses right in 100, is often used as the minimum acceptable criterion for non-chance phenomena. Critical ratios in the twenties and thirties and above for individuals are reported in the literature of professionally-supervised tests for ESP. If you can achieve 3 sigma results on 100-trial runs you are definitely material for the Society for Psychical Research.(3)

Random number generation

We now understand what parameters the computer should record and compute to appraise the results of an ESP test. We have not, however, discussed how the computer is to "shuffle" and "cut" the ESP deck to ensure that the input to the subject is determined purely by chance. Here, too, there are unexpected subtleties which must be dealt with. You may be in for a rude awakening if you thought your computer could "generate" random numbers.

In BASIC the command RND(X) produces a decimal fraction between 0 and 1. Generally in micro-computers X is a dummy argument; that is, the value of X does not affect the numbers generated. In some dialects of BASIC the argument is omitted and the function is simply written as RND. One or another algorithm is used which is known to generate a non-repeating series of decimal fractions, and each time RND(X) is called for, the next term in the series will be produced. Such a system is known as a *pseudo-*

random number generator. If you are not sure about how your computer responds to RND or RND(X), make a series of tests. I will wager that, *starting from power on each time*, the following program will give the same set of numbers for any positive value of X:

```
10 FOR J=1 TO 10
20 PRINT RND(X)
30 NEXT J
```

If it doesn't, write to the editor and your inputs may make a useful collation for a survey article. In this connection, the Commodore PET User's Handbook implies that a different set of numbers is generated for each different X. This is not the case on PETs I have tested. However, it is true for PET that if X is given zero or negative values, fixed numbers are generated; that is:
RND(0) always gives .564705882
RND(-1) always gives 2.99196472E-08, etc.

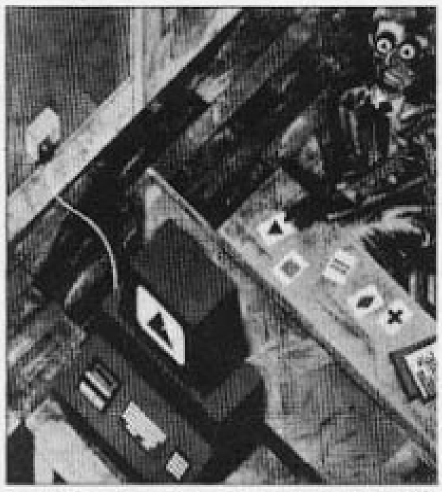
How truly non-repetitive is *your* RND algorithm? *My* answer for PET is that it is non-repetitive enough for all practical purposes, which is to say that inanity set in after running 25,000 terms and finding that none of them was the same as the first term. You can try this with:

```
10 X=RND(1):PRINT "X="X
20 Y=RND(1):N=N+1:
PRINT N;Y
30 IF Y=X THEN STOP
40 GOTO 20
```

. . . or something similar in your own dialect.

How random is your RND algorithm? Here's a well-known little demonstration, sometimes done with darts and a board, which should give you confidence in your computer's randomicity (if it merits it): Fig. 2 shows a square enclosing the quadrant of a circle of unit radius. The ratio of the area of the square to the area of the quadrant is $4/\pi$ as can be determined by observation.

The following program generates random points in X, Y within the square. It tests to see if they are within the quadrant, and sums



them in K if they are. The step number and the result to date are continuously updated until the final step. Punch this in and let it run for a few days. You may be surprised by the results.

```
80 REM: 'πGENERATOR'
90 PRINT"[CS] ":K=0:FOR J=1
TO 5000000:PI=4*K/J
100 X=RND(1):Y=RND(1):
IF X↑2+Y↑2)1 THEN 120
110 K=K+1
120 PRINT"[HO] J=" JTAB
(22)"π="PI
130 NEXT J:PRINT"[CS] π="PI
```

I note with humility that the legislature in the sovereign state of Kentucky in the United States of America once planned to pass a law making π equal to 22/7. An even better approximation, incidentally, is 355/113.

Randomizing the start

Clearly we can't afford to have our ESP Test program always generate the same sequence of symbols from power on! Like Caesar's wife, the test must be above suspicion in this department. One good way to ensure this is to randomize the point of entry into the table of random numbers. If your computer has a real-time clock, a simple technique applicable also to games is to use the clock time to determine how many successive generations of numbers are to be ignored before the first number is used. This need only be done once in a series of tests. For example, the PET has among other clocks a dedicated variable named TI which counts the number of jiffies (1/60 secs.) since power on up to 5184000, or 24 hours. At the point in the program where the first random number is required one can instruct the computer:

```
10 FOR J=1 TO TI
20 X=RND(1)
30 NEXT J
```

. . . or the running time can be held to a certain maximum by substituting VAL(RIGHT$(STR$(TI),3)) or TI-1000*INT(TI/1000) or a similar expression for TI. Another good way is to let the random number generator run continuously until the subject is through reading the instructions. For example the program might read:

```
10 PRINT"PRESS ANY KEY
WHEN READY."
20 X=RND(1)
30 GET A$:IF A$=""GO TO 20
```

This is the technique incorporated in the accompanying program.

Single digit random numbers

Having achieved the eminence of Caesar's wife, random-wise, you may now ask what we are to do with these random nine-digit numbers between zero and one, since what is required is random one-digit numbers from one to five inclusive. Those who know the answer may skip to the next paragraph. The answer is: one can generate integers from 1 to A with the expression:

INT(A*RND(X)+1)

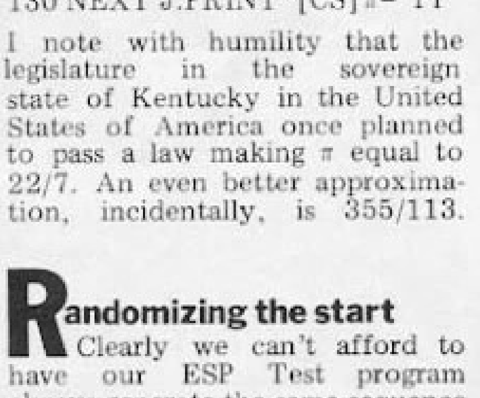
A*RND(X) will change the range from between zero and one to between zero and five. In most micros, INT(Y) will generate the greatest integer not greater than Y, changing the range of our expression to the integers from zero to four. The addition of one, either within or without the parentheses, will shift the range to the integers from one to five. (There is one intentional error in this article. Readers who spot it and who approach the editor on or about the birthday of the dog Laika, (and who can correctly identify the dog Laika) will be treated to a drink — in all Laika-lihood).

Progam comments

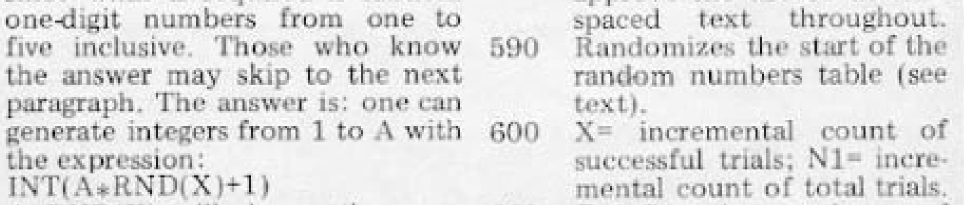
With all these concepts firmly in mind we can now confidently analyse the appended program, "ESP TEST", submitted by F. Chambers. An earlier version is available on cassette from Commodore's library.

LINE	COMMENT
100	Good practice to give clear statement of title, author, and date, with room for addition of notes on future modifications by others without renumbering. 400 lines is a bit more than enough, however! Good practice not to use line numbers below 10, as single digits can sometimes be entered in error in the "ready" state by novice operators, thus wiping out lines having those numbers.
500	Instructions: I am glad to see the title printed as part of the instructions; I also approve the use of double-spaced text throughout.
590	Randomizes the start of the random numbers table (see text).
600	X= incremental count of successful trials; N1= incremental count of total trials.
630	F=1 flags the sample run of symbols.
645	Good practice.
670	N= number of trials.
690	SS= symbol selected by sub-

The π maker



Probability of guessing ESP cards correctly by chance in 100 trials



ject; S= numerical equivalent.

800 Generates random numbers from 1 to 5; R= symbol selected by computer.

910 See Table 1 for PET program listing conventions. These graphics can be redesigned for computers that do not have PET's facilities in this department. The author has not copied the original ESP graphics faithfully for technical reasons. They included a circle and parallel, wavy lines.

1500 Routine to display sample graphics. Delay using the TI clock.

1570 Routine completed, flag reset.

1600 Increment trials count.

1610 If guess was correct, increment success count. Print response.

1640 End of run.

1710 Rounds off percentage to three figures (five places because of decimal point and the blank which precedes all numerals. Same affect could have been achieved with INT(X/N*1000)/1000.

1740 Equivalent to (X-Mean)/Std. Dev.

1800 Some may not care for this kind of touch or for line 675, but I find them refreshing.

PET Program Listing Conventions (4)

[ ]:	any text in brackets is to be interpreted as instructions to print cursor, clear, home, space, or reverse symbols.
CU:	cursor up.
CD:	cursor down.
CL:	cursor left.
CR:	cursor right.
CS:	clear screen.
HO:	home.
RE:	reverse.
RO:	reverse off.
SH:	shift (hold shift down for next symbol outside of brackets.
SL:	shift lock (hold shift down for all symbols outside of brackets until advised otherwise or end of line.
SR:	shift release (cancels SL).
SP:	space.

Example: "[5CU 3CR]" : print 5 cursor up symbols followed by 3 cursor rights. Note that it is not necessary to specify that these require the use of the shift key.

References

1: This simplified statement of Bernoulli's Theorem is due to James R. Newman and is taken from "The World of Mathematics" vol.3 p. 1448: Simon & Schuster: New York 1956.

2: "Extra-Sensory Perception After Sixty Years": Rhine et al: Henry Holt: New York 1940. Devotes one chapter and an appendix to the mathematics of proper statistical analysis of ESP tests.

3: The Society for Psychical Research: 1 Adam and Eve Mews, London W8

4: Derived from suggestions by J. Collins and other sources: readers' reactions will be appreciated.

Program listing

```
100 REM 'ESP TEST' BY F.CHAMBERS              REV.5/79.
500 PRINT"SPC(15)"ESP TEST
505 PRINTSPC(14)"
510 PRINT"THIS PROGRAM TESTS ONE VARIETY OF
520 PRINT"EXTRA-SENSORY PERCEPTION: THE ABILITY TO PREDICT FUTURE EVENTS.
530 PRINT"I WILL SELECT ONE OF FIVE SYMBOLS
540 PRINT"RANDOMLY BUT WILL NOT DISCLOSE IT UNTIL
550 PRINT"YOU HAVE RECORDED YOUR PREDICTION.
560 PRINT"THE FIVE SYMBOLS ARE DIAMOND, PARALLELS,"
570 PRINT"CROSS, TRIANGLE, AND SQUARE.
575 PRINTSPC(14)"
580 PRINT"         PRESS ANY KEY TO TURN PAGE.
590 R=RND(1) GET K$ IFK$="" GOTO590
600 X=0 N1=0
610 PRINT"DO YOU WISH TO SEE THE SYMBOLS BEFORE
620 PRINT"STARTING? PRESS 'Y' FOR YES, 'N' FOR NO.
630 INPUT Y$ IF Y$="Y" THEN F=1 GOTO1500
640 IF Y$="N" THEN 650
645 PRINT" WRONG KEY  TRY AGAIN."  GOTO 630
650 PRINT"HOW MANY TRIALS DO YOU WISH TO MAKE? I
660 PRINT"SUGGEST A MINIMUM OF 25 FOR STATISTICAL SIGNIFICANCE.  ";
670 INPUT N IFN<=0THEN650
675 IF N>100 THEN PRINT"I HAVE THE TIME IF YOU DO!
680 INPUT"OK. ENTER YOUR FIRST SYMBOL SELECTION   D,P,C,T,OR S)";S$ GOTO700
690 INPUT"ENTER YOUR NEXT SELECTION (D,P,T,C,OR S)";S$
700 IF S$="D" THEN S=1 GOTO 800
710 IF S$="P" THEN S=2 GOTO 800
720 IF S$="C" THEN S=3 GOTO 800
730 IF S$="T" THEN S=4 GOTO 800
740 IF S$="S" THEN S=5 GOTO 800
750 PRINT"WRONG KEY  TRY AGAIN."  GOTO 690
800 R=INT(RND(5)*5)+1
820 IF R=1 GOTO 900
830 IF R=2 GOTO 1000
840 IF R=3 GOTO 1100
850 IF R=4 GOTO 1200
860 IF R=5 GOTO 1300
900 REM DIAMOND GRAPHICS.
910 PRINT"J"SPC(19)"
920 PRINTSPC(18)"
930 PRINTSPC(17)"
940 PRINTSPC(16)"
950 PRINTSPC(15)"
960 PRINTSPC(15)"
965 PRINTSPC(16)"
970 PRINTSPC(17)"
975 PRINTSPC(18)"
980 PRINTSPC(19)"
985 IF F=1 THEN 1540
990 GOTO 1600
1000 REM PARALLEL GRAPHICS.
1010 PRINT"J"SPC(15)"
1020 PRINTSPC(15)"
1030 IF F=1 THEN 1540
1040 GOTO 1600
1100 REM  CROSS GRAPHICS.
1110 PRINT"J"SPC(19)"        "
1120 PRINTSPC(19)"  "
1130 PRINTSPC(19)"  "
1140 PRINTSPC(15)"        "
1150 PRINTSPC(15)"        "
1160 PRINTSPC(19)"  "
1170 PRINTSPC(19)"  "
1180 PRINTSPC(19)"  "
1190 IF F=1 THEN 1540
1195 GOTO 1600
1200 REM TRIANGLE GRAPHICS.
1210 PRINT"J"SPC(19)"
1220 PRINTSPC(18)"
1230 PRINTSPC(17)"
1240 PRINTSPC(16)"
1250 PRINTSPC(15)"
1260 PRINTSPC(14)"
1270 PRINTSPC(14)"
1280 IF F=1 THEN 1540
1290 GOTO1600
1300 REM SQUARE GRAPHICS.
1310 PRINT"J"SPC(16)"
1320 FOR W=1TO6 PRINTSPC(16)"          "
1330 NEXT W
1340 PRINTSPC(16)"
1350 IF F=1 THEN 1540
1360 GOTO 1600
1500 REM PRINT SAMPLE GRAPHICS.
1510 FOR K=1 TO 5
1520 R=K
1530 T=TI  GOTO 820
1540 IF TI<>T+120 THEN 1540
1550 T=TI
1560 NEXT K
1570 F=0  GOTO 650
1600 N1=N1+1
1610 IF S=R THEN X=X+1  PRINT"RIGHT!";  GOTO 1630
1620 PRINT"WRONG. ";
1630 PRINT X"CORRECT OUT OF"N1"TRIALS."
1640 IF N1=N GOTO1700
1650 GOTO690
1700 REM END-OF-GAME ROUTINES
1704 PRINTSPC(14)"
1705 PRINT"      PRESS ANY KEY TO SEE RESULTS.
1706 GET K$ IF K$="" GOTO1706
1710 PRINT"YOUR PREDICTIONS WERE"LEFT$(STR$(X/N*100),5)"% CORRECT.
1720 PRINT"CHANCE RESULTS WOULD GIVE 20% CORRECT.
1730 PRINT"YOUR PREDICTIONS HAD A CRITICAL RATIO OF"
1740 PRINTLEFT$(STR$((5*X-N)/(2*SQR(N))),3)" STANDARD DEVIATIONS."
1750 PRINT"A FIGURE GREATER THAN 2.7 S.D. WOULD BE"
1760 PRINT"SIGNIFICANTLY HIGH FOR A RUN OF 25 OR    MORE."
1770 INPUT"ANOTHER RUN? ('Y' OR 'N')";Y$ IF Y$="Y" GOTO600
1780 IF Y$= "N" GOTO 1800
1790 PRINT"WRONG KEY  TRY AGAIN"  GOTO 1770
1800 PRINT"OK. IT'S BEEN A PLEASURE TO COMPUTE FOR"
1810 PRINT"YOU. TRY ME AGAIN SOON!" END
```

Ready for business.

CBM™ Business Computer System.

From the truly bewildering torrent of superlatives, claims, promises and come ons in the computer marketplace, how do you choose a microcomputer that's right for you?

Four things determine the suitability of a microcomputer: Hardware, Software, Service and Price.

At Hanimex we've got it all together with Commodore Business Machines:

☐ All components and equipment are designed and manufactured by Commodore ensuring total System compatability. Every CBM System is tested prior to installation to guarantee maximum reliability from the day you plug it in.

☐ An array of software programmes is available for business and other applications including General Ledger, Creditors, Debtors and Word Processing.
However, if you require additional or specialised programmes these can be produced easily and inexpensively because of the advanced software tools built into CBM Systems.

☐ Hanimex and your Commodore Dealer have a serious commitment to service. Thanks to modular design and Self-Diagnostics, problems can be identified and remedied quickly and complete customer maintenance service is available.

☐ And when you finally get down to the price you will find that the Commodore Business System is more computer for less money. And a computer that will pay for itself faster.

☐ Along with Commodore, we at Hanimex are planning for the future needs of the business and professional worlds. Our main customer is the small businessman, but professional programmers, engineers and others are also finding our products invaluable.

☐ We are proud to offer this high quality product at an incredibly low price and CBM Systems are available for immediate delivery from the nationwide network of Commodore Dealers.

commodore

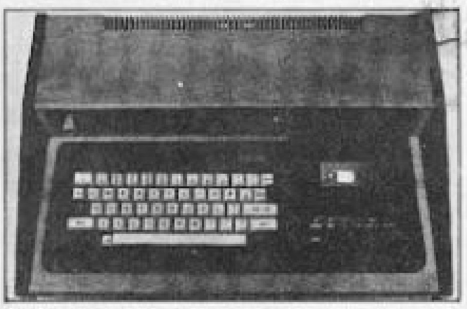
Distributed by
HANIMEX
Commodore Business Machines Division
Ph: (02) 938 0275

HANIS 80

CHECKOUT

SYSTEM 80

APC recently had access to a System 80; Z-80 based, it's fully compatible with the TRS-80 level II. The machine, an integral processor, keyboard and cassette drive, plugs into the domestic TV and is no bad way of "getting into" computing.



It has a socket for attachment of an external cassette, quite useful if you encounter load problems, as I did. It seems that some of the commercially available tapes, while being suitable for the TRS-80, need some means of volume adjustment on the System 80.

We lent the machine to Ian O'Neill (who has used a TRS-80 for some time now) to see what he thought of it. Here are some of his comments:

"The built-in cassette recorder worked well and cut down the number of leads needed to connect the system to the mains and to other units. It also offered a manual/computer control switch, ideal for rewinding etc. All the TRS-80 programs from my own system loaded perfectly, both on the internal cassette and on an externally connected one. I also liked the built-in power supply."

One or two things are worth mentioning in addition to Ian's comments. It has a double width character switch which stretches the characters that appear on the screen, thus making editing much less of a strain on the eyes. There is a reset button tucked away behind the keyboard which resets the machine when and if it locks. This can happen if reading a poorly recorded tape for example. Finally, the S100 bus ensures compatibility with a wide range of peripheral devices. Returning to Ian's comments, he also noticed a few things that he didn't like:

"The monitor I used had a severe attack of the shakes when attached to the '80' and I also failed to get it to work with either of our domestic televisions. It seems that I should have tuned them in, something I didn't realise at the time. Perhaps the instructions could be clarified. I found the convention of calling the return key "new line" repulsive — I don't know why, I even prefer ' enter '. Ah well, it's not that important I suppose. One fairly serious omission from the keyboard was a ' clear ' key; and, despite the assurances of the sales blurb, I could find no justification for the claim of ' full cursor controls '; there is only backspace and new line.

The manuals, although first attempts, appear sufficient and are probably easier on the inexperienced owner than the detailed, though excellent, TRS-80 Level II manual. The literature seems to have been written with the American user in mind and hence it is over-simplified in places and littered with bad grammar and American spelling.

"The BASIC is very compatible with that of the TRS-80 as is shown by the benchmark timings (in seconds), which are as follows:

BM1: 2.7, BM2: 11.6, BM3: 28.0, BM4: 28.5, BM5: 31.3, BM6: 51.9, BM7: 81.0, BM8: 11.7

One final thing — numeric keypad freaks will be sorry to hear that there is no convenient place on the System 80 to fit a keypad, as there is on a TRS-80."

Our view of the system is that it is an economical way of "getting into" personal computing. Its main disadvantage is that if you use the family telly, as I do, then either the family has to prefer watching you playing with the computer or your computing time will be severely curtailed in the interests of domestic harmony. *— David Tebbutt*

TECHNICAL DATA

CPU	Z80
Memory	4K RAM
Screen	Own television or monitor 64 or 32 characters per line
Cassettes	Integral or own domestic connected through DIN connector
Bus	S100
Ports	Up to 256 through expansion box
Languages	TRS-80 Level II compatible

SYSTEMS

SALES LEDGER

The life blood of most businesses is their cash flow. In fact many small businesses go bankrupt because they find it impossible to get their customers — particularly the large ones — to pay their accounts on time. Not surprisingly, we are concentrating the first of our regular software features on the control of a major source of business cash — The Sales Ledger.

OBJECTIVES OF SALES LEDGER

The job of the Sales Ledger is to control and record details of monies owing to a company from the sale of their products or services. If you were to ask the accountant what he would expect to find in his Sales Ledger system he would probably reply something like this:—

"I must be able to post dated invoice or credit note amounts to the account of the customer concerned. Similarly I must be able to post any cash I receive from the customer. I want a free choice in the type of accounting system to be used. If I choose a balance forward system I would only expect to see details of transactions in the current period, but I may wish to have dated balances. If I chose an open item system I'd want reference numbers against each invoice and credit note. I might want to produce remittance advices so that my customers can tell me which invoice they are paying and I'd certainly need to be able to allocate cash paid against invoices. I'd also need to be able to deal with cash I can't allocate which may mean that I need to be able to indicate any invoices which may be in dispute. Whichever accounting system I may choose I would like to be able to change to the other easily; of course I'd accept a compromise in going from balance forward to open item. I might want to have some of my customers on open item and the rest on balance forward. I'd definitely expect customer statements to be produced at the end of each period and I'd probably want an aged debtor analysis to pinpoint my bad paying customers."

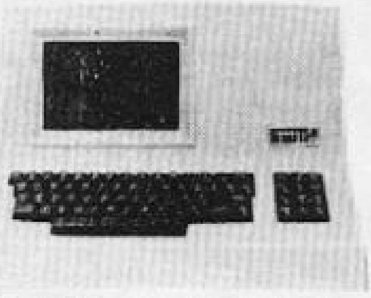
Now although Sales Ledger is primarily the province of the accountant there are other interested parties. The customer for instance may like to see payment terms clearly stated, particularly if he can take advantage of any prompt payments discounts. The Sales manager may wish to see some analysis codes in the system so that he can do reports by Rep. or area. The auditor may wish to see Sales and Cash day book listings to assist him in audit trials. The salesman may wish to enquire at any time on the indebtedness of his customers and he may need to know how near to their credit limit they are.

Finally Sales Ledger packages are rarely implemented on their own so linkages may be important. Are the input transactions produced automatically by the order processing or invoicing system? Are nominal ledger transactions created automatically by the system? Having to re-print information already known by the system can be a time consuming job.

SOUTH WEST TECHNICAL PRODUCT CORPORATION

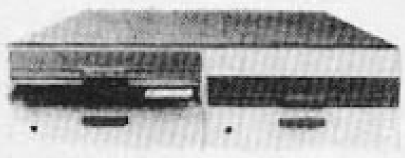
MICRO COMPUTER

SYSTEMS SOLUTIONS



Intelligent Terminal

IF YOU HAVE A BUSINESS OR SCIENTIFIC PROBLEM THAT CAN BE SOLVED BY COMPUTER — WE HAVE THE SYSTEMS SOLUTION, AND AT PROBABLY MUCH LESS THAN YOU WOULD EXPECT TO PAY FOR OTHER SIMILAR SYSTEMS.



Southwest Technical Products Systems give you unmatched POWER, SPEED and VERSATILITY.

- **Two Central Processers with 56K & 128KRAM expandable to 768K using Motorola's latest 6809 micro.**
- **Three Disk Drive types, 175K, 2.4M & 16M Bytes.**
- **Large range of Printers, 80 & 130 Column High Speed DOT MATRIX and DAISY WHEEL.**
- **Powerful and versatile Intelligent Terminal.**

COME AND SEE THE SWTP RANGE AND TALK OVER YOUR SYSTEMS PROBLEM

GFS ELECTRONIC IMPORTS
15 McKEON ROAD,
MITCHAM, VICTORIA, 3132
PH (03) 873 3939 CABLE:
"CQMIMPORTS" MEL.

SYSTEMS

Distributor	IMS	Hanimex	Retail Sciences Dicker	Pitt St.	Comp. Power	Byte Shop	GFS	Comp. Company
TASKS								
Post Invoices	x	x	x	x	x	x	x	x
Post Credit Notes	x	x	x	x	x	x	x	x
Post Cash	x	x	x	x	x	x	x	x
Balance forward system	x	x	x	x	x	x	x	x
Aged balances	x	x	x	x	x	x	x	x
Open item system			x	x	x		x	
Allocate cash	x	x	x	x	x	x	x	x
Unallocated cash	x	x	x	x	x	x	x	x
Mixed systems		x	x		x		x	x
Statements	x	x	x	x	x	x	x	x
Aged debtor report	x	x	x	x	x	x	x	x
Sales tax analysis	x	x	x	x	x	x		x
Sales day book	x		x	x	x			x
Link to Invoicing	x	x	x	x	x	x	x	x
Link to Nominal Ledger	x	x	x	x	x	x	x	x
Link to Stock Control	x				x		x	x
VOLUMES / SIZES								
Max. customers / disc system	1000	1000	600 1200-2000	200	varies	varies	399	900/6000
Max. transactions / customer system	3000 30000	3600	1200 3000	1750	varies	varies	>500	1900/14000
Max balance ($)	1,000,000	9,999,999	10,000,000		varies	varies	99,999	9,999,999
Max. transaction amount ($)	100,000		1,000,000		varies	varies	99,999	999,999

EVALUATIONS

IMS Sales Ledger
This system was written by Integrity Management Services of Melbourne (03-51 9156). It is available from most hardware retailers at around $500.

The price does not include customisation but it is available at a cost related to the modification involved. The minimum requirements of the package are a 48K CP/M system, dual floppy disc and a printer. There are sixty users at present throughout Australia.

Documentation comprises an operators manual and, if the program source is purchased, a design manual. The 70 page operator manual includes a step by step introduction to running the system and sample printouts, while program descriptions and file structures and layouts are contained in the design manual.

The package is written in BASIC and a three months warranty against software bugs is provided. Training is available at an extra charge.

Melbourne Byte Shop
Written in Australia by Davidson Software, this system is available from Melbourne's Byte Shop (03-568 4022). The basic price of the package is $1000. The system is very easy to operate and needs little reference to the comprehensive, user-orientated documentation. Minimum hardware requirements are a 64K CP/M system, 8" dual drive double density floppy disc and printer. The program is written in BASIC. Training and customisation are available at an agreed price, while any bugs that creep in are corrected free of charge.

Dicker Pty. Ltd.
This system was written by Retail Sciences and is distributed in Australia by AJ & JW Dicker (02-524 5639).

The package is an integrated Invoicing, Accounts Receivable, Accounts Payable and General Ledger system and costs $1950.

The cost includes installation and a period of training to be arranged between user and dealer. The minimum hardware requirements are a 48K CP/M system, 500K of on-line disc storage and a 132 column printer. There are over a thousand users at present.
The documentation provided is a three part users' guide which includes a general system overview, system design and capacities, program descriptions, explanation of error messages and suggested operating procedures. The package is written in BASIC, and customisation is done as required at an agreed price.

OTHER SALES LEDGER PACKAGES KNOWN BUT NOT EVALUATED

Pitt Street Microcomputers	02-569 8228
Computer Power Australia	03-645 3333
GFS Electronic Imports	03-873 3939
The Computer Company	
Hanimex	02-938 0400

FASTER, MORE EFFICIENT PROGRAMMING

What can you do to make your programs faster and more efficient? This article will hopefully provide the personal computer programmer with ways of speeding up his programs. Avenues already exist for those with money — companies and firms where ten thousand dollars here and there is easily found. Those with money have several options open to them: they can buy a faster computer, buy a faster interpreter[1], or pay a "software house"[2] to write the program for them. I have, however, yet to come across a program written by an amateur or professional that cannot be made faster.

It is not always a good idea to concentrate solely on making the program faster, because in doing so it may become longer and/or less accurate. Nevertheless there are ways that are well worthwhile implementing which do not make the program longer or less accurate.

Improvements in speed are particularly worthwhile on slower computers. Today's large mainframe computers are so fast that a 20% saving in a response time of one second is unnoticeable; but a similar saving to a micro user could save ten seconds or more.

If you are like me then I think that you will get great personal pleasure from improving your programs — even ones which are supplied or bought can usually be improved.

I have written hundreds of programs and have yet to be entirely satisfied; they could all be improved in one way or another. I have a program that takes 15 seconds to reply in a 3-D noughts and crosses game; it used to take over 30 seconds, but by observing the points outlined later I managed to halve its time of response.

Okay then, let's get started on some ways to speed up programs. These processes may only make the program 10-20% faster, but they could make a staggering improvement. I once wrote a program that would have taken 8600 years to perform some routine. It only required a small modification and the same routine worked in 0.4 seconds — a slight improvement. Here are some ways in which a student of computer science is taught to manage programs more efficiently.

One line Improvements

The statement LET X = A 2 could be made 7 times faster written in this way: LET X = A*A. Powers are much slower than multiplications.
Similarly replacing LET X = A 3 with LET X = A*A*A will give a 6 fold improvement; non-integer and higher powers are not worth altering.
The statement LET X = 2*A could be made 6% faster written in this way: LET X = A + A. Multiplications are slower than additions.

Function Evaluations

Remember that certain operations are faster than others, the usual order for decrease in speed is given below. If a statement can be replaced by one further up the list then usually it results in a faster execution.

Each line takes successively longer than the previous one. The order of these functions may vary with different interpreters and should be checked with the manuals, or by running the benchmark tests mentioned later in the article.

Array Variables

Do not use array variables unless absolutely necessary. Always replace LET A(1) = 3*A*B with LET A1 = 3*A*B unless of course you need the modified address that A(1) can provide. It takes most interpreters 65% longer to find an array variable in store than an ordinary variable.

Another failing is the use of the same array element on successive lines. The left hand example is 10% slower than the right hand example.

```
1Ø LET X = A(P)+3      5 LET T= A(P)
2Ø LET Y = A(P)+4     1Ø LET X= T+3
3Ø LET Z = 6*A(P)     2Ø LET Y= T+4
                      3Ø LET Z= 6*T
```

Successive calls of the same array variable are inefficient. Even though the right hand program is longer, it is faster.

FOR . . . NEXT LOOPS
FOR . . . NEXT loops are the fastest way of performing a loop, so should always be used in preference to the LET. . . IF. . . THEN loop. The former are staggeringly faster, often anywhere between 200% to 2000% faster. Example below

```
1Ø LET A = Ø             2Ø FOR A = 1 To 1ØØØ
2Ø LET A = A + 1         3Ø statements within
                                          loop
3Ø statements within loop  4Ø NEXT A
4Ø If A  1ØØØ THEN 2Ø
```

In addition to the FOR loop being one line shorter, it only takes 3 seconds for execution as compared to 12 seconds for the LET. . IF. . . THEN loop.

Nested FOR loops are a source of slowness too, compare these two programs:—

```
1Ø FOR A=1 TO        1Ø FOR B=1 TO 4
   2Ø
2Ø FOR B=1 TO 4      2Ø FOR A=1 TO 2Ø
3Ø LET X(A,B)=Ø      3Ø LET X(A,B)=Ø
4Ø NEXT B            4Ø NEXT A
5Ø NEXT A            5Ø NEXT B
```

The left hand program will execute 25% slower than the right hand program. Why? In the left method the inner loop is set up 20 times. Setting up means that the computer has to store the start, stop and increment parameters. In the right method the loop has only to be set up 4 times.

If possible, stick to integers as parameters in a FOR loop. (The opposite of integers are reals. Integers are whole numbers that can be stored exactly; reals are numbers that may not store exactly — try converting 2.1 into binary). Most interpreters treat integers in a different way to real variables, and integer arithmetic is around 10% faster than real arithmetic.

Another common fault is evaluating a constant inside the loop, when it could have been evaluated outside it. Example:

```
1Ø FOR A=Ø TO       1Ø LET X=3.145296
   9Ø STEP 1Ø          25/18Ø
2Ø LET X=3.1452     2Ø FOR A=Ø TO 9Ø
   9625/18Ø            STEP 1Ø
3Ø LET B=B +        3Ø LET B=B +
   SIN (A*X)           SIN (A*X)
4Ø NEXT A           4Ø NEXT A
```

The left hand program is poor programming as well as taking 15% longer.

Here is an interesting program, the output achieved has been varied depending on the interpreter used.

```
1Ø LET A = 1
2Ø FOR A = 1 TO 3* A STEP A
3Ø PRINT A;
4Ø NEXT A
```

What happens if we use as parameter for the loop the loop variable itself? Here are two outputs achieved with different interpreters: 1 2 3 and 1 2 4 8 16 32 64 128 256 for ever. If when you run this program you do not get the output 1 2 3 then the following is another way of speeding up your programs:

```
1Ø LET B=6          1Ø LET B=6
2Ø FOR A=B TO       2Ø LET C=3*B
   3*B
STEP B/3
3Ø statement        3Ø LET D=B/3
4Ø NEXT A           4Ø FOR A=B TO C
                    STEP D
                    5Ø statement
                    6Ø NEXT A
```

The right hand program will execute faster, because the loop parameters will only have to be calculated once, at the beginning of the loop only.

Apart from the last point made, all the previous points are universal and all interpreters that I have used benefit from the above methods of speeding up.

Brackets

Brackets make no difference to the speed at which a function is evaluated. LET X=(((A + B))/C) is evaluated at the same speed as LET X = (A + B)/C. So the moral is always use too many brackets rather than too few. An example of too few brackets is given here in finding a solution to the quadratic $ax^2 + bx + c = Ø$.

LET X1=(–B + SQR (B*B – 4*A*C))/2*A would give a wrong answer.
LET X1 = (–B + SQR(B*B – 4*A*C))/(2*A) is correct.

Trigonometric Functions

If you are evaluating more than one of the functions at the same time then it is worth using the following trig identities to evaluate the other:

$\sin^2 x = 1 - \cos^2 x$
$\cos^2 x = 1 - \sin^2 x$
$\tan x = \sin x / \cos x$

In a program if we evaluate sin x we can evaluate cos x and tan x from it:

```
1Ø LET A=SIN(X)       1Ø LET A=SIN(X)
2Ø LET B=             2Ø  LET  B=COS(X)
   SQR(1 – A*A)
3Ø LET C=A/B          3Ø  LET  C=TAB(X)
```

The left-hand program should be faster than the right-hand program.

Looking for the place to speed things up.

The place to look is inside a loop. If the loop is traversed enough times a saving of 10% each time around can make an appreciable difference. If a statement is not inside a loop it is unlikely to be worthwhile altering it.

Testing your Computer/Interpreter's speed

It is possible, indeed it is a very good idea, and only needs to be done once, for the amateur to check and list the speed of execution of various routines and functions. The tests used for measuring are called 'Benchmark tests' and you will find a series of 8 such tests in APC Volume 1, Issue 1, page 14.

Benchmark tests are devised to find the speed and efficiency of a particular computer/interpreter; using them it is possible to time a particular routine, addition say. Here is a method for timing addition, but it can be extended to time multiplication, SQR, SIN, RND etc.

```
1Ø PRINT'S'
2Ø FOR A = 1 TO 1ØØØ
3Ø LET X = 2
4Ø NEXT A
5Ø PRINT 'E'
6Ø END
```

Time this program using a stopwatch from the printing of the 'S' to the printing of the 'E'. Now replace the line 30 LET X = 2 + A and retime the program. The difference in time is due to the 1000 additions performed in line 30, so divide the time difference by 1000 and you have the time for one addition.

Replacing statement 30 with 30 LET X = 2*A or LET X = SQR(X) and you can time various functions. Make a note of the various speeds and you can modify your programs to work faster.

One final word of warning, your efforts may only lead to a 10% saving in run-time but they could accumulate to something more worthwhile.

Glossary

1. An INTERPRETER is a program written in machine language, which translates a source program, written in a high-level language (BASIC, COBOL, ALGOL, FORTRAN) into a machine language program. Remember that computers cannot understand any other language except their own machine language, and all other languages must be translated before they can be executed.
2. SOFTWARE HOUSES are independent companies formed to provide computing services to clients. They offer many services, including program writing and hiring of computer time. They also provide software packages.

Apple Owners

GET HARD COPY OF YOUR HI-RES SCREEN GRAPHICS.

This picture was printed on a TRENDCOM 200 printer using an Apple II computer.

The TRENDCOM 200 also prints 80 columns ASCII – upper and lower case. $590 + tax

Also available from Computerware is the TRENDCOM 100 – a low cost thermal ASCII printer with upper and lower case and forty characters per line . . . . $390 + tax.
We also have plug in and go interface modules for

TRS-80 SORCERER PET & SUPERBOARD II

COMPUTERWARE 63 PAISLEY STREET, FOOTSCRAY, VIC. 3011 Telephone: (03) 68 4200

COMPUTER BOOKS from DEFOREST SOFTWARE

Basic Basic	$11.00
Advanced Basic	$11.00
Basic from the Ground up	$11.00
Basic Work-book	$7.50
Discovering Basic	$8.50
Common Basic Programs	$12.50
Sargon Chess	$19.00
How to Build a computer controlled Robot	$15.00
How to profit from your personal computer	$11.00
An Introduction to Microprocessors 0	$11.75
" " " " 1	$12.50
Z80 Programming for Logic Design	$12.50
Z80 Assembly Language Programming	$13.60
Pay Roll & Cost Accounting* *(Software to match $99)*	$20.00
Account Payable & Receivable* *(Software to match $99)*	$20.00
General Ledger* *(Software to match $99)*	$20.00
6502 Assembly Language Programming	$13.60
The 6800 Microprocessor	$11.00
Basic Microprocessors & the 6800	$16.00
Computer Mathematics	$16.00
Consumers guide to personal computing & Microprocessors	$11.00
Mini Computers, structure & programming	$18.00
Fundamentals & Applications of Digitallogic circuits	$13.00
Small Computer systems Handbook	$11.50
The first book of Microcomputers	$6.50
Computers in Society	$9.50
Computers in Action	$7.00
Standard Dictionary of Computers	$25.00
Programming Programmable calculators	$13.50
110 Basic Computer Programs	$8.00
Cobol with style	$9.50
Basic with style	$8.00
Fortran with style	$9.50
Pascal with style	$8.50

* Due in early July

Programming Proverbs	$9.50
Fortran Fundamentals	$6.50
Fortran 4 Programming	$9.00
Microcomputer System Design	$21.00
Microprocessor Data Manual	$10.00
Microprocessor Basics	$16.00
S-100 Handbook	$10.00
Digital experiments	$11.00
Digital Trouble Shooting	$12.50
Telephone Accessories	$7.50
More telephone accessories	$7.50
Logical Design Using IC's	$24.50
Statistical Pattern recognition	$21.50
Fundamentals of Data Base Systems	$21.00
400 Ideas for Design Vol 2	$18.00
Vol 3	$18.50
Vol 4	$17.50
Integrity & recovery in Computer Systems	$12.50
Management of Information Systems	$11.00
File Structure for On-Line Core Systems	$18.50
Data Management for On-Line Core Systems	$18.50
Digital Signal Analysis	$27.00
Computer Security Risk Analysis	$16.00
Character readers & Pattern recognition	$17.50
Modern Electronics Security Systems	$16.00
Printed Circuit Assembly	$6.50
Basic Mathematics Vol 1	$8.50
Vol 2	$8.50
Mathematics for Electronics	$9.50
Human Communication Handbook Vol 1	$12.00
Vol 2	$12.00
Computer Dynamics	$13.00
Computer Aided Design Techniques	$32.00
Electronic Game Projects	$6.00
* TRS80 Disk and other Mysteries "Pennington"	$25.00

POSTAGE IS INCLUDED AT NO CHARGE, HOWEVER WE SUGGEST REGISTRATION OR CERTIFICATION

26 STATION STREET, NUNAWADING PTY. LTD. VIC. AUSTRALIA Tel: (03) 877 6946, 878 9276.

WORD PROCESSING BY COMMODORE

FULL GLOBAL SYSTEM

The Configuration consists of:–

1 x Commodore 32K Micro-Computer	— Central Processor Unit (CPU) — 32K RAM/16K ROM.
	— Video Display Unit (VDU) — 1000 Characters.
	— Business Keyboard
1 x Commodore Dual Disk Drive	— DOS resident in Unit.
	— Uses single sided, single density 5¼" Floppy Disks.
	— 170K (approx. 60 A4 pages) actual user storage per disk.
1 x Diablo 1355 Hy Feed II	— 55 characters per second.
Daisy Wheel Printer	— Produces letter quality print.
"WordPro III" Word Processing Software	
Installation and Operator Training	

ALL FOR UNDER $10,000 *

In addition, for a small extra cost, a powerful computing system with software including:–
General/Creditors/Debtors Ledgers
Stock Control System
COMBIS — Random Access Package — Business Information System
Price List; Mailing List, and many more.

*ex. Sales Tax

THIS MONTH'S SPECIAL OFFERS:— (Please quote Ref: NNS when ordering).

8K PET Micro-Computer — Only $999 — includes "Space Invaders" Program FREE plus $130 worth of other software.

Memorex 5¼" Soft Sectored Floppy Diskettes — 1 Box (10) — $60.00 (with Sales Tax)
1 – 9 each — $6.50 (with Sales Tax)

9½" x 11" Plain Paper Continuous Stationery — 1 Box (2000 sheets) — $25.00 (with Sales Tax)

THE COMMODORE COMPUTER CENTRE

bankcard
welcome here

MICROCOMP
MICROCOMPUTER SYSTEMS DESIGNERS

B.S. MICROCOMP,
4th floor,
561 Bourke Street,
MELBOURNE, 3000.
Tel: 614 1433 / 614 1551

10 PART PASCAL SERIES

THE COMPLETE PASCAL

BY SUE EISENBACH AND CHRIS SADLER

CHAPTER 2 FUNDAMENTALS: ACTION AND DATA

The British Standards Institute (BSI) has produced a draft definition for a standard PASCAL language which the American National Standard Institute (ANSI) and the International Standards Organisation (ISO) are currently examining. Everyone is working to avoid the sort of situation BASIC finds itself in where a large number of the statements for one machine either won't execute or give total rubbish on another.

Niklaus Wirth, author of PASCAL, was very firm on the idea of standardization, and we shall use his book *PASCAL User Manual and Report* (Kathleen Jensen and Niklaus Wirth, Springer-Verlag) as the ultimate reference work for this series.

However, the team at the Institute for Information Systems at the University of California in San Diego, like most other compiler writers, could not resist 'improving' the language slightly for their version. Since this version is currently the most widely available on personal computers, we shall be pointing out the occasional differences between 'Wirth PASCAL' and 'UCSD PASCAL' as they arise.

In any case, until ISO and ANSI publish their final report, we don't know which of these will be closer to the ultimate PASCAL.

Every language (computer or human) has rules of grammar. However it is very difficult to achieve fluency just from a list of rules; examples bring individual points to life. On the other hand an example cannot illustrate all the possible applications of a new rule. In this series we will present sample programs to illustrate points and then, more formally, provide the rules so that the new constructions can be used in a variety of ways and checked for 'legality'.

Format of a program

A good programming language is one that can achieve an acceptable compromise between the following conflicting goals. First, it should provide the programmer with sufficient flexibility to allow programs to be written in a natural, logical way, and second, the programs should have a highly predictable structure so that the *compiler* (the program that translates the source program into a machine-code object program) can be fast and efficient.

In some languages, each statement must appear on its own line. This is equivalent to saying that the *statement separator* is ⟨CR⟩⟨LF⟩ (i.e. the code transmitted when you press RETURN). This limits the maximum length of a statement to some fixed amount (usually 80 characters); it also makes a program with lots of short statements very stilted and space-wasting. In PASCAL the statement separator is a semi-colon which allows several short statements to be compressed onto one line, or a single statement to overflow onto several lines. Even so, the compiler can still rapidly sort out one statement from another

It has become accepted that every language must include some means of documenting a program within the text itself. This provides the reader with additional explanations beyond the bare lines of essential code. In PASCAL the primary method of documentation comes about through the use of a very flexible naming convention. Every name (or identifier), whether it is the name of the program itself or that of one of the variables or other elements within the program, can consist of an unlimited number of characters. The only restrictions are:

1 that the identifier should not be a *reserved word* i.e. one of the instructions of the language, like WRITE etc.

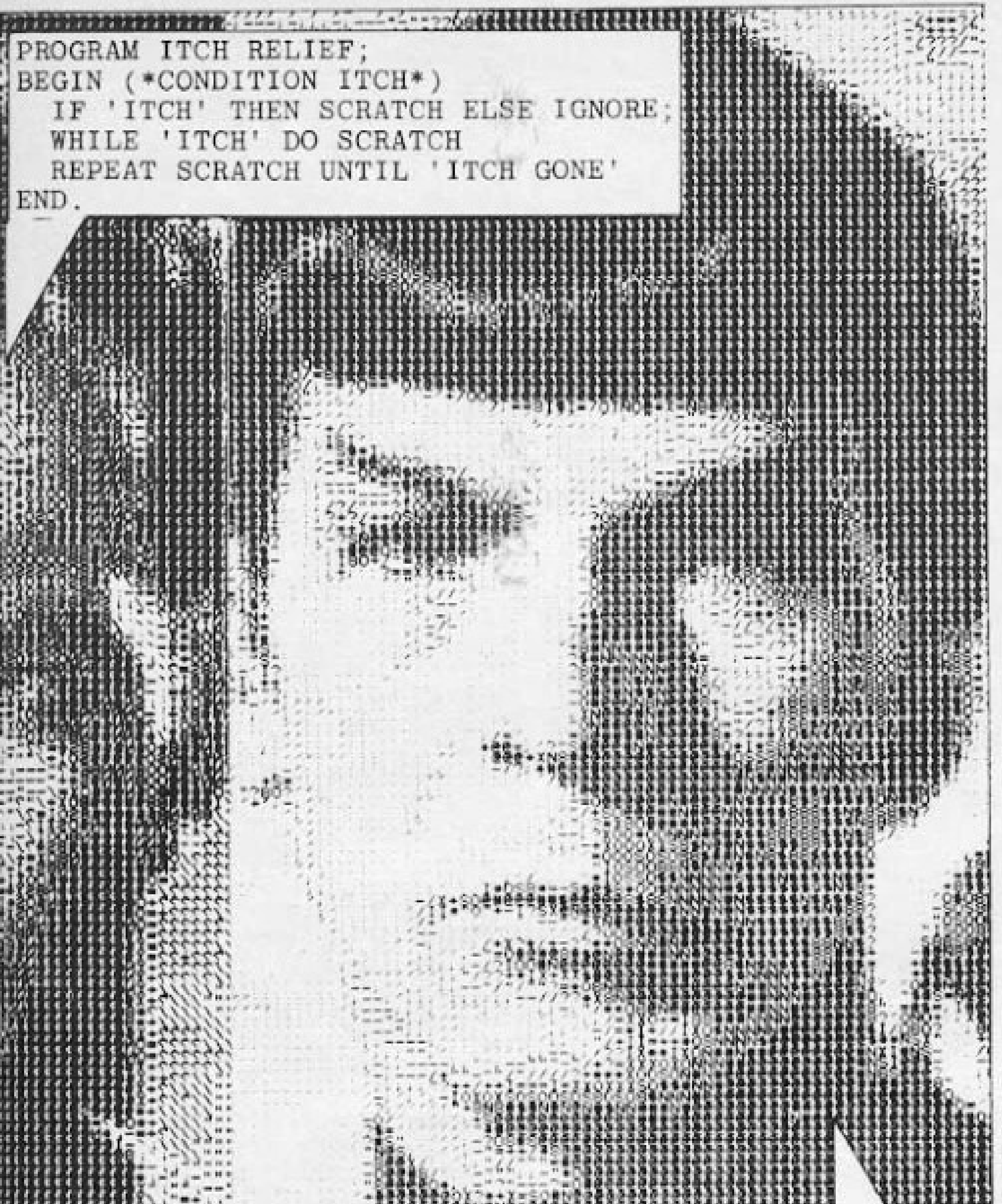
2 that the first character should be a letter followed by an unbroken string of alphanumeric characters.

3 that only the first eight characters are recognized by most compilers. Any additional characters will be there for the benefit of the reader, to explain the functions of the object being named.

Thus line 1 of the program in Box 1 has the form:

```
PROGRAM        EVENINGALL       ;
   ↑               ↑            ↑
reserved word   identifier   separator
```

The identifier "EVENINGALL" gives some idea of what the program is about. The same approach should be adopted when naming variables, strings and all the other



```
PROGRAM ITCH RELIEF;
BEGIN (*CONDITION ITCH*)
  IF 'ITCH' THEN SCRATCH ELSE IGNORE;
  WHILE 'ITCH' DO SCRATCH
  REPEAT SCRATCH UNTIL 'ITCH GONE'
END.
```

Logical, Captain: But was it Wirth it?

program elements, although care should be exercised to ensure that the first eight characters are unique, for the sake of the compiler. For instance it would probably treat identifiers ACCOUNTSPAYABLE and ACCOUNTSRECEIVABLE as being identical.

The second method of documentation is the *comment*. In PASCAL this consists of a string of explanatory text enclosed by the character pairs (* and *) as shown on line 5 in Box 1. (* You can use [and] if you can find them on your keyboard *). When the compiler encounters the left-hand *delimiter*, "(*", it ignores everything until the right hand delimiter, so that the message contained therein is for the human reader only.

Good programmers always use a lot of documentation in their programs, whatever language they are writing in. However, in PASCAL they would probably concentrate their documentary efforts on the various identifiers chosen and use correspondingly fewer comments than they would include in (say) BASIC or FORTRAN, which have more restricted naming conventions.

The program in Box 1 illustrates the general format of a PASCAL program. The box is divided into three sections. Apart from the section containing the PASCAL code there is a section with line numbers down the left hand side.

```
 1| PROGRAM EVENINGALL ;
 2| BEGIN
 3|    WRITE('HELLO') ;
 4|    WRITE('  ', 'HELLO HELLO') ;
 5|    WRITELN ; (*MOVES TO A NEW LINE*)
 6|    WRITELN('AND WHAT DO WE HAVE HERE?')
 7|    WRITELN('HELLO HELLO HELLO')
 8| END.
--------------------------------------------
 9| HELLO HELLO HELLO
10| AND WHAT DO WE HAVE HERE?
11| HELLO HELLO HELLO
```

BOX 1 PROGRAM EVENINGALL

These are *not* a part of the code but are used purely for reference purposes within the text. The section below contains a representation of the sort of dialogue one would expect to see on a VDU or teletype, were this program to be executed.

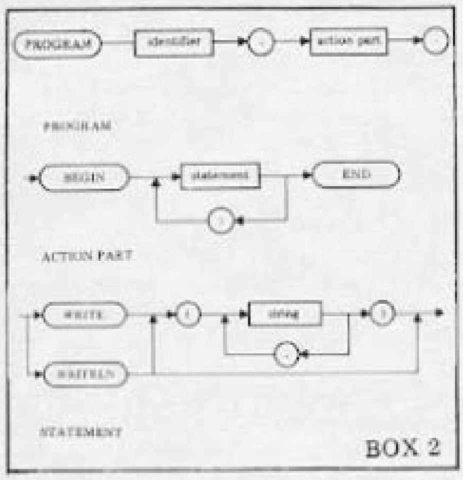
EVENINGALL consists of a program title (line 1), an *action part* (lines 2—8) and a terminator (the full stop on line 8). All programs must close with a full stop as this is a message to the compiler to say that the end of the program has been reached. The actual *executable* part of the program stretches from line 3 to 7 and the results of execution appear on lines 9 to 11. The instruction WRITE causes whatever follows it in brackets to be output. WRITELN will have the same effect except that (CR) (LF) are appended to the end of the text. If there is no text after a WRITELN, the type head is simply moved to the beginning of the next line. Thus lines 3 to 5 in Box 1 have the same effect as line 7 (see lines 9 and 11). The brackets which enclose the output are 'output delimiters', while the single quotes surrounding each item are 'string delimiters'. In line 4 there are two output items separated by a comma.

The structure of the program and various subsections are illustrated, by means of syntax diagrams, in Box 2. These syntax diagrams show what a program looks like, *from the point of view of the compiler*, and as such are worth taking a bit of trouble over. If one knows how the compiler will view a program, then code can always be written which will *compile* even though it may still misbehave when it executes.

CREATIVE COMPUTING

Look at the first diagram in Box 2. When the compiler encounters the word PROGRAM (a reserved word), it looks for a ";". Anything between these is the identifier or program name (provided it obeys the rules). Likewise, everything between the ";" and the "." is the action part which, looking at the second diagram, starts with the reserved word BEGIN and finishes with the reserved word END. Between these are *statements*, separated by ";"'s and they are defined in the succeeding diagram. Check each statement from 3 to 7 in Box 1 against the definition of a statement in Box 2 to ensure that each one is 'legal' — this is exactly what the compiler has to do.

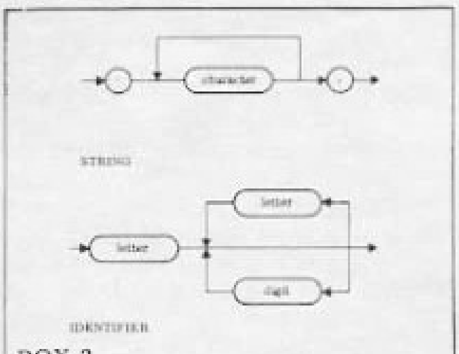
As this series proceeds, the elementary definitions will be expanded and enhanced to include all the PASCAL facilities. In the meantime, below are some rules for interpreting a syntax diagram.
1 Symbols in *circles* are PASCAL



BOX 2

punctuation marks — ie. separators, delimiters and terminators etc.
2 *Sausages* contain either the reserved words (in capitals) or one of 'letter', 'digit' or 'character' — which includes anything on a keyboard.
3 *Rectangles* enclose names of elements which are defined in other diagrams (eg. 'action part' in the first diagram is defined in the second). They can be considered therefore as symbols for other complete diagrams.

In Box 3, two diagrams are presented to complete the set of definitions begun in Box 2 and stated earlier in the text.



BOX 3

Exercise 1: Draw syntax diagrams for a comment.

Programs that do things

Every part of PASCAL, each concept, method and programming trick, has its accompanying syntax diagram which, although perhaps acceptable for compilers, is rather heavy going for the potential PASCAL devotee. At the same time, nobody could be expected to stay satisfied with little Noddy-programs that simulate policemen. Therefore, pausing only to promise that in no future chapter will there be so many (or such complex) syntax diagrams, we proceed to develop more PASCAL features, widening the range of problems with programmable solutions.

Almost every program functions by obtaining some data (input), manipulating or processing this data and presenting its results (output). In PASCAL, this functional aspect of the program (the action part of the previous section) is separated from the more organisational task of deciding how the information is to be stored and used at each stage of the operation. These decisions must be made and announced in a 'declaration part' immediately before the action part is begun.

In program PAY, Box 4, lines 2 & 3 form the declaration part while lines 4 (ie. BEGIN) and onwards constitute the action part. Looking first at the action part, lines 5 & 6, together with line 14, show how a dialogue can be constructed within a program. WRITE outputs text but allows the response to be typed on the same line. READLN requires a ⟨CR⟩ to terminate the input. Finally, line 11 is a typical assignment statement. The values of the variables HOURS, RATE and OVERHOURS are arithmetically manipulated, together with the value 1.5 (from OVER), to produce a numeric value which is assigned to the variable WAGE. The *assignment operator* ":=" is used to emphasize that this activity occurs in the action part, and indicates that the contents of a memory location (referenced by WAGE) is to be altered. The final diagram in Box 6 defines all the new statements introduced in program PAY.

Although the action part of this program must seem straightforward for a BASIC programmer, the declaration part probably looks rather peculiar. At machine-code level, all data is represented by sequences of ones and zeroes at specific locations in memory. Higher level languages must provide a means of accessing and interpreting this data in a more readable form — numbers and characters, arrays and words. Generally, memory locations are accessed by means of variable names (or identifiers) and some languages use restrictions in the naming convention to help the compiler to interpret the data stored at the named location. So FORTRAN distinguishes between names for REALS and those used for INTEGERS, while BASIC has REALS and STRINGS (of characters).

These restrictions are inefficient in two ways — first, program readability is hindered and second, the programmer has to *force* his data into the rigid data types provided. Thus, in most versions of BASIC, a flag (taking values 1 or ∅) which need only occupy one bit will, in fact, occupy 32 bits in the guise of a REAL variable. In PASCAL, variable identifiers have no such restrictions so one function of the declaration part is to give the programmer the opportunity to name variables and state what type they are. So Box 4, line 3 reads:

```
  VAR     HOURS, RATE, OVERHOURS, WAGE:     REAL
   ↑                  ↑                      ↑
reserved word     identifiers              type
```

This enables the compiler to set up all the necesaary memory locations at one go (before starting on the action part). Clearly this is more efficient than the alternative, where memory allocation must occur in conjunction with other compilation activities.

Moving from efficient compilation to efficient execution, another PASCAL feature, the declared *constant*, comes into its own. When a computer executes an arithmetic assignment, all the relevant numbers have to be extracted (via the variable identifiers) from memory. Because this activity uses a significant fraction of the execution time required for the operation, the facility exists to incorporate actual values into an arithmetic statement. Hence PASCAL allows for the declaration of constants (line 2, Box 4). When the program is compiled, every occurence of the specified identifier (OVER) is replaced by the value indicated (1.5). No location in memory is associated with constant identifiers.

Constants can also be used in the more traditional way. For example line 11 in Box 4 could have '1.5' instead of 'OVER'. But if union negotiations managed to push up overtime rates to double-time, someone (imagine a larger

```
 1   PROGRAM PAY ;
 2   CONST OVER=1.5 ;
 3   VAR HOURS, RATE, OVERHOURS, WAGE:REAL ;
 4   BEGIN
 5      WRITE('NORMAL HOURS WORKED? ') ;
 6      READLN(HOURS) ;
 7      WRITE('RATE OF PAY? ') ;
 8      READLN(RATE) ;
 9      WRITE('OVERTIME HOURS— TYPE ∅ IF NONE' ) ;
10      READLN(OVERHOURS) ;
11      WAGE:=HOURS*RATE + OVER*RATE*OVERHOURS ;
12      WRITE('WAGES=' , WAGE, 'DOLLARS')
13   END.
     ------------------------------------------------
14   NORMAL HOURS WORKED? 40
15   RATE OF PAY? 4
16   OVERTIME HOURS— TYPE ∅ IF NONE 3
17   WAGES=178 DOLLARS
```

BOX 4 PROGRAM PAY

```
 1  PROGRAM TEMPERATURECONVERSION ;
 2  CONST FREEZING=32 ;
 3        LINE='                 ---------------------' ;
 4  VAR CENTDEGREE, IFAHRENHEIT : INTEGER ;
 5      RFAHRENHEIT : REAL ;
 6  BEGIN
 7    WRITE('PLEASE TYPE IN A TEMPERATURE IN DEGREES CENTIGRADE:—') ;
 8    READLN(CENTDEGREE) ;
 9    RFAHRENHEIT:=CENTDEGREE*9/5 + FREEZING ;
10    IFAHRENHEIT:=CENTDEGREE*9 DIV 5 + FREEZING ;
11    WRITELN(LINE) ;
12    WRITELN(CENTDEGREE, ' C = ', RFAHRENHEIT, 'F OR APPROX ';
13             IFAHRENHEIT, ' F' ;
14    WRITELN(LINE)
15  END.
-----------------------------------------------------------------
16  PLEASE TYPE IN A TEMPERATURE IN DEGREES CENTIGRADE:—21
17                 ---------------------
18  21 C = 69.8 F OR APPROX 69 F
19                 ---------------------
```

BOX 5 PROGRAM TEMPERATURECONVERSION

program with tax and NI calculations, etc) would have to look through the entire program to adjust it.

The role of the declaration part is therefore to allocate memory locations and to assign constant values for use by the action part. The first three diagrams in Box 6 cover the descriptions of the last few paragraphs. The third diagram in particular shows the exact format of constant and variable declarations. In Box 8 the words 'constant identifier' and 'variable identifier' occur. By these we mean a legal identifier that has been previously declared in a constant or variable declaration.

TEMPERATURECONVERSION in Box 5 contains a wider range of data types. On line 3 there is an example of a string constant. This facility will be familiar to most programmers. In line 4 CENTDEGREE and IFAHRENHEIT are declared as INTEGERS. This means that they can only take whole number values (and in machine terms take up less storage space than REALS).

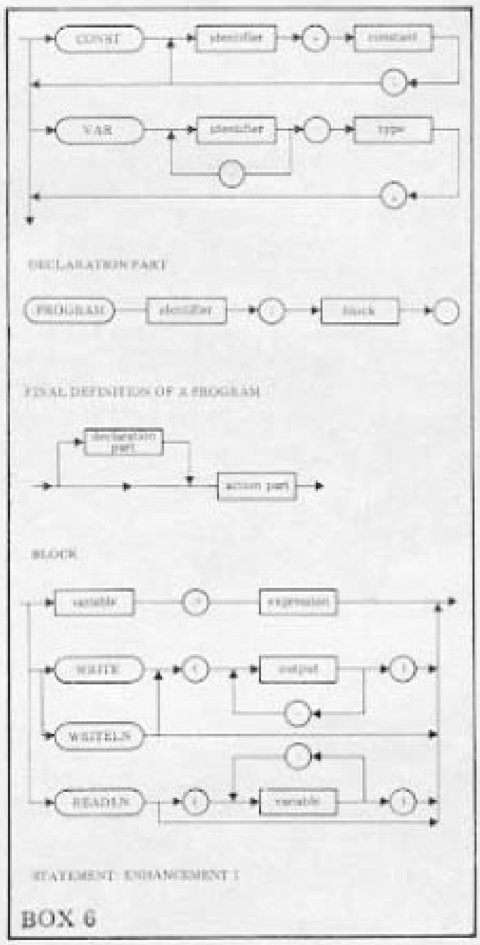
If the left hand side of an assignment statement is a variable of type INTEGER, then all terms on the right hand side must also be INTEGERS. When adding, subtracting and multiplying two integers, the result will always be an integer. However, when dividing two integers the result may be a real. PASCAL provides two division operators. '/' (as in line 9, Box 5) is the division operator for reals and it always produces a real result. It can be used between integers but the result must be assigned to an identifier that has been declared as REAL. The operator DIV (as in line 10 in Box 5) is used between two integers when an integer result is required. Any fractional part is chopped off (that is truncation rather than rounding occurs). So 11/4 gives 2.75 while 11 DIV 4 gives 2.

The syntax diagrams in Box 7 and 8 deal with the fine points of PASCAL language grammar that have been illustrated in programs PAY and TEMPERATURECONVERSION. Unfortunately the syntax diagrams fail to show that reals cannot be assigned to integers explicitly.

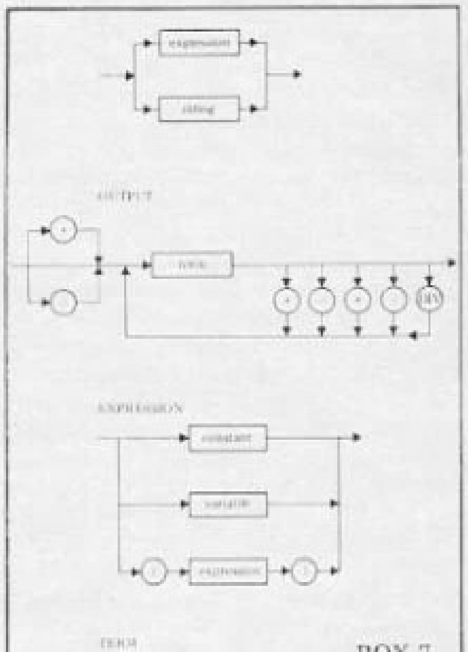
Exercise 2: Write a program called PAYPACKET that (like PAY) asks for hours worked, rate of pay and overtime hours and which displays as well as the wage, the number of five dollar notes and one dollar notes required to make up the wage packet. Use integer rather than real variables and an overtime rate of 2.

Refining a problem

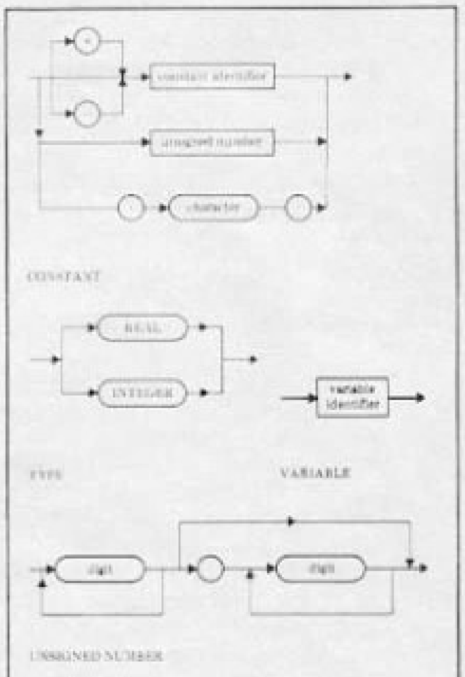
Look at the action part in Box 9. Is it possible that Niklaus Wirth would have incorporated instructions for walking down a VDU screen? It's highly unlikely so the question arises, how do the statements LEFTFOOT and RIGHTFOOT produce the indicated output.



BOX 6



BOX 7



BOX 8

```
 1  PROGRAM WALKING ;

    DECLARATION PART

15  BEGIN
16    LEFTFOOT ; RIGHTFOOT ;
17    LEFTFOOT ; RIGHTFOOT ;
18    LEFTFOOT
19  END.
-----------------------------------------

20  TRAMP
21
22
23               TRAMP
24
25
26  TRAMP
27
28
29               TRAMP
30
31
32  TRAMP
```

BOX 9

PASCAL provides the facility to add new instructions for the duration of an individual program just as it allows for the introduction of variables and constants. These new statements are in fact programs within a program and are called *procedures* (similar to subroutines in BASIC).

Returning to WALKING, what would programs that produced the output required of LEFTFOOT and RIGHTFOOT look like? Box 10 contains programs that produce appropriate output. These programs are conceptually simpler than a single program that outputs TRAMPs down the screen.

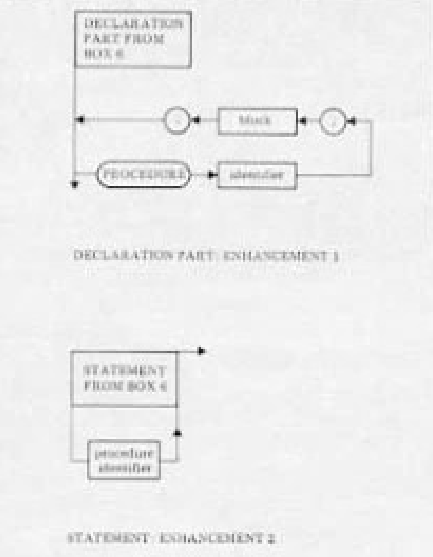
One of the advantages of using procedures is that the production of the main program is straightforward: it will consist of procedure calls (using their descriptive names) whose execution will be as desired and whose details can be considered at another time. The production of the procedures themselves is not difficult because each one should accomplish only one task. (When the tasks get more complicated than WALKING, the number of times they are subdivided — called *stepwise refinement* — is increased. Refinement stops when there is little point in further subdivision.

Looking at lines 2-14 in Box 10 and the first syntax diagram in Box 11, it can be seen that a procedure follows the same format as a program, with just two differences:
1 The title line is PROCEDURE identifier; rather than PROGRAM identifier;
2 There is no full stop at the end of a PROCEDURE since that is the signal to the compiler that the whole program is completed.
Like any other identifier, before using a procedure identifier in the action part of a program it must be declared (see Box 11); that means giving a full listing of the associated code.

Since procedures are very much like programs they too can have declaration parts (see Box 9 line 9). And like whole programs, upon exit from a procedure, the items declared in it become available, because their memory locations are released. Any identifier declared within a procedure is said to be *local* to it. Identifiers in the declaration part of the main program are said to be *global* and can be used within any procedure. The one exception to this rule is that a procedure can only call another procedure if it has been previously defined.

```
 2 | PROCEDURE LEFTFOOT ;
 3 | BEGIN
 4 |   WRITELN('TRAMP') ;
 5 |   WRITELN ;
 6 |   WRITELN
 7 | END ; (*LEFTFOOT*)
 8 | PROCEDURE RIGHTFOOT ;
 9 | CONST SPACE='          ' ;
10 | BEGIN
11 |   WRITELN(SPACE, 'TRAMP') ;
12 |   WRITELN ;
13 |   WRITELN
14 | END ; (*RIGHTFOOT*)
```

BOX 10 PROCEDURES



BOX 11

Exercise 3: Write a program called MARCHING that prints out twice:

```
LEFT

LEFT

LEFT

          RIGHT

LEFT
```

This program should contain three procedures, called LEFT, RIGHT and QUICKSTEP. QUICKSTEP should call the other two procedures.

Conclusion

We have had to cover a lot of ground in this fundamental section. Although future chapters may be concerned with more sophisticated ideas, there will never again be the need to absorb so much new material. Drawing all the various lines together, there emerge three significant points:
1 PASCAL is a language which gives flexibility to the programmer without interfering with the predictable format required for an efficient compiler.
2 The major means of achieving this in PASCAL is through the separation of a program into declaration and action parts.
3 Programmer control over the declaration part offers freedom of specification of both variable names and types as well as giving powerful operational procedures that enable the straightforward coding of (often repetitive) tasks.

Anyone looking over the program examples would be forgiven for wondering — why all the fuss over PASCAL flexibility and structure? It is certainly true that small, simple programming tasks can be solved with fairly similar efficiency in any language. And each problem could have been solved with fewer lines of code, without constants, integers or procedures to produce the given output — even in PASCAL. It's in large programs that these features really come into their own.

In our next chapter we will look at another method for coding repetitive tasks — loops and, as well, we shall expand the range of data types that can be used in a program.

Look up table

Chapter 2: JARGON

Compiler		
Identifier		
Syntax diagrams		— Circles
		— Sausages
		— Rectangles
Punctuation		— Separators
		— Delimiters
		— Terminators
Reserved Words		
Variable	— Type	— Real
		— Integer
	Scope	— Local
		— Global
Constant		
Statement		
Block		
Stepwise Refinement.		

UCSD DEVIATION

In UCSD PASCAL™ the screen is automatically cleared upon completion of execution. To keep the display visible make the last statement before the final END a READLN (which will wait for a ⟨CR⟩).

PASCAL Reserved Words
PROGRAM
BEGIN
END
WRITE
WRITELN
CONST
VAR
REAL
INTEGER
READLN
DIV
PROCEDURE

Exercise Summary
1 Syntax Diagram for comment line
2 Pay program with wages in bank notes
3 Marching

Interrupt is the place in APC where readers can unburden their grievances and air controversial views. New subjects are always welcome; the 'right to reply' shall be wielded at the discretion of the Editor.

Frogs on the hop

In line with United Nations and Western Europe, for Australia, a problem exists — high unemployment and the consequent search for a remedy. There's also a common aim . . . to raise the status of the individual in society.

The theory has been that machines can help to improve levels of unemployment and raise living standards. Now a report coming from France refutes this idea completely. In effect it says (and it is difficult to fault its conclusions) that the spread of computerisation allied to teleprocessing, data banks and so on, will not solve unemployment. In fact, it will increase it and at the same time devaluehuman skills and tighten the grip of an all-powerful state machine over most sections of society. A danger also exists that computerisation could undermine the traditional sovereignity of nations.

This sensational report — L'Information de la Societe — should not be seen as the work of a group of cranks. . . but of a commission set up by the French Government. Headed by M. Simon Nora, Inspector of Finance, the commission, being charged with the task of examining the effects in French Society of computerisation and teleprocessing, points out that in the 1960's computers were expensive, had limited capability and that 80% of France's computer installations were in the hands of 250 firms.

But, says the report, microprocessors have effected an enormous drop in computer prices with a multitude of small, efficient and reasonably priced machines now available on the market. We live, it goes on, in the era of mass data processing. Tending to merge into one multipurpose communications network, the fields of tele-communications and TV channels are now capable of linking computers to data banks and transmitting speech, pictures, sound and data.

Quoting from a summary of the report: "Types of services that until recently were in the realm of science fiction, are coming within our grasp. For instance, newspapers being transmitted directly to our homes, telecopying access to databanks simply by telephoning electronic postal and TV services.

"Used in conjunction with satellites, the marriage of computer and telecommunication technology has produced remote data processing. It has become a dominant means of communication — powerful, universal, soon accessible by low-cost aerials and capable of flooding vast areas. What's more, the computer revolution will upset a number of traditional balances."

This means that the massive productivity that will come with data processing, particularly in services, should lead to a major cut-back in the number of employees. And, according to research estimates this could affect up to 30% of staff in banks and insurance companies in 10 years time. Over a longer period of time, a similar trend will probably cut back employment in social security, post office and secretarial work.

Computerisation of administration in manufacturing, particularly in the car industry, will mean that firms will be able to expand production without increasing — possibly even decreasing — their labour force. And the commission concludes that the French Government's seventh plan's ambition to create 1,300,000 jobs may well be put at risk by the effects of increasing computerisation. What may happen in France can equally take place in other highly industrialised countries.

Then again, what is going to be the effect on the individual of computerisation? Will it create a robotisised sub-proletariat, as human skills are devalued by all-powerful "know-alls" or will teleprocessing, like electricity, flood into all aspects of social life, shaking up the nervous systems of society as a whole.

No attempt to give categorical answers to these questions was contained in the French Government's report, a document which many may regard as being pessimistic. It does, however, contain the evidence on which these questions can be based and does not, in any way, attempt to brush them under the carpet. Moreover, is there likely to be a threat to the traditional sovereignity of states over their telecommunication systems? After all, direct transmission satellites — linked to databanks — will soon be operating regardless of national boundaries. Equally, is there a parallel danger to national security in the field of defence?

M. Simon Nora, and other members of the commission envisage a world in which the treatment of illness would rest with general practitioners, or perhaps even just medical auxiliaries. They would supplant the specialists whose knowledge the computer databank would make available to all. In education, the status of the teacher is seen as being changed profoundly by teaching aids. . . and many other professions and trades would be affected in the same way.

The question is — will civilisation be revolutionised by data processing and the new telecommunication technology and, if so, for better or for worse?
Melville Hawthorn

Micro mania

There is a sequence in the film of 'Hair' when a couple of mounted police, complete with clubs and helmets, come upon a group of hippies dancing in a park. For a moment, it looks as if the police are going to attack — then the horses begin to dance, forwards, backwards and sideways to the music.

I want to draw an analogy between those dancing horses and computers. The film sequence is meaningless. At most, it raises an ironic laugh, because we know the horses wouldn't *really* have danced. We know the police would have dispersed the dancers. But the film, in its facile way, seems to be saying "Hey — you know — if it could only be like that — with a little love — *wow* man!" But it *won't* be like that, it *isn't* like that; and love can't make police horses dance.

We think of personal computers as liberating us in some sense. People talk about the information technology revolution, of computing-power to the people, of the new scope the individual will have to. . . well, to *compute*, at least.

But it isn't like that, it won't be like that; computers can't make police horses dance either. Nor will they make nasty people nice, poor people rich or starving people well-fed.

Yet we say "Hey — you know — if it could only be like that — with everyone programming — *wow* man!"

Computers, whether they act as straightforward tools or as advanced, 'artificially intelligent' decision makers, cannot of themselves change political, social or economic relations in this country. You think they can? With Startrek, Mastermind and noughts-and-crosses? With balance sheets, mortgage packages and, for God's sake, *chicken* recipes? Oh, I see — we haven't really begun yet. Please — when will it start? When Woolworths sell pocket computers with sound, graphics etc.? Can you imagine what these machines will do? How they will be presented and sold? Or will it be when even 'poor' people can afford a computer — so they can see how their shares are doing?

If you point to individual cases where computers have helped, for example, handicapped people, I must still enquire how this is going to be revolutionary in any way. Further, I might ask whether spin-off from, say, Russian defence research, even if perhaps it helps crippled people, necessarily and sufficiently justifies that research.

I believe that in many ways the shape of personal computing has already been defined. It is to do with leisure, with exercising the mind, with money and with retreating from most of the world to a small space in between the screen and our eyes, and then pretending that space is ever so important and significant.

The trouble is, as well, that we inevitably incorporate the tools we use into our model or 'internal representation' of the world. Thus we run the risk of beginning to see the world in terms which are increasingly 'to do with' computers. We may even alter our lives to fit the demands of the computer, becoming, for instance, the 'compulsive programmers' that others have already warned about.

Worse, we may start to model the world using the computer, kidding ourselves we actually simulate parts of it, including people; these models will then affect our perception, which affects the next model, which affects. . . and so on.

How else, even in these 'early' days, can we explain the unutterably boring 'computer art', that might as well be done with a Spirograph or pen and ruler, but which is exhibited for all the world as if it were at the frontiers of creativity? Or the fact that virtually all available software is either business programs or games: the former just glorified pen and paper, the latter rapidly becoming as superficial and irritating, though perhaps as addictive, as a slot machine.

The games are even advertised for these attributes! "Absolutely addictive" says the catalogue; "Yes, boring old hangman", admits the advert.

Addicted, bored, dehumanised, unable to think of much to justify the hundreds or even thousands we spent on the machine, we fly like moths to every new software candle to bask in the light. . . for a time, anyway. And will *that* be revolutionary? How about when your computer is obsolete? Just buy the latest wonder? will *that* make us free and powerful and able to. . .you know. . . control things?

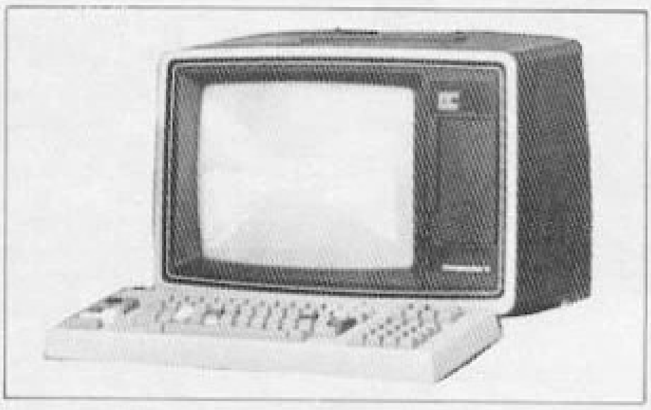
Oh yes, I have a computer. I play games and make 'art'. I hope to be contributing a regular column about anything that I can justify as being a 'valid' use of computers in the arts.

But I keep having a sneaking suspicion, as the winking cursor beckons me across the room, that the best thing one can do with one's 32K, all-singing, all-dancing, micro-processor-based delight, is to drop it from a great height onto the head of someone who's on their way out to get one — and that could be quite a few people, if I do my job right.

Brian R. Smith

COMPUCOLOR II

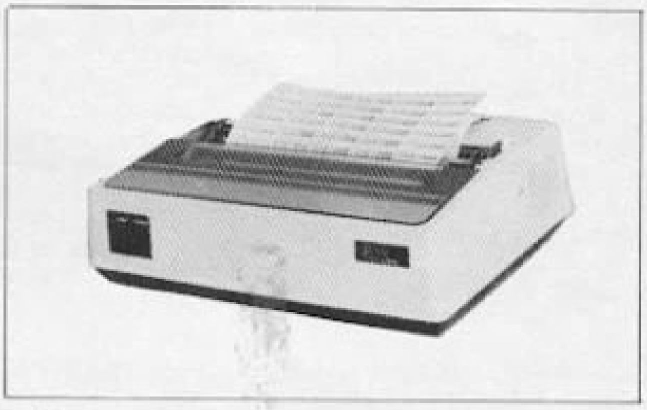
MICRO COMPUTER
from $1970 tax paid



Features
- UP TO 32K USER RAM
- 5" DISK DRIVE
- 8 COLOUR VIDEO DISPLAY
- RS232C PRINTER PORT
- FULL GRAPHICS DISPLAY

MICROLINE 80

IMPACT PRINTER
from $960 tax paid



Features
- FULL ASCII UPPER & LOWER CASE
- FULL GRAPHICS
- 80 CHARACTERS PER SECOND
- 9 x 7 DOT MATRIX
- 80 or 132 CHARACTER LINE

$ THE LOGIC SHOP PTY. LTD.

212 HIGH STREET, PRAHRAN, 3181 (03) 51 1950
91 REGENT STREET, CHIPPENDALE, 2008 (02) 699 4910

COMPUTER GAMES

AND THEN THERE WERE TWO

Chess Master, David Levy, continues his series of articles on the principles behind playing computer games with a study of the added difficulties involved in introducing a second player.

Two-person games

Two-Person games, such as chess, backgammon and draughts, are usually more interesting and challenging than one-person games, and it is to these that we shall be devoting most of our studies. The introduction of a second player creates manifold difficulties that do not exist in a one-person game, but fortunately for today's programmers these difficulties have been extensively analyzed in the computing literature and the problems are now rather well understood.

The two-person game tree

Game trees become more complex structures when an opponent appears on the scene. Let us consider a relatively simple game, noughts and crosses (tic-tac-toe to our American cousins), and examine how its tree will look after a move or two of look-ahead. We shall assume that "cross" moves first.

From the initial position there are three essentially different moves:

1) e (the centre)
2) a,c,g and i (the corners)
3) b,d,f and h (middle of the edges)

On the first move, any of group (2) is equivalent to any other, since all four moves are merely reflections or rotations of each other. Similarly, within group (3) all moves are equivalent. This technique of utilizing symmetry to reduce the magnitude of the problem is well worthwhile when programming a game that lends itself to a symmetrical analysis. By reducing the number of moves that need to be examined at any point in the tree you will be cutting execution time dramatically, because the combinatorial effects of tree growth are enormous. The savings in time that can be achieved through using symmetry can be extremely valuable when improving the performance of the program by making its evaluation function more sophisticated (and slower).

If we so decide, our program can terminate its search of the tree after looking at each of its possible moves from the root. This is called a 1-ply search because the program only looks one "ply" deep. (The term "ply" is used to denote a single move by one player.) In order to decide which move to make, out of m_1, m_2 and m_3, the program will then apply its evaluation function to the three positions at the lower end of the tree (these are called the terminal positions). Whichever position had the best score would then be assumed to be the most desirable position for the program, and the program would make the move leading to that position.

How should we set about designing our evaluation function? This is one of the fundamental problems in game playing programming because a good evaluation function will help the program to make good judgements, and hence to play well, even though the depth of look-ahead may be shallow. A poor function, on the other hand, might well result in poor play even with a deep and time consuming search of the game tree. It is therefore very much worthwhile putting some careful thought

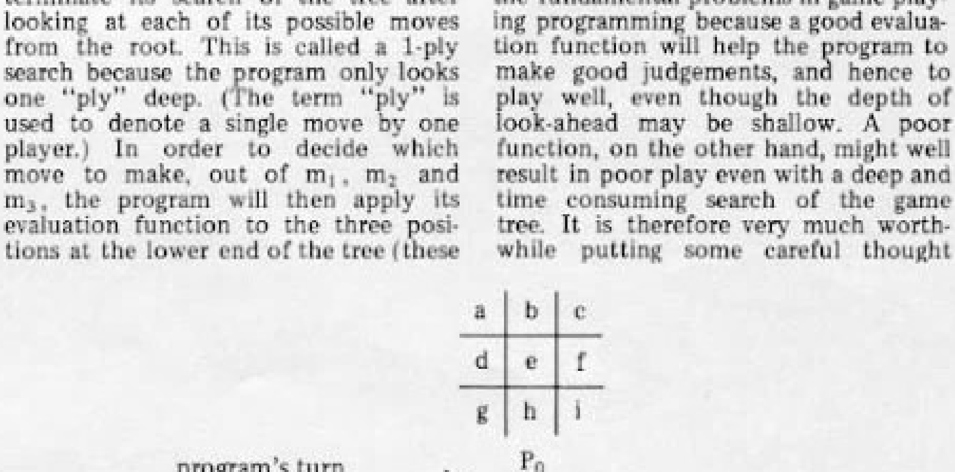


Figure 1

into the design of the evaluation function, and the following example should illustrate the type of thinking that is necessary.

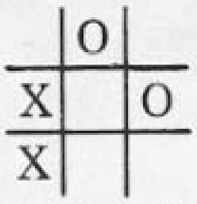
The object of the game is to create a row of three of your own symbols. We shall call this a "3-row". The next most important thing is to prevent your opponent from making a 3-row, which means that he should not have a 2-row after you move (a 2-row has two symbols of one player and one empty space). Next most important is the creation of your own 2-rows; then it is important not to leave your opponent with 1-rows (one of his symbols and two empty spaces); and finally you should try to create your own 1-rows. All of these features could well be incorporated into a noughts and crosses evaluation function.

If we denote the number of cross' 3-rows by c_3, the number of nought's 2-rows by n_2, the number of cross' 2-rows by c_2, the number of nought's 1-rows by n_1, and the number of cross' 1-rows by c_1. . . then one measure of the merit of a position from cross' point of view would be:

$$c_3 - n_2 + c_2 - n_1 + c_1$$

but this measure has one obvious drawback. It does not allow for the fact that the term c_3 is more important than n_2, which is more important than c_2, and so on. This can be done by multiplying each of the terms in the evaluation function by some numerical weighting, in such a way that the weightings (hopefully) reflect the relative importance of each feature. The evaluation function then becomes
$(k_3 \times c_3) - (k_2' \times n_2) + (k_2 \times c_2) - (k_1' \times n_1) + (k_1 \times c_1)$
where k_3, k_2', k_2, k_1' and k_1 are the numerical weightings. Since one c_3 is worth more than all the n_2s in the world, i.e. a winning row is more important than any number of 2-rows, we can set k_3 to be some arbitrarily high number, say 128. By studying the game for a few minutes it is possible to see that if one side has a 3-row, the other side may have at most two 2-rows, so to reflect the relative importance of one's own 3-rows and enemy 2-rows it is necessary to ensure that $k_3 > 2 \times k_2'$. We can therefore try $k_2' = 63$. (If one side has a 3-row and his opponent two 2-rows, the opponent will not have any 1-rows to upset this scoring mechanism).

If there are no 3-rows, but one side only has a 2-row, his opponent cannot have more than three 1-rows, as in the following situation.



So $k_2' > 2 \times k_1$ and $k_2 > 2 \times k_1'$
and we can try $k_2 = 31$, $k_1' = 15$ and $k_1 = 7$. Remember that we can modify these values in the light of experience with the program, the values 128, 63, 31, 15 and 7 are merely our first estimates. Having made these estimates we should then ensure that the score for a noughts and crosses position will never cause an overflow, and we do this by setting up positions which will have the largest and smallest possible scores, and counting the number of 3-rows etc. in each. This is a very important part of evaluation function design, and I remember a chess programmer who could not understand why his program crashed whenever it was winning or losing by a great margin — he had forgotten to allow for the possibility of one side being two queens ahead and when that happened his evaluation calculations created an overflow.

If we now return to figure 1 we can see that each of the three possible first moves results in the creation of a different number of 1-rows. Applying the evaluation function.

$128 \times c_3 - 63 \times n_2 + 31 \times c_2 - 15 \times n_1 + 7 \times c_1$

to the three positions P_1, P_2 and P_3 we find that in each case $c_3 = n_2 = c_2 = n_1 = 0$, and therefore:

$$S_1 = 128 \times 0 - 63 \times 0 + 31 \times 0 - 15 \times 0 + 7 \times 4 = 28$$
$$S_2 = 128 \times 0 - 63 \times 0 + 31 \times 0 - 15 \times 0 + 7 \times 3 = 21$$
$$S_3 = 128 \times 0 - 63 \times 0 + 31 \times 0 - 15 \times 0 + 7 \times 2 = 14$$

and S_1 is the most desirable of these scores so the program would make the move m_1 to reach position P_1 (i.e. it would play in the centre).

The 2-ply search

The 1-ply search is the simplest form of tree search in a two-person game, but it does not take into account the fact that once the program has made its move there is an opponent waiting to reply. It may be the case that a move which, superficially, looks strong, is seen to be an error when we look a little bit further into what may happen. The 2-ply search will "see" more than the 1-ply search and so moves made on the basis of a 2-ply search will be more accurate (provided that the evaluation function is not a disaster area). How can we take into account this extra dimension of the opponent's move?

Let us look at the same tree, grown one ply deeper, i.e. to a total depth of two ply — one move by the program and one move by its opponent.
If "cross" plays in the centre, "nought" has two essentially different replies, in a corner or on the middle of an edge (represented by positions P_{11} and P_{12} respectively). If "cross" makes his first move in a corner (P_2), "nought" will have five different reply moves (m_{21} m_{22} m_{23} m_{24} and m_{25}) leading to positions P_{21} P_{22} P_{23} P_{24} and P_{25}. After "cross" plays move m_3, "nought" again has five replies. It is easy to see how the tree grows. In last month's example, the 8-puzzle, the

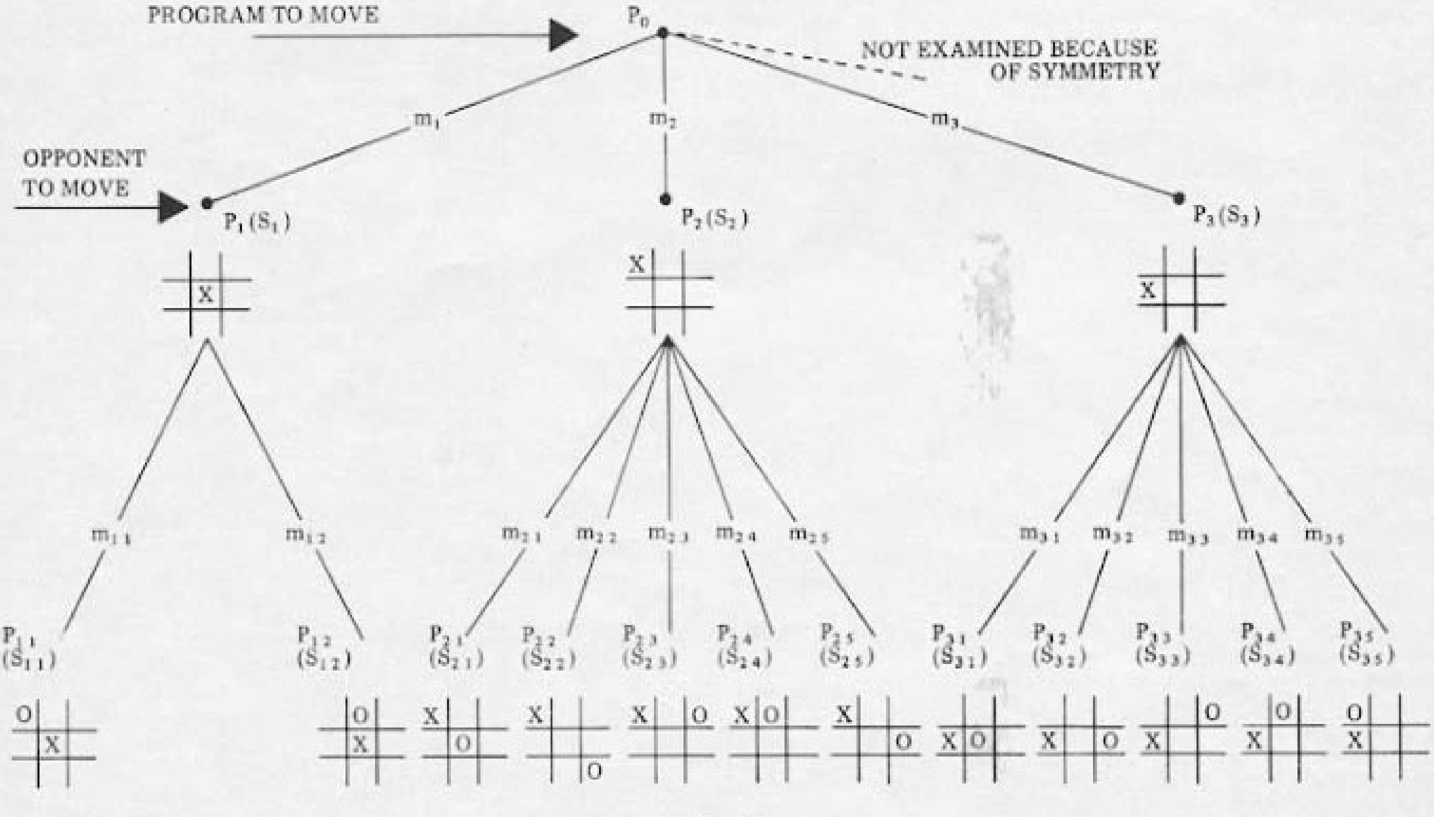


Figure 2

branching factor (number of branches from each position on the tree) was never more than three. Here it is more, even allowing for symmetry.

Let us consider how the program might analyze the situation. It uses its evaluation function to assign scores to the terminal nodes P_{11} and P_{12}. In each case $c_3=n_2=c_2=0$. In position $P_{11}, c_1 = 3$ and $n_1 = 2$. In position $P_{12}, c_1 = 3$ and $n_1 = 1$.

We now have:

$S_{11} = (-15 \times 2)+(7 \times 3) = -9$

$S_{12} = (-15 \times 1)+(7 \times 3) = 6$

This information indicates that if the program is sitting in position P_1, with its opponent to move, its opponent may choose between moves m_{11} (leading to position P_{11} of value -9) and m_{12} (leading to position P_{12} of value 6). The program's opponent wants to minimize the score and so it would choose move m_{11}, for a score of -9, and so the real value of position P_1, represented by S_1, is this *backed-up* score of -9.

If we apply the evaluation function to positions $P_{21} \ldots P_{25}$ we will get:

$S_{21} = (-15 \times 3) + (7 \times 2) = -31$

$S_{22} = (-15 \times 2) + (7 \times 2) = -16$

$S_{23} = (-15 \times 2) + (7 \times 2) = -16$

$S_{24} = (-15 \times 1) + (7 \times 2) = -1$

$S_{25} = (-15 \times 2) + (7 \times 3) = -9$

Wishing to minimize the score when making its move from P_2, the program's opponent would choose move m_{21}, leading to position P_{21} and a score of -31.

Similarly, when applying the evaluation function to positions $P_{31} \ldots P_{35}$, we get:

$S_{31} = -38$

$S_{32} = -8$

$S_{33} = -31$

$S_{34} = -16$

$S_{35} = -23$

so the program's opponent, when making its move from P_3, would choose move m_{31} for a score of -38.

We now have the following situation. If the program makes move m_1, its opponent, with best play, can achieve a score of -9. If the program plays m_2 then its opponent can achieve a score of -31. If the program plays m_3 then its opponent can score -38.

Just as the program's opponent wishes to minimize the score, so the program wishes to minimize the score. The program must now choose between m_1 (for -9), m_2 (for -31) and m_3 (for -38). Since the maximum of these three values is -9, the program will play move m_1, and the backed-up score at the root of the tree will be -9. This represents the score that will be achieved with best play from both sides.

This procedure of choosing the maximum of the minimums. . . etc. is known, not surprisingly, as the minimax method of tree searching. It is an algorithm that finds the move which will be best, assuming correct play for both sides, provided that the evaluation function is reasonably accurate.

Memory requirements for a minimax search

One of the great advantages of the minimax type of search is that it is not necessary to retain the whole tree in memory. In fact it is necessary to keep only one position at each level of look ahead, together with a certain amount of information about the moves from each of these positions. Let us see how this works for our 2-ply tree.

From the initial position P_0, the program generates the first move for cross, to position P_1. Before proceeding to the other moves that cross can make, the program generates the first reply move by nought, m_{11}, reaches position P_{11} and assigns it the score S_{11} (-9). This is the first terminal node to be evaluated, so the score of -9 represents the best score found so far and this is the score that is assigned to S_1. Since P_1 is the first move at 1-ply to be examined, this score of -9 also represents the best score found so far at the 1-level, and this is the score assigned to S_0

The program now looks at P_{12}, which we sometimes refer to as the brother of P_{11} (and P_1 is father to both of them). The program determines the score S_{12}, compares this value (6) with the best score found so far at this level (-9) and finds the -9 preferable, so the scores S_1 and S_0 need not be adjusted at this stage. The program next looks for a brother to P_{11}, but finding none it goes back up the tree and looks for a brother to P_1, which leads it to position P_2 and then to P_{21}. On the way down this part of the tree the program assigns to P_2 a score of -9, since this is the best that can be achieved so far. When looking at P_{21} the program finds a score of -31, which is better for the program's opponent than -9 and so S_2 is now set to -31.

Note that as this process continues, the brother nodes that have been examined in the past no longer serve any useful purpose and so they can be discarded. At the present point in our search we no longer need the brother of P_2 that has already been examined (P_1), so P_1 and its successor nodes are not kept in the tree at this time. The tree, at this moment, comprises only P_0, P_2 and P_{21}.

Having evaluated P_{21} we throw it away and look at P_{22}, which has a score of -16. The program's opponent would not prefer this to the -31 already discovered, and so no change is made to S_2. The program discards P_{22} and replaces it with P_{23} for a score of -16, also of no value to the program's opponent, and this is replaced in turn with P_{24} and P_{25} which also produce no change in S_2.

Since S_2 (-31) is less attractive for the program than the best score found so far (-9 at S_0), the score at P_2 is not backed-up. P_2 itself is discarded to make way for P_3, and the same process continues, with the program looking in turn at the scores of $P_{31} \ldots P_{35}$

Task for the month

The evaluation function for noughts and crosses which we have been using in this example has five features. Try to devise evaluation functions with as few features as possible, for playing noughts and crosses with (a) a 2-ply search; and (b) a 3-ply search, and test your functions by writing a program to play the game using a minimax search. The fact that deeper search will sometimes compensate for a less powerful evaluation function may make it possible for you to reduce the number of features while still writing a program that can play perfectly. If you complete this task, or even if you do not, you might like to think of a way to make the search much faster. This will be the subject of next month's article.

Good heavens a floppy disc!

BUZZWORDS

F

Files
A collection of related records treated as a unit.

Firmware
Software that is permanently stored in hardware, such as an interpreter in ROM.

Flags
Bits which provide information about specific aspects of an operation performed by the CPU, such as whether the result is positive or negative.

Floppy Disc Unit
Similar to 'Disc Memory Unit' but utilizing flexible rather than hard discs. The unit is also characterized by having a slower mode of operation and reduced memory capacity as compared to a hard disc system.

Flow Chart
A graphic representation of the more important steps of work in a process, using symbols such as rectangles and circles to denote key classes of operations. The emphasis is on where and by whom work is done rather than on how it is done. A flow chart normally forms part of any systems analysis preparatory to the design of a computer program.

FORTRAN – FORmula TRANslation
A long-established computer programming language, particularly designed for scientific applications. As with other programming languages, various dialects evolve after a few years: thus FORTRAN IV and FORTRAN 77 may be found.

Front Panel
The collection of switches and indicators (or the place where they are located) whereby the operator may control a computer. In many modern machines the functions formerly served by separate rows of lights and switches have now been incorporated in the normal keyboard and display.

G

Graphics
The use, particularly on a computer's display screen, of graphs, bar charts and other designs in place of or in addition to the usual alphanumeric character set.

H

Hard Copy
Refers to the printed output of a computer as opposed to, for example, output appearing on a VDU or from a voice synthesizer.

Hardware
A term applied to the physical parts of a computer. To be distinguished from software.

Handshake
An exchange of signals between a computer and a peripheral device which establishes synchronisation for transmission of data.

High Level Language
A program coding system orientated towards the programmer rather than the computer. High level languages tend to save programming time at the expense of less efficient use of hardware and slower processing. An interpreter or compiler is required to translate each high level instruction into machine language (usually more than one machine instruction is generated from each high level instruction). Examples of popular high level languages are BASIC, FORTRAN and COBOL.

High Level Programming
Programming a computer using a high level language instead of the complicated machine language level.

I

I/O-Input/Output
Refers to the transfer of data between a computer and peripheral devices.

I/O Ports
The physical devices which provide electrical access to a computer by external devices for the transmission of data.

IC
A miniaturized electronic circuit formed from a piece of semiconductor and housed in a nonconducting, heat dissipating package.

Immediate Address
In this mode, the instruction itself contains the operand rather than the operand being stored in a separate memory location. This form of addressing is typically used in operations requiring the operand to be a constant.

Indexed Addressing
The memory location to be addressed is pointed to by the sum of the contents of an index register and the last byte of the instruction.

Indirect Addressing
Addressing a memory location which contains the address of data rather than the data itself.

Interrupt
The temporary suspension of the execution of a program following detection of a specified condition, for example: the lack of paper in a printer.

J

Justify
To arrange printed (or typewritten) words so that the right-hand margin of each line is vertically below that of the preceeding line.

K

Keyboard
A group of buttons arranged in a manner similar to that of a typewritter and used to input alphanumeric characters to a computer.

Keypad
A keyboard with a smaller number of buttons typically used on calculators or as an additional input device for microcomputers. It usually comprises numeric characters and, in the case of calculators, instruction keys such as add and divide.

Keyword
1. A group of characters which identifies an item or record for data retrieval.
2. A secret combination of characters which identifies an authorised user to a computer.
3. A group of characters which indicates the main function of a program line in a high level language.

L

Labels
One or more characters used to identify the location of an instruction (when line numbers are not used) within a program. The process of compiling such a program will replace each label with an absolute address.

Library
A collection of programs and routines written for a particular computer, minicomputer or microcomputer.

Load Instruction
An instruction which, when executed, will transfer operand data from memory to a CPU register or from one CPU register to another. In some computers, it also applies to a transfer of data from a CPU register to memory.

LSI–Large Scale Integration
The fabrication on a small piece of crystalline silicon of a circuit embodying large numbers of semiconductor devices (normally between 100 and 10,000).

MS Microsoftware

TAPE 1 LEVEL 2

Mortgage calculations, Dow Jones Industrial, cash flow, inventory-change, California income tax, journal ledger (8K), loan amortization, perpetual calendar, bio rythm, payroll, diet planning, speed reading, touch typing, sales receiptally, decision maker, mail addressing, straight depreciation, double-declining depreciation, and revolving charge account.

Also, math problems, queen, Star Trek I, number guessing, wheel of fortune, World War II bomber, rock-scissors-paper, seek, Star Trek II, Red Baron, mini-Trek, strategy, pilot, battleship, "On A Snowy Evening", Mastermind, tic-tac-toe, grand prix auto race, capitals, etch sketch, hangman.

TAPE 2 LEVEL 2

Fully documented in Some Common Basic Programs by Lon Poole & Mary Borchers:

Investment, future value regular deposits; regular withdrawals, initial minimum (for withdrawals); nominal interest, effective & earned-interest; depreciation rate, amount depreciation; salvage value; discount commercial paper; loan principal, regular and last payment, remaining balance, term-loan; mortgage amortization; greatest common denom. - integer prime factors; polygon area; triangle parts; analysis, operations two vectors; radian-degree., degree-radian conversion; co-ordinate, polar equation, functions plot; linear, curvilinear interpolation; Simpson's & trapezoidal rules, Gaussian quadrature integration; derivative.

Side 2 – quadratic equation, polynominal (Newton) & half inteval-search roots; trig. polynominal; simultaneous equations; linear programming; matrix addition, subtraction; scalar multiplication, inversion; permutations & combinations; Mann-Whitney U test; mean, variance, standard deviation; geometric mean & deviation; binomial, Poisson normal, Chi-square distribution; Chi-sq., student's T distribution test; F-distribution; linear correlation coefficient; linear, multiple-linear, Nth order, geometric, exponential regression; system reliability; future projections; Federal withholding taxes: tax depreciation schedule: check writer; recipe cost; map check; day of week; days between two dates; anglo to metric; alphabetize.

TAPE 4 LEVEL 1

Election returns, business percentage, ups and downs of business, index, inventory control, sales receipt tally, gas mileage, driving distance, mixed monthly sales report, payroll, annual earnings, speech recording aid, and double-declining depreciation.

Also, math problems, cash register, chase, snoopy, commander-in-chief, Christmas graphic, air raid, balance scale, stock market, tic-tac-toe and On A Snowy Evening.

TAPE 5 LEVEL 2

Memory test, mortgage payments, tension breaker, lineprinter-screen & vice-versa utilites, Federal income tax, election returns, business percentage, vacation planner, car pool (disk), diet planning 2, mailing list (disk) and first aid.

Also spelling bee, Star Trek 3, mind bender, tachistoscope, chase, common factor, klingon capture, spelling practice, Hamurabi, animals, Snoopy, cryptogram, starship, ants, Yesterday, and Pilot (disk). Pilot is the language of computer-aided instruction (CAI).

TAPE 7 LEVEL 2

Disassembler, Pilot, roster, dropout, memory loader, memory sort, inventory control, graph, land surveying, mixed monthly sales report, shopping list, diet planning 3, loan progress chart, hex-decimal conversion.

Also Star Trek 4, states and capitals, battleships 2, spelling practice 2, number guessing, hangman 2, snark, slot machine, cipher, target, surround, adder, termites, lunar lander, multiplication exercise, five-in-a-row, Bastem, and write. A number after a program indicates there are other similar People's Software programs. Pilot is the same as the disk pilot on tape 5, except runs on 16K tape systems.

Each Tape $9.90 (incl. postage within Australia)

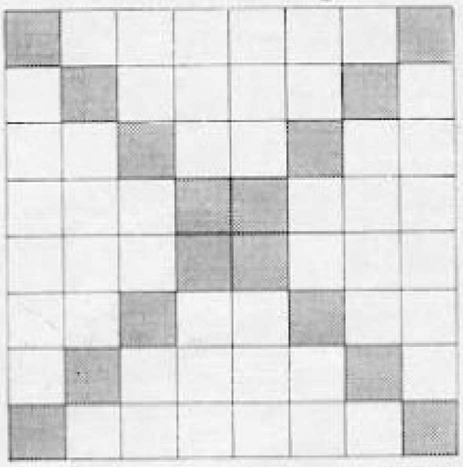
Please mail your order to:

MS Microsoftware,
P.O. Box 119, Essendon, Victoria, 3040.

LEISURE LINES

Well, you've had a month's recuperation period since the last issue, so I hope you're all fit and fresh and ready for June's edition of Leisure Lines. First of all it's time for a 'quickie' . . . that means no answers, no prizes.

Three men in a restaurant. The bill came to $30, so each man gave a $10 note to the waiter. The waiter took the bill and delivered the payment to the manager – who decided he'd overcharged. He gave the waiter $5 to return to the men. On the way back to the table the waiter dishonestly pocketed $2 of the $5 for himself, and therefore he gave $1 back to each of the men. That means that each man had paid $9 – making a total of $27. The waiter had taken $2, the men had paid $27. . . that comes to $29. What happened to the missing $1?



So much for the 'kid's stuff', time now for this month's prize puzzle.

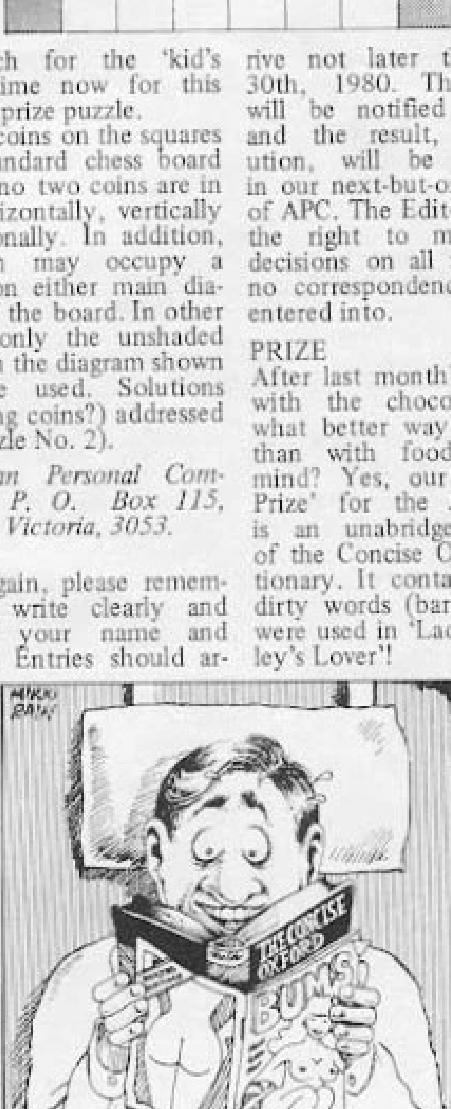
Place 8 coins on the squares of a standard chess board so that no two coins are in line horizontally, vertically or diagonally. In addition, no coin may occupy a square on either main diagonal of the board. In other words, only the unshaded square in the diagram shown may be used. Solutions (including coins?) addressed to: (Puzzle No. 2).

Australian Personal Computer, P. O. Box 115, Carlton, Victoria, 3053.

Once again, please remember to write clearly and include your name and address. Entries should arrive not later than June 30th, 1980. The winner will be notified by post and the result, plus solution, will be published in our next-but-one edition of APC. The Editor reserves the right to make final decisions on all matters – no correspondence will be entered into.

PRIZE

After last month's gluttony with the chocolate bars, what better way to follow than with food for the mind? Yes, our 'Silly (?) Prize' for the June issue is an unabridged version of the Concise Oxford Dictionary. It contains all the dirty words (bar two) that were used in 'Lady Chatterley's Lover'!



C.I.S.A.
159 KENT STREET, SYDNEY

The One-stop Microcomputer Shop Providing a Total Service to TRS-80* and System 80 Users.

(TRS-80 is a registered Trademark of Tandy Radio-Shack)

HARDWARE AND MODIFICATIONS

Having system crashes? Plug in our line filter and get main-frame stability	$ 55.00
Use your TRS-80 as a telephone book and automatic dialler. No hardware mods required. All software included.	$ 59.50
LIGHT PEN with demonstration software and full programming instructions STILL ONLY	$ 19.50
Lower case mod for ELECTRIC PENCIL and SCRIPSIT. Fitted and 90 day guarantee	$ 49.95
VIDEO STABILIZER CRYSTAL. Get professional stability on your VDU. CRYSTAL only	$ 19.95
Fitted and guaranteed	$ 39.00
16K MEMORY KITS	$ 85.00
We install	$105.00
RS-232 PRINTER DRIVER. Operates from cassette cable. Including software.	$ 54.75
CISA DATA DIGITIZER. Our most popular hardware product. No hardware mods required. Reads and re-records tapes normally unreadable by any other method	$ 57.50
CASSETTE CABINET. Protect your valuable cassettes in this strong, handsome 36 cassette cabinet	$ 27.75

DIGITAL CASSETTES – Top quality

C15's	1 – $1.65	10 – $14.85
C20's	1 – $1.85	10 – $16.85
C30's	1 – $1.99	10 – $18.50

This is an entirely new product. We are so confident of this product that we are now recording all our commercial software on it.

5¼" PRIME QUALITY DISKETTES

1 – $4.95	10 – $45.00

All tracks guaranteed to load

We are extending our range of computer paper and labels for the enthusiast and the business user.

LATE SPECIAL!

EXATRON STRINGY FLOPPIES	P.O.A.
77 Track Micropolis Drives complete with DOS	P.O.A.

BOOKS

We believe that we have the best and most comprehensive range of titles of interest to the TRS-80 owner/user. Over 200 titles presently in stock. COME IN AND BROWSE AROUND!

JUST A SELECTION

First Book of Microcomputers	$ 6.50
Pascal with Style	$ 8.50
Basic with Style	$ 8.00
TRS-80 Disk and Other Mysteries	$29.50
Level II ROM Reference Manual	$24.95
Basic and the Personal Computer (highly recommended)	$15.95
Basic Programming for Scientists and Engineers	$15.50
The Incredible Secret Money Machine	$ 7.95
Z-80 Microcomputer Handbook	$11.95
Microcomputers for Business Applications	$11.95
Z-80 and 8080 Assembly Language Programming	$10.00
How to Profit from your Personal Computer	$12.00
Little Book of Basic Style	$ 7.95
Microcomputer Interfacing	$25.75
Microcomputers and the 3 R's (A Guide for Teachers)	$10.00

COMPUTER CLASSICS

Wirth/Algorithms + Data Structures = Programs	$29.75
Tausworth/Standardized Development of Computer Software	
Vol. I – Methods	$29.75
Vol. II – Standards	$29.75
Klingman/Microprocessor Systems Design	$30.50
Gilmour/Business Systems Handbook	$29.75
Gane/Structured Systems Analysis	$26.50
Garrett/Analog Systems for Microprocessors	$26.50
Rhyne/Fundamentals of Digital Systems Design	$26.95

TRS-80 AND SYSTEM 80 SOFTWARE

Ecology Simulations 1 and 2 (disk and lower case) each	$24.95
Mission Impossible and Voodoo Castle (disk)	$24.95
Adventureland and Pirate Adventure (disk)	$24.95
Investment Analysis (disk or L2 cassette)	$49.95
Sargon II (cassette)	$37.50
Strategy Games (5) (cassette)	$ 7.95
Space Games (4) (cassette)	$ 7.95
5 Adventures (cassette) each	$14.95
Advanced Statistics (disk or cassette)	$24.95
U Boat (cassette)	$ 8.50
Basic Monitor (cassette)	$19.95
Micromazing (games cassette)	$ 7.50
RPN Calculator (cassette)	$14.95

PLUS MANY MORE

We stock and highly recommend all RACET COMPUTES professional standard system programs.

Infinite Basic	$ 49.95
Infinite Business	$ 29.95
Remodel	$ 24.95
Remodel + Proload	$ 34.95
GSF	$ 24.95
DOSORT (sorts disk files)	$ 34.95
COPSYS (copies system tapes)	$ 34.95

NEW DOS FROM APPARAT

35 Track	$ 99.00
40 Track	$110.00

This DOS emulates many mainframe capabilities. Has features usually restricted to much larger systems.

TARANTO and ASSOCIATES

General Ledger	$155.00
Accounts Payable	$155.00
Accounts Receivable	$155.00
Print Invoice	$ 99.00
Complete Business System	$500.00

This is professional quality software and really provides a superb business system for the TRS-80. Software maintenance is available to purchasers of the complete system at $15 per hour, to tailor the programs to your specific needs.

PLUS MANY MORE

We are the Australian agents for many first class U.S. software houses. Send for our full catalogue.

Remember, for all your TRS-80 and System 80 needs, phone, write or call

TRADE ENQUIRIES INVITED

Showroom:
Ground Floor, GIO Building, 159 Kent Street, Sydney 2000
(opposite IBM Building)
Phone: (02) 241 1813

We regularly conduct a series of evening periods of small group tuition from 7.30 p.m. onwards in BASIC with specific emphasis on commercial applications.

These sessions are conducted in a friendly and helpful atmosphere and are designed to give you a complete mastery of BASIC techniques up to and including all aspects of disc I/O and file handling.

We will teach you tricks even Tandy don't know about!

All stationery and workbooks are provided. Textbooks available at an extra charge.

Fee for 10 periods (up to 3 hours each) $150.00

Special private or group tuition on any aspect of BASIC or commercial/business applications of microcomputers available on request.

Tuition includes assignment work if required – and assessment.

A certificate of competence is awarded on successful completion of a tuition series.

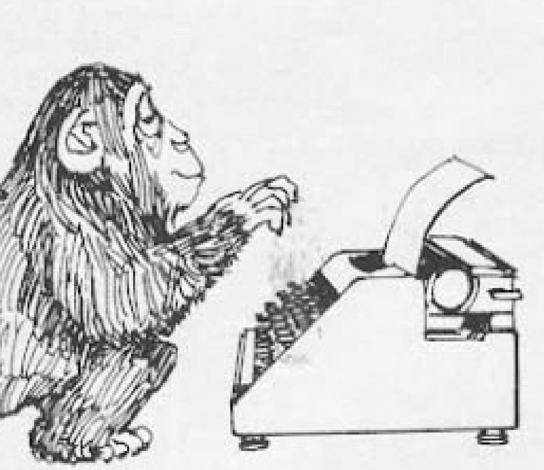
REMEMBER – for all your TRS-80 needs – phone, write or call

CISA

159 KENT STREET,
SYDNEY 2000
(opposite IBM building)
Phone: (02) 241 1813

We are currently preparing a correspondence course for out-of-town TRS-80 users. Details will be available shortly.

We carry the full range of MICRO-80 newsletters and recorded software plus a range of popular microcomputing journals.



RANDOM WRITINGS

By Michael James

If the idea of a book, a fairly hefty volume too, containing nothing but a million "random" numbers[1] seems absurd to you then so should the idea of your computer or pocket calculator producing random digits, e.g., 1 to 6, for your computer games. The whole idea of randomness seems to be against the act of filling a book with random numbers. The fact that one could look at the same page more than once to find out what was coming wipes out the usual element of surprise that the word random contains! It is perhaps less obvious that computer generated random numbers are just as repeatable – and hence expected. In fact some applications demand that a sequence of random numbers is repeatable. Where does all this madness take us? In short, what is random about "random" numbers?

Randomness v. Pseudo-randomness

We all have a clear idea about randomness. The flip of a coin. The fall of a dice. These events are random. Their outcome is not predictable and not repeatable (at will). If we wanted to build an electronic dice into a computer, to enable us to play games say, then a direct solution would be to take some electronic device which behaved randomly, for example, the output of a Zener diode or a saline cell. We could use the device to determine the state of a memory location in our computer and bingo! we have our random number generator. This is the way ERNIE, your friendly premium bond selecting machine works. This is a very good technique for generating random numbers that nobody can even guess at – it is the lack of repeatability which is important here. However, even though fair and right for the job, machines such as ERNIE are very expensive things to construct well and for a lot of applications a much simpler solution will do.

Let us think about the properties we would like a computer-simulated dice to have.

1) Each digit should come up, on average, as often as any other.
2) By observing which digits have come up already it should not be possible to predict the next digit.

Condition one concerns the fairness of the dice and condition two concerns the independence of consecutive throws. Fairness is not a difficult condition to satisfy. It is the second condition which causes all the problems. First, we should notice that there is nothing in our two conditions which says that these digits should be produced randomly in the sense of coin tossing. I could have a list of numbers in which every number occurred about equally often and in which knowledge of any set of numbers would not help me to guess the next. These would satisfy our conditions and if I read them from the list one after the other you could not tell if I was tossing a dice or reading a list. Such numbers are called pseudo-random numbers because they are not produced by a random mechanism and because they are, in principle, repeatable.

The requirement of not being able to predict the next random number in the sequence, given knowledge of the rest, is a problem because, if the numbers are generated by a non-random method, i.e. they are repeatable, then there must exist a method of predicting them! (This is, of course, using another copy of the program or list which gave rise to them in the first place). So it seems that we cannot meet the second requirement. On closer examination it is obvious that we are asking too much of our random numbers – all we need is that they are not predictable in the circumstances that we are using them in. For example, if the method of prediction is either too obscure to be deduced or too difficult to be used by a human then our random numbers are O.K. for game playing on a computer. (For most applications we usually settle for successive numbers being uncorrelated with one another).

Generating Random Numbers

One of the first computer (pseudo) random number generators, the mid-square method, was suggested by Von Neumann in 1951. It is easy to use but generates fairly low quality random numbers – it has a tendency to produce numbers like OOXY and XYOO periodically, but it is easy to understand:

1) Specify the number of digits to be generated – say four. 2) Choose any starting value – 5069. 3) Square the starting value – 25694761. 4) The next random number is in the middle four digits – 6947. 5) Steps 3 and 4 are repeated with the new random number.

A short BASIC program for the mid-square method is given below and the reader might have some fun experimenting with it.

Mid-square

```
10  INPUT "STARTING VALUE", A
20  L = LEN(STR$(A))
30  P = 10↑(INT(L/2))
40  Q = 10↑L
50  A = INT(A*A/P)
60  A = A – INT(A/Q)*Q
70  PRINT A
80  GO TO 50
```

The most popular type of random number generator in use today is the so-called congruential generator. It is not as easy to understand as the mid-square method but it does give high quality numbers with known properties. A typical generator is given below as a BASIC program. (This particular generator is also of historic interest as it was first used on ENIAC).

Congruential

```
10 INPUT "STARTING VALUE", A
20 A = A*23
30 A = A - INT(A/100000001)
                   *100000001
40 U = A/100000001
50 PRINT A,U
60 GO TO 20
```

The general congruential generator works by multiplying the old random number by a constant and then expressing it modulo some other constant to get the new random number, i.e.

$$A_{n+1} = \left[A_n * K\right] \bmod P$$

Expressing a number modulo P is simply done by finding the remainder after dividing by P. In our example K = 23 and P = 100000001. A further refinement is to divide A_n by P to give a random number between 0 and 1 (U is our example). Congruential generators repeat themselves eventually but this can take a long time and depends on the choice of K, P and A_1. (Our example can generate 5,882,352 numbers before repeating). Constructing a very good congruential generator is difficult, but our example will do for most applications.

Monte Carlo

Random numbers can be used to solve some types of mathematical problems as well as in computer game playing. For example, suppose we are about to design a garage and we want to decide how many petrol pumps to install. Too many and some will stand idle and we could have saved our capital. Too few and we will lose customers as the queues get longer. Putting this another way, what we need to know is the average length of the queue for various numbers of pumps. The answer to this problem depends on the number of customers per second and the time it takes to serve them. It is not easy to get the answer by the usual mathematical methods.

A method of solving the problem is to simulate it using a random number generator. By writing a program in which customers arrived and were served with the right probabilities, we could obtain answers simply by running the program and keeping a count of the number of customers served and turned away.

The collection of methods based on using random numbers to solve mathematical problems is generally called the Monte Carlo method. The previous simulation example is easy to understand and the role of the random numbers is obvious. However, random numbers can be used to solve problems which seem to have nothing to do with randomness.

For example, suppose we wish to evaluate

$$\theta = \int_0^1 x^2 dx$$

In other words, find the area below the graph of x^2 in the interval 0 to 1 (Fig.1). We could use the usual methods of numerical integration, i.e. Simpson's Rule, or even solve the problem directly by $x^2 = 1/3\ x^3$. But suppose we instead generate two random numbers U_1 and U_2 which define a point in the unit square, i.e. they are both positive and less than one (see Fig. 1). If U_1 and U_2 are evenly distributed then the probability of the random point being below the curve is exactly equal to the area beneath the curve. Thus if we generate N random points, the area under the curve is estimated by the probability H/N where H is the number of points below the curve. A BASIC program to carry out this method for x^2 is given below.

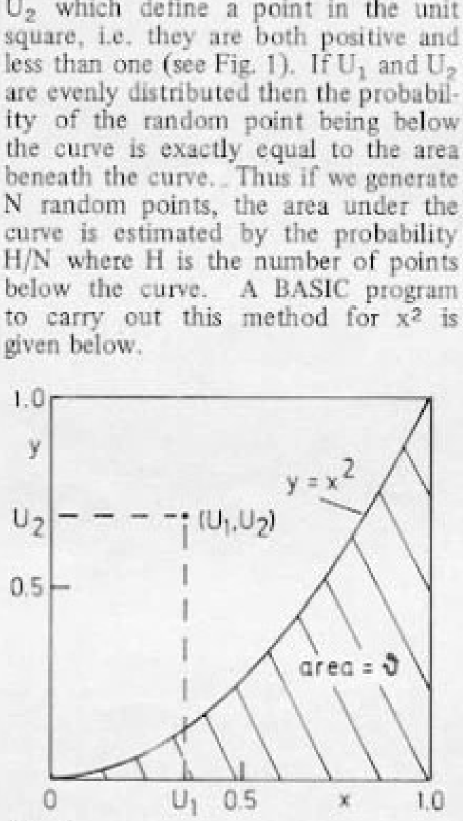


Fig. 1

Integration program

```
10  H = 0
20  N = 0
30  DEF FNA (X) = X*X
40  U1 = RND
50  U2 = RND
60  IF U2 < FNA (U1) THEN H = H + 1
70  N = N + 1
80  PRINT "AREA ESTIMATE = ",
      H/N, "N = "; N
90  GO TO 40
```

It is easy to find any one-dimensional integral over 0-1 by changing the function (FNA) statement. A quick look at the program shows that the method is simple when compared with other methods. However, a little experimentation will soon reveal its disadvantage – you have to do a lot of work to get a reasonable answer. For example, at N = 200 the estimate was 0.435 and even at 1000 it was only 0.361. (The correct value is 0.333). This would seem to make Monte Carlo integration of little use, but with a few improvements it is one of the best techniques we have for high-dimensional integrals. It is rarely used for one-dimensional integrations, i.e. finding areas, but it is nearly always preferred for two-dimensional cases, i.e. finding volumes.

There are other examples of turning a non-random problem into a random one and then solving by simulation but the reader is referred to the suggested reading at the end of this article for more details.

Testing Random Number Generators

Whenever you use a random number generator you should always satisfy yourself that it is good enough for your purpose. This can be done either by statistical tests or, for the least exacting work, simply by examining a histogram of the output.

For game playing most random number generators are good enough. For the various Monte Carlo techniques it is advisable to conduct statistical tests before relying on the results. (Details of these tests can be found in the further reading). I have tested a number of random number generators supplied with various versions of BASIC and found them all reasonable – none of them have been as bad as the mid-square method! One annoying feature of some BASIC random number generators is their randomisation. By starting the generator off with a new starting value, obtained from some arbitrary memory location, we lose the repeatability of pseudo-random numbers. This is excellent for game playing – otherwise you'd play the same game every time – but for Monte Carlo methods this is a nuisance. It is impossible to say how good such a randomised generator is because its properties depend on the starting value used.

Conclusion

Random numbers play an important part in the personal computer revolution. For game playing the random number generator supplied with BASIC (or some other high-level language) is usually good enough. Monte Carlo simulation techniques increase the usefulness of random numbers but also require better generators. A good Monte Carlo simulation is simple, effective and can be fun – after all a computer game is usually nothing more than a Monte Carlo simulation of some "real" game.

Reference:

1. A Million Random Digits with 100,000 Normal Deviates, The Rand Corporation, 1955.

Further Reading

The Generation of Random Variates, T.G. Newman and P.L. Odell, Griffin, 1971.

The Monte Carlo Method, ed. Yu. A. Schreider, Pergamon Press, 1966.

USER GROUPS INDEX

Recently APC circularised clubs and user groups – in fact, all the addresses we were able to find. Surprisingly, many proved long defunct, and indeed, a few claimed never to have existed in the first place! The list published here is correct as of going to press . . . though obviously incomplete. If YOUR group hasn't been included then you can help us make amends by posting all relevant information to: User Group Index, APC, P.O. Box 115, Carlton, 3053. Updates on alterations would also be appreciated.

VICTORIA

Catering for the SORD M100 series users of Melbourne, S.M.U.G. meets at 60 Winmalee Drive, Glen Waverley on the second Saturday of the month at 2 p.m. Although fairly small (membership of 25), this highly active group concentrates on maintaining a large range of user generated software for the SORD M100, which is free to its members.

Other projects include the development of an entire disc system which will be CP/M compatible, as well as D.O.S. compatible with the present range of SORD specific languages. If you are interested, contact Mr Robin Miller, 60 Winmalee Drive, Glen Waverley, 3150.

AUSOM (Apple Users' Society of Melbourne) is open to any Apple owner residing in Victoria, with the sole condition being that the user must not only be interested in business. The group's aim is to offer a forum through which Apple users may communicate and obtain any necessary software and documentation.

The membership is currently 20 and rapidly increasing; the annual fee being $10.00. AUSOM usually meets every third Saturday of the month at the AMRA hall, Wills Street, Glen Iris, (opposite the Glen Iris station), at 3.30 p.m. Prospective members should contact Computerland Melbourne and ask for the President, Mr David Turk.

A large group in Melbourne has recently been renamed as the Sorcerer Computer Users (Australia). Its objective is to promote the interests of Sorcerer owners. A monthly newsletter is published containing articles submitted by members, including program listings and handy hints. Subscriptions are $12.00.
The group meets each first Sunday of the month at 2 p.m. at the corner of Barkers Road and Elphin Grove, Hawthorn in the Swinburne Community Junior School Hall. Current membership is 225. Further details may be obtained from the Secretary, S.C.U.A. PO Box 144, Doncaster, 3108.

The newly inaugurated Compucolor Users' Group of Melbourne meets at 7.30 p.m. on the first Tuesday of the month at 212 High Street, Windsor. The group plans to establish some sort of software exchange service for its members. An education course on assembler language for the 8080 microprocessor is being organised – it will probably run for three meetings, commencing in July. For more information please contact Mr. L. Ferguson, 12 Morphett Avenue, Ascot Vale.

The Geelong Computer Club was established 18 months ago to allow communication between people with microcomputer interests. The group meets monthly (except January) on the second Thursday at 7.30 p.m. The usual venue is the premises of Tybar Engineering, Hampton Street, Newtown,Geelong.
Guest speakers, demonstrations or tours are arranged for the monthly meetings. As the club is sponsored by a local company, subscription fees have not been necessary. There are thirty active members, with a special subgroup devoted to the TRS-80. Prospective members should contact Mr. Peter McKeon, PO Box 93, Geelong, Victoria, 3220.

A group is currently being formed to cater for Melbourne's Pet users. Further details next month.

NEW SOUTH WALES

The Commodore Computer Users' Association has regular meetings every fourth Wednesday at Room 3, 9th Floor, the Teachers' Federation, 300 Sussex Street, Sydney at 8 p.m. A guest speaker lectures on a topic of interest at each meeting. The group's membership is approximately 50.

The Compucolor Users' Group of Sydney meets on the first Tuesday of the month at 91 Regent Street, Chippendale. The meetings begin informally at 6 p.m., giving members the opportunity for private conversation, with the official start being at 7 p.m. Like their Melbourne counterparts, this club has only recently formed. They intend to present guest speakers at some meetings and generally help their members to gain the most from their Compucolors. Current membership is 14. For more details contact Mr Andrew MacIntosh, C/- Regent Street, Chippendale.

QUEENSLAND

The IREE Microcomputer Interest Group of Brisbane meets every second Friday of the month. The meetings commence at 7.30 p.m. at the T.A.F.E. building, Old South Brisbane Town Hall, corner Vulture and Graham Streets, South Brisbane. The club membership is currently 100, and visitors are always welcome.
At each meeting, a lecture and demonstration covering an area of interest to members are given. Several times a year, the group publishes a newsletter which details the topics to be discussed at forthcoming meetings and gives information concerning the local

USER GROUPS INDEX

hardware and software retailers. In conjunction with T.A.F.E., courses have been arranged to assist those interested in microcomputers. More details from Mr N. Wilson, PO Box 81, Albion, Queensland, 4010.

An ambitious club has been formed in Queensland – The Brisbane Youth Computer Group. Its aim is to allow students to gain experience with computers. A Software Exchange Service is currently being established to operate free for members. The group is keen for new members and contributors to their Exchange Service.
The President is interested to hear from anyone willing to form a similar group in other cities or states, eventually to form a national group. Mr A. Harrison, PO Box 396, Sunnybank, Queensland 4109, should be contacted for more information.

SOUTH AUSTRALIA

An opportunity for TRS-80 owners and users to meet informally and discuss ideas and problems is offered by the TRS-80 Users' Group of Adelaide. The club meets on the first Thursday of each month (except January) at 7.30 p.m. (Although officially approved by Tandy, the group remains independent).
No fee is charged and the group does not have a library or newsletter. However, a member speaks on a subject requested each month, and then general discussion and private talk is encouraged between members. The organisers feel that this informal structure of meetings is very successful. Further information, including meeting venues, can be obtained from Mr. G. Stevenson of 36 Sturt Street, Adelaide, 5000.

A.C.T.

MICSIG was founded by members of the Canberra branch of the A.C.S., but is not confined to A.C.S. members. Meetings are held at 7.30 p.m. on the second Tuesday of the month, and are advertised in the "Canberra Times" on the preceding Saturday. The annual subscription fee is $10, with a half-rate concession for full-time students and pensioners. Further information is available from The Registrar, MICSIG, C/– PO Box 446, Canberra City, ACT, 2601.

NEW ZEALAND

The Wellington Microcomputing Society Inc. was formed in mid 1977 with the aim of providing a means of communication and interchange of ideas between members. The membership of 60 includes people from a wide range of microcomputing areas and levels. The fee is $10.00, which is reduced to $5.00 for those joining in the latter half of the year or who reside out of the area.
The club priviledges include the option of borrowing a TRS-80 or D2 kit as well as books from hardware and software libraries; accessability to PROM programmers; an on-going constructional course building an extended M6802 system; and free advertising in their newsletter to buy, sell or swap within the club.
Meetings are held on the second Tuesday of each month (excluding December and January) at the Victoria University of Wellington, Lecture Block 2, between 7.30 and 10.30 p.m. The format is formal business, general discussion and then a guest speaker; visitors are welcome. Further details may be obtained from Lindsay Williams, 2 Pope Street, Plimmerton, New Zealand.

FAX

APC introduces the first of a series of reference sheets with, this month, the 8080 instruction set. We plan to give you similar charts with the op-codes for all the common processors. Other areas we shall cover are standard codes — ASCII, EBCDIC, BAUDOT etc, hardware interface standards and protocols and anything else which lends itself to this format.

THE 8080 MNEMONICS ARRANGED BY OP CODE

Compiled by John A. Coll.

MSB / LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP				MOV B,B	MOV D,B	MOV H,B	MOV M,B	ADD B	SUB B	ANA B	ORA B	RNZ	RNC	RPO	RP	0
1	LXI B	LXI D	LXI H	LXI SP	MOV B,C	MOV D,C	MOV H,C	MOV M,C	ADD C	SUB C	ANA C	ORA C	POP B	POP D	POP H	POP PSW	1
2	STAX B	STAX D	SHLD	STA	MOV B,D	MOV D,D	MOV H,D	MOV M,D	ADD D	SUB D	ANA D	ORA D	JNZ	JNC	JPO	JP	2
3	INX B	INX D	INX H	INX SP	MOV B,E	MOV D,E	MOV H,E	MOV M,E	ADD E	SUB E	ANA E	ORA E	JMP	OUT	XTHL	DI	3
4	INR B	INR D	INR H	INR M	MOV B,H	MOV D,H	MOV H,H	MOV M,H	ADD H	SUB H	ANA H	ORA H	CNZ	CNC	CPO	CP	4
5	DCR B	DCR D	DCR H	DCR M	MOV B,L	MOV D,L	MOV H,L	MOV M,L	ADD L	SUB L	ANA L	ORA L	PUSH B	PUSH D	PUSH H	PUSH PSW	5
6	MVI B	MVI D	MVI H	MVI M	MOV B, M	MOV D,M	MOV H,M	HALT	ADD M	SUB M	ANA M	ORA M	ADI	SUI	ANI	ORI	6
7	RLC	RAL	DAA	STC	MOV B,A	MOV D,A	MOV H,A	MOV M,A	ADD A	SUB A	ANA A	ORA A	RST O	RST 10H	RST 20H	RST 30H	7
8					MOV C,B	MOV E,B	MOV L,B	MOV A,B	ADC B	SBB B	XRA B	CMP B	RZ	RC	RPE	RM	8
9	DAD B	DAD D	DAD H	DAD SP	MOV C,C	MOV E,C	MOV L,C	MOV A,C	ADC C	SBB C	XRA C	CMP C	RET		PCHL	SPHL	9
A	LDAX B	LDAX D	LHLD	LDA	MOV C,D	MOV E,D	MOV L,D	MOV A,D	ADC D	SBB D	XRA D	CMP D	JZ	JC	JPE	JM	A
B	DCX B	DCX D	DCX H	DCX SP	MOV C,E	MOV E,E	MOV L,E	MOV A,E	ADC E	SBB E	XRA E	CMP E		IN	XCHG	EI	B
C	INR C	INR E	INR L	INR A	MOV C,H	MOV E,H	MOV L,H	MOV A,H	ADC H	SBB H	XRA H	CMP H	CZ	CC	CPE	CM	C
D	DCR C	DCR E	DCR L	DCR A	MOV C,L	MOV E,L	MOV L,L	MOV A,L	ADC L	SBB L	XRA L	CMP L	CALL				D
E	MVI C	MVI E	MVI L	MVI A	MOV C,M	MOV E,M	MOV L,M	MOV A,M	ADC M	SBB M	XRA M	CMP M	ACI	SBI	XRI	CPI	E
F	RRC	RAR	CMA	CMC	MOV C,A	MOV E,A	MOV L,A	MOV A,A	ADC A	SBB A	XRA A	CMP A	RST 8	RST 18H	RST 28H	RST 38H	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

AUSTRALIA'S MOST UP-TO-DATE BUYER'S GUIDE FOR MICROCOMPUTERS

Month by month, every effort will be made to keep In Store up-to-date and accurate. And that means APC will always be happy to hear from its readers of any errors, and additions that seem worthy of inclusion.

LIST OF ABBREVIATIONS

A	– Assembler	K/B	– Keyboard
A/D	– Analog to Digital	M/A	– Macroassembler
B/W	– Black and White	N/A	– Not Available
C	– Cassette	N/P	– Numeric Pad
cps	– Characters per second	O/S	– Operating System
Doc	– Documentation	P/P	– Parallel Port
E	– Extensive	RAM	– Random Access Memory
Ed	– Editor	ROM	– Read Only Memory
Ex	– Extended	res	– Resolution
F/D	– Floppy Disc	S	– Software
H	– Hardware	S/P	– Serial Port
I	– Introductory	T/E	– Text Editor
I/O	– Input / Output	U	– Utility
int	– Interface	VDU	– Video Display Unit

All prices shown are exclusive of sales tax, except where indicated by an asterix.

Software items listed in *italics* are not included in the basic price of the equipment.

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
Apple II plus	Computerland (03) 62 5581 (02) 29 3753	16-48K RAM: 6502: colour VDU int.: 81/O slots: games paddles: option 5¼" F/D (116K) and 11 MB disc	O/S: BASIC: *Pascal:* games	E	$1395	280x192 high res colour graphics: Applesoft BASIC in 12K ROM
Century	Abacus Computer Store (03) 429 5844	C100, 48-64K RAM: Z80: 12" VDU: 2x5¼" F/D (2x143K): 112 cps printer: RS232 port: S100 bus: C200 includes 2xF/D (2x315K): hard disc: 4xRS232: 2xP/P	*COBOL: FORTRAN: BASIC*	I	C100–$4950: C200–$5400	Also available: C300
Challenger IP	Systems Automation (02) 439 6477	4-32K RAM: 6502: C int: 24x32 VDU int: RS232 port: option – dual 5¼" F/D (140K)	O/S: BASIC: A: games	I	$448	8K microsoft BASIC in ROM: expansion board available
Challenger 4	Systems Automation (02) 439 6477	8-48K RAM: 6502: colour 32x64 VDU int: RS232 port: P/P: option – 6502C microprocessor; dual 5¼" F/D (140K)	BASIC: *Pascal*	I	$871	BASIC in 8K ROM
Compucolor II	Anderson Digital Equipment (03) 543 2077	8-32K RAM: 8086: 13", 32x64 8 colour VDU: single 5¼" F/D (51K): RS232 port	ExBASIC(ROM):*A*	I	$2095	16K model, $2395: 32K $2695: maintenance manual available
Cromemco System 2, System Z2H, System 3	–	64-512K RAM: Z80A: System 2, dual 5¼" F/D (346K): System Z2H, also Winchester disc (11MB): System 3, 8" dual 1MB: S/P: P/P	CDOS: *BASIC: COBOL: FORTRAN: M/A: ExBASIC: Structured BASIC*	E	System2 $3990 System Z2H $9650 System 3, $6750	All Systems expandable to multi user (2-7 users) $2880–$8825
Exidy Sorcerer Model II	Dick Smith Electronics (02)888 3200	8-48K RAM: Z80: 30x64 VDU int.: RS232 Port: P/P: S100 bus:extra C int.	O/S: ExBASIC (ROM): *M/DOS: CP/M*	I	$1295*	High res graphics capability: 16K version $1395*: 32K version $1525*: 48K version $1655*: User programmable character set

IN STORE

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
HP-85	Hewlett Packard Australia (03)89 6351	16-32K RAM: N/A: 5",16x32 B/W VDU: C(200K): 64 cps printer: RS232 port: 4 x P/P	BASIC	S	$3550	Full dot matrix graphics: N/P: compact portable unit
IPS-100	Microprocessor Applications (03) 754 5108	32-896K RAM: 8085: 2 RS232 ports: S100 bus: dual 5¼" F/D (630K)	O/S: Ex BASIC: Ed: A: *CP/M: CBASIC: FORTRAN: COBOL*	E	$3750	
Microengine	Daneva Control (03)598 9207 Abacus Computer Store for complete system (03) 429 5844	64K RAM: MCP 1600: 2 x RS232 ports: 2xP/P: Options – dual 5¼" F/D (Single or dble density); 8" F/D (single or dble density)	BASIC: Pascal: File Manager: U	E	$2995	Also available as board
North Star Horizon	Melbourne's Byte Shop (03) 568 4022	32-64K RAM: Z80A: 5¼" F/D (170K): 2xS/P: 1P/P: optional – VDU ($1350); Quad density F/D	DOS: BASIC: *COBOL: FORTRAN: Pascal: CP/M: M/A*	E	$2695	
Pet 2001	Hanimex (02) 938 0400	8-32K RAM: 6502: C: 9" 25x40 B/W VDU: extra C Int: IEEE488 port	O/S: BASIC: *A*	I	8K$1199 16K$1859 32K$2249	Options – dual 5¼" F/D (353K), $2329: $109 for disc operating ROM
Sord M100 ACE III	Alliance Digital Corporation (02) 436 1600 Abacus (03) 429 5844	48K RAM: Z80: 24x64, 12" VDU: RS232 ports: 2x5¼" F/D (2x143K): S100 bus: 2 octave speaker: A/D Conv.: option–8 colour graphic controller ($1450)	O/S: Ex*BASIC FORTRAN*	I	$4500	M100 ACE IV – 8 colour graphics controller incl.
Sord M223	Alliance Digital Corporation (02) 436 1600 Abacus (03) 429 5844	64K RAM: Z80: 12" 24x80 VDU: 2xRS232 port: S100 bus: 5¼" F/D (350K)	O/S: *ExBASIC: FORTRAN: COBOL*	I	$7500	
TRS-80 Level 1	Tandy Electronics (02) 638 6633	4-16K RAM: Z80: C: 12" 16x64 B/W VDU	BASIC: Games: A	I	$699*	BASIC in 4K ROM: upgradable to Level 2
TRS-80 Level 2	Tandy Electronics (02) 638 6633	4-48K RAM: Z80: C:12", 16x64 B/W VDU: RS232 port: P/P	BASIC: *M/A: FORTRAN: COBOL*		$879*	16K machine includes N/P: 4-16K upgrade $320*: ($250* without N/P): max. config. $1169*: option – single 5¼" F/D (78K), (max of 4)
Vector Graphics System B	AJ & JW Dicker (02) 524 5639	64K RAM: Z80: Dual 5¼" F/D (630K): 12", 24x80 B/W VDU: S/P: 2xP/P	DOS: BASIC: A: CP/M: Ed	E	$6350	Graphics and numeric pad.
Versatile 4	Microprocessor Applications (03) 754 5108	32-56K RAM: 8085: 9", 24x80 B/W VDU: dual 5¼" F/D (630K): S100 bus: 2xRS232	MBASIC: MDOS (including T/E and A): *Version 4 MDOS AND BASIC: CP/M*	E	$5692	

SINGLE BOARDS

Acorn	Cottage Computers (03) 481 1975	1-8K RAM: 6502: EPROM socket: Hex K/B: C int: 8 digit LED display: up to 16 ports: options – Euro-card 64 way connector; VDU card; Full K/B card	½K monitor: *BASIC*	S& H	System1 $285: SystemII $1224: SystemIII (incl.a5¼" F/D), $2763	Universal interface card available.
Aim 65	Dwell Pty Ltd (02) 487 3111	1-4K RAM: 6502: 8K ROM: full K/B: 20 character LED display: 20 character thermal printer: Cx2 int: 1 P/P	8K monitor in ROM: *A: BASIC*	E	$525*	Case available $75*
SBC100	Microtrix (03) 718 2581	1K RAM: Z80: 8K ROM: S100 bus: 1S/P: 1P/P	1K monitor: DOS in ROM	E	$299	Also available assembled $374
Superboard	Systems Automation (02) 439 6477	4-32K RAM: 6502: 10K ROM: full K/B: 24x32 VDU Int: C int: options–RS232; dual 5¼" F/D (140K)	BASIC: games	I	$360	BASIC in 8K ROM

SOUND FOR THE TRS-80

by Roxton Baker

This machine language routine provides an easy method for adding sound to programs. Written to be accessed from TRS-80 Level II or Disc BASIC, it produces a square wave output to the AUX output line (large grey plug). Although this output is strong enough to drive an earplug or headphones, better effects are achieved by placing the cassette in the record mode and connecting the AUX plug to the cassette unit by the AUX jack and a speaker to the ear phone jack. The best sound of all results from connecting the cassette output to a nice, powerful hifi system. Phaser blasts through this can really shake up the troops.

Soundex can be POKEd anywhere in memory and is completely relocatable (it can even be embedded in a REM statement at the beginning of a program).

If high memory is to be used to store the machine language drive, MEMORY SIZE must be set at 32695 for 16K machines, 49079 for 32K, and 65463 for 48K. If TRSDOS is being used, an additional 51 bytes should be reserved.

After CLOADing or keying in the BASIC program and running it, the Soundex drive can be accessed via the USR command.

Tone duration is specified by using a negative argument in the USR command:

X = USR (dd)

The variable dd can have any value from –1 to –32767, representing tone durations from 0 to 2 seconds.

To generate a tone and specify its pitch, the USR command should have a positive argument. Any value from 1 to 32676 can be used but, practically, only those between 1 and 100 are useful.

Once set, the tone duration will control all subsequent tones until a new duration is specified.

Another option available is that of the 'voice' of the tone. This is changed by POKEing any of the following values into memory location 16672:
1 5 6 17 18 22 25 86 90 102
As an example, near the beginning of a program the user might write:

POKE 16672, 18

All tones generated after this would be fairly smooth, sine wave approximations.

Under Disc BASIC, the option exists to disable interrupts with CMD"T". If this is not done, the tones will have a staccato quality.

Soundex achieves its relocatability by using six bytes in low memory (reserved RAM) as temporary data storage. These positions, hex 411C to 4120, are not believed to be used by Level II or any DOS.

```
10 REM        >>>>>>  SOUNDEX  <<<<<<
20 REM    BY ROXTON BAKER
30 CLS
40 DEFINT J,K,L
50 INPUT"WHAT MEM SIZE DID YOU SET ";DS
60 REM  SOUNDEX WILL START AT MEM SIZE PLUS ONE
70 MS=DS
80 IF MS>32767 THEN MS=MS-65536
90 REM  POKE THE 72 BYTES IN
100 FOR J=1 TO 72
110 READ K
120 POKE MS+J,K
130 NEXT J
140 REM  NOW SEE IF WE'RE IN LEVEL II OR DISK BASIC
150 IF PEEK(16433) = 0 THEN 200
160 REM       MUST BE DISK
170 CMD"T"
180 DEFUSR=MS+1
190 GOTO 290
200 REM     IT'S LEVEL II
210 N=DS+1
220 REM  CHANGE N FROM DEC TO HEX FOR USR ENTRY POINT POKE
230 FOR I=1 TO 4
240 D(I)=INT(N/16[(4-I))
250 N=N-D(I)*16[(4-I)
260 NEXT I
270 POKE 16527, 16*D(1)+D(2)
280 POKE 16526, 16*D(3)+D(4)
290 REM  CHOOSE SQUARE WAVE FOR DEMO
300 POKE 16672,102
310 REM  THIS IS THE BIG DEMO . . .
320 X=USR(-100)
330 FORL=65TO1STEP-1:X=USR(L):NEXT
340 FORL=1TO70STEP2:X=USR(L):NEXT
350 REM
360 REM  THESE FIVE DATA STATEMENTS CONTAIN THE RELOCATABLE
370 REM  CODE OF SOUNDEX:
380 REM
390 DATA 205,127,10,203,124,40,4,34,28,65,201,34,30,65
400 DATA 219,255,31,31,31,47,230,248,95,58,57,65,254,4,32,2
410 DATA 171,95,58,32,65,87,237,75,28,65,43,124,181,40,6
420 DATA 221,227,221,227,24,12,42,30,65,122,7,7,87,230,3,179
430 DATA 211,255,3,120,177,32,228,123,211,255,201

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

The advantage of POKEing the routine into space normally reserved for a REM statement is that the code becomes part of the BASIC program and no memory protection is required. Furthermore, the BASIC Soundex program can be deleted, while permanently retaining the machine language driver which is then stored with the BASIC program under the one CSAVE command.

To ensure the REM statement containing the machine language routine always occupies the same position in memory, it should be line number Ø . This way the REM statement cannot be proceeded by any other lines. The REM statement must contain at least as many dummy characters as there are bytes in the Soundex driver (i.e. 72 bytes). So line number Ø should look like: Ø REM DDDDDDDDDD (continue for 72 D's) DDD.

The address of the first line in a BASIC program is stored in memory locations 16548 and 16549 and the first dummy character is the fifth byte in the first line. So the BASIC Soundex program must begin POKEing the machine language driver at the memory position given by PEEK (16548) + 256*(16549) + 5. This address should also be defined as the USR entry point using DEFUSR in Disc BASIC or its POKE equivalent in Level II BASIC. A word of warning: don't try to renumber a program which contains the machine language driver in a REM statement as some renumbering utilities will confuse the machine codes for BASIC statements.

The program above should be used to POKE the machine language driver into the beginning of a Disc BASIC program.

To define the USR entry point in Level II BASIC, line 3 should be changed to:
3 POKE 16526, S-INT(S/256)*256: POKE 16527,INT(S/256)

After running the program once delete lines 4 to 12, retaining your program and the Soundex driver in the REM statements.

```
1REM****************** (72 *'s total) *******...
2 S=PEEK(16548) + 256*PEEK(16549) + 5
3 DEFUSR = S
4 FOR J=1 TO 72
5 READ X
6 POKE S+J-1,X
7 NEXT J
8 DATA ...........
9 DATA ...........
10 DATA ..........
11 DATA ..........
12 DATA ..........
13 etc. user's program
      from here on.
```

Submitting programs to APC

Having written and thoroughly tested your original program (be it an application, game, or useful subroutine) send it to us along with a suitable explanation. In order of preference we would like your programs submitted as a clear, dark listing on plain paper; on cassette or disc; clearly, accurately typed; or, clearly, accurately handwritten.

We pay the sender of any listing published at least $20, and often much more, depending on the size and quality of the contribution. If the program is too large or complex for the "Programs" section we will sometimes publish it as a feature in the magazine.

For the sake of balance, how about some of the owners of less popular machines pitching in as well? Post your submissions to APC Programs, PO Box 115, Carlton, Victoria, 3053. We look forward to hearing from you.

APPLE WORMS

by Ray West, freelance programmer

TAPEWORMS: A KEYBOARD VIDEO GAME FOR THE APPLE

'Tapeworms' is a game for two players which uses the keyboard interactively. Each player has four allocated keys, which are identified by the keyboard PEEK function. Two shape tables are loaded by the program, and these give each 'worm' a different appearance. To improve the appearance of the display, the rotation feature of the shape table is used so that the direction of movement of the worm is matched by the rotation of the shape. A game ends either when a player crosses the rectangular border of the playing area, or collides with a previously plotted shape; the collision counter provides a way of checking for this event. For a detailed explanation of the listing, now read on...

Lines 510-640 are the main program control. There are essentially six subroutines which are called.

SUB 20000. This sets up a shape table of two shapes. Line 20000 sets up pointers in locations 232 and 233, the low and high bytes respectively. Since 117*256+48=30000, the Apple expects the shapes to start at 30000. This works for a 48K or 32K machine. Line 20001 tells the machine there are two shapes in the table, and lines 20005 and 20010 give their addresses, offset from 30000. So shape 1 begins at 30000+256*0+159=30159, for example. The two shapes are 'A' and 'V', and were used because they happened to be available. If you don't like them, try adjusting the table!

SUB 10000. This prints the title page onto the screen, enabling one of three playing speeds to be selected. In addition, variables are stored just after the program; line 10010 ensures that the coordinates and directions of each 'worm' are stored where retrieval time is minimised. Random start points and directions are generated for each player; in line 10220 they are checked to avoid starting too close to each other. Directions 1 to 4 are converted into the relevant keystroke equivalent for each player.

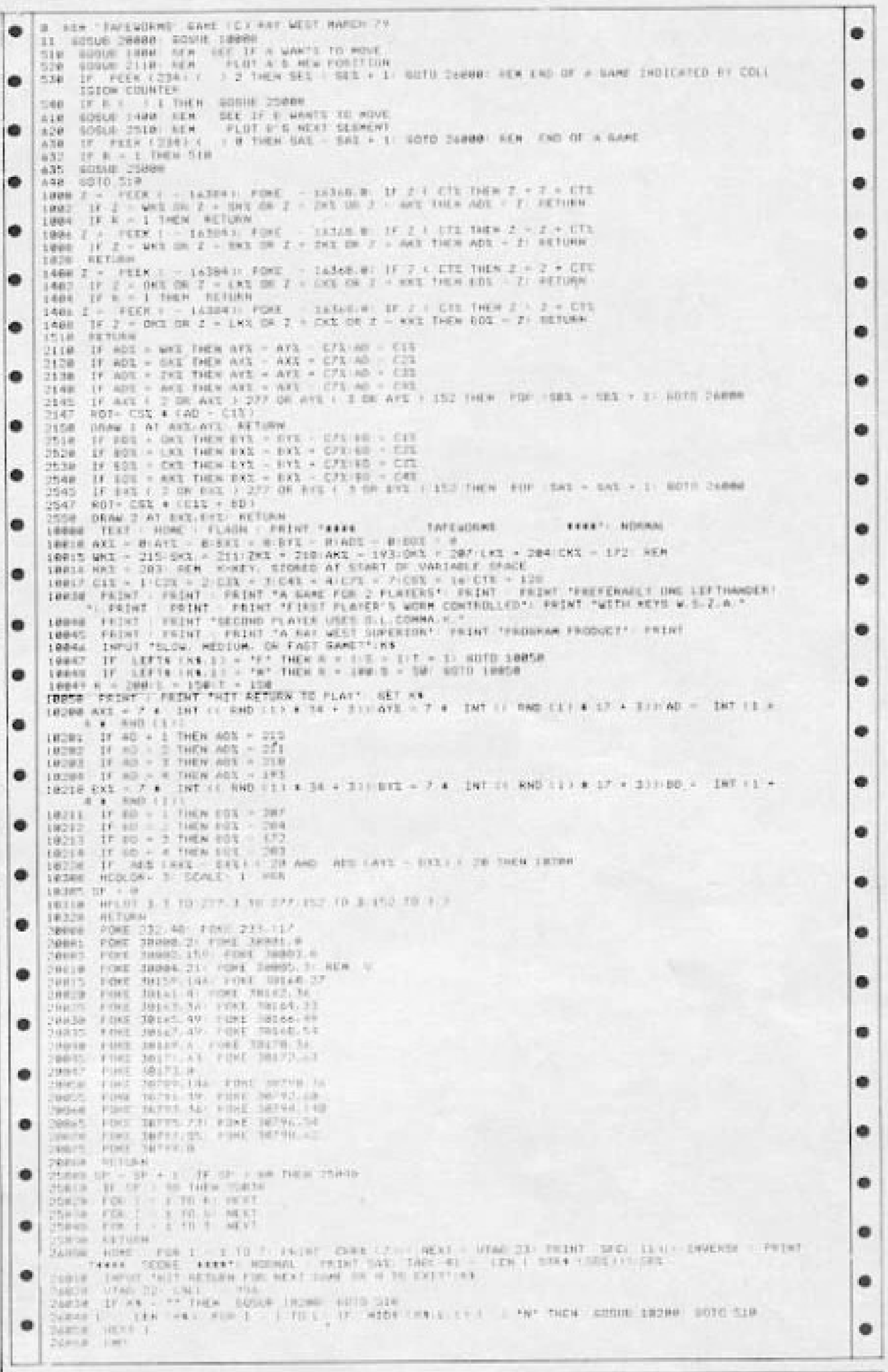
SUB 1000 & SUB 1400. This symmetrical pair of routines reads the keyboard. The point of the last statements of lines 1000 and 1400 is that the Apple seems sometimes to admit a low ASCII value. If on A's turn his part of the keyboard registers an input, its ASCII value is saved; and similarly on B's turn. In a fast game, only one peek at the keyboard is allowed.

SUB 2110 & SUB 2510. The x or y coordinate is incremented/decremented as required, and the direction indicator AD or BD set to correspond. Lines 2145 and 2545 test the new plot. If it is an acceptable move, the other person's score is increased by 1 and exit to the end-of-game routine occurs. The POP instruction removes the subroutine's return address from the stack: were this instruction omitted, after about 24 games the stack would fill up and an OUT OF MEMORY message appear.

The formulae for ROT need to introduce multiples of 16, for which the values differ for the shapes plotted, so that lines 2147 and 2547 use different calculations. The direction is coded as 1 for north, 2 for east, and so on.

SUB 25000. This is entered only in a slow or medium speed game. It uses simple delay loops, which, however, have diminishing effect as the game proceeds. So the tempo accelerates towards the end.

SUB 26000. This routine displays the aggregate scores to date, the player sitting on the left having his score shown at the left of the screen and vice versa. The set of games can be terminated in order to change speed, or start afresh, by entering 'N'. Since some characters may remain in the buffer, line 26040 checks for the presence of an 'N' within it. If the set of games continues, line 26040 loops back to reset new starting positions and directions, before returning to the program's main control.



```
Ø  REM 'TAPEWORMS' GAME (C) RAY WEST MARCH 79
1Ø  GOSUB 2ØØØØ: GOSUB 1ØØØØ
51Ø  GOSUB 1ØØØ: REM  SEE IF A WANTS TO MOVE
52Ø  GOSUB 211Ø: REM    PLOT A'S NEW POSITION
53Ø  IF  PEEK (234) <  > 2 THEN SB% = SB% + 1: GOTO 26ØØØ: REM END OF A GAME INDICATED BY COLL
     ISION COUNTER
54Ø  IF R >  > 1 THEN  GOSUB 25ØØØ
61Ø  GOSUB 14ØØ: REM   SEE IF B WANTS TO MOVE
62Ø  GOSUB 251Ø: REM    PLOT B'S NEXT SEGMENT
63Ø  IF  PEEK (234) <  > Ø THEN SA% = SA% + 1: GOTO 26ØØØ: REM  END OF A GAME
632  IF R = 1 THEN 51Ø
635  GOSUB 25ØØØ
64Ø  GOTO 51Ø
1ØØØ Z =  PEEK ( - 16384): POKE  - 16368,Ø: IF Z < CT% THEN Z = Z + CT%
1ØØ2  IF Z = WK% OR Z = SK% OR Z = ZK% OR Z = AK% THEN AD% = Z: RETURN
1ØØ4  IF R = 1 THEN  RETURN
1ØØ6 Z =  PEEK ( - 16384): POKE  - 16368,Ø: IF Z < CT% THEN Z = Z + CT%
1ØØ8  IF Z = WK% OR Z = SK% OR Z = ZK% OR Z = AK% THEN AD% = Z: RETURN
1Ø2Ø  RETURN
14ØØ Z =  PEEK ( - 16384): POKE  - 16368,Ø: IF Z < CT% THEN Z = Z + CT%
14Ø2  IF Z = OK% OR Z = LK% OR Z = CK% OR Z = KK% THEN BD% = Z: RETURN
14Ø4  IF R = 1 THEN  RETURN
14Ø6 Z =  PEEK ( - 16384): POKE  - 16368,Ø: IF Z < CT% THEN Z = Z + CT%
14Ø8  IF Z = OK% OR Z = LK% OR Z = CK% OR Z = KK% THEN BD% = Z: RETURN
151Ø  RETURN
211Ø  IF AD% = WK% THEN AY% = AY% - C7%:AD = C1%
212Ø  IF AD% = SK% THEN AX% = AX% + C7%:AD = C2%
213Ø  IF AD% = ZK% THEN AY% = AY% + C7%:AD = C3%
214Ø  IF AD% = AK% THEN AX% = AX% - C7%:AD = C4%
2145  IF AX% < 2 OR AX% > 277 OR AY% < 3 OR AY% > 152 THEN  POP :SB% = SB% + 1: GOTO 26ØØØ
2147  ROT= C5% * (AD - C1%)
215Ø  DRAW 1 AT AX%,AY%: RETURN
251Ø  IF BD% = OK% THEN BY% = BY% - C7%:BD = C1%
252Ø  IF BD% = LK% THEN BX% = BX% + C7%:BD = C2%
253Ø  IF BD% = CK% THEN BY% = BY% + C7%:BD = C3%
254Ø  IF BD% = KK% THEN BX% = BX% - C7%:BD = C4%
2545  IF BX% < 2 OR BX% > 277 OR BY% < 3 OR BY% > 152 THEN  POP :SA% = SA% + 1: GOTO 26ØØØ
2547  ROT= C5% * (C1% + BD)
255Ø  DRAW 2 AT BX%,BY%: RETURN
1ØØØØ  TEXT : HOME : FLASH : PRINT "****          TAPEWORMS          ****": NORMAL
1ØØ1Ø AX% = Ø:AY% = Ø:BX% = Ø:BY% = Ø:AD% = Ø:BD% = Ø
1ØØ15 WK% = 215:SK% = 211:ZK% = 218:AK% = 193:OK% = 207:LK% = 204:CK% = 172: REM
1ØØ16 KK% = 203: REM  K-KEY  STORED AT START OF VARIABLE SPACE
1ØØ17 C1% = 1:C2% = 2:C3% = 3:C4% = 4:C7% = 7:C5% = 16:CT% = 128
1ØØ3Ø  PRINT : PRINT : PRINT "A GAME FOR 2 PLAYERS": PRINT : PRINT "PREFERABLY ONE LEFTHANDED!
     ": PRINT : PRINT : PRINT "FIRST PLAYER'S WORM CONTROLLED": PRINT "WITH KEYS W,S,Z,A."
1ØØ4Ø  PRINT : PRINT "SECOND PLAYER USES O,L,COMMA,K."
1ØØ45  PRINT : PRINT : PRINT "A RAY WEST SUPERIOR": PRINT "PROGRAM PRODUCT": PRINT
1ØØ46  INPUT "SLOW, MEDIUM, OR FAST GAME?";K$
1ØØ47  IF  LEFT$ (K$,1) = "F" THEN R = 1:S = 1:T = 1: GOTO 1ØØ5Ø
1ØØ48  IF  LEFT$ (K$,1) = "M" THEN R = 1ØØ:S = 5Ø: GOTO 1ØØ5Ø
1ØØ49 R = 2ØØ:S = 15Ø:T = 15Ø
1ØØ5Ø  PRINT : PRINT "HIT RETURN TO PLAY": GET K$
1Ø2ØØ AX% = 7 *  INT ( RND (1) * 34 + 3):AY% = 7 *  INT ( RND (1) * 17 + 3):AD =  INT (1 +
     4 *  RND (1))
1Ø2Ø1  IF AD = 1 THEN AD% = 215
1Ø2Ø2  IF AD = 2 THEN AD% = 211
1Ø2Ø3  IF AD = 3 THEN AD% = 218
1Ø2Ø4  IF AD = 4 THEN AD% = 193
1Ø21Ø BX% = 7 *  INT ( RND (1) * 34 + 3):BY% = 7 *  INT ( RND (1) * 17 + 3):BD =  INT (1 +
     4 *  RND (1))
1Ø211  IF BD = 1 THEN BD% = 2Ø7
1Ø212  IF BD = 2 THEN BD% = 2Ø4
1Ø213  IF BD = 3 THEN BD% = 172
1Ø214  IF BD = 4 THEN BD% = 2Ø3
1Ø22Ø  IF  ABS (AX% - BX%) < 28 AND  ABS (AY% - BY%) < 28 THEN 1Ø2ØØ
1Ø3ØØ  HCOLOR= 3: SCALE= 1: HGR
1Ø3Ø5 SP = Ø
1Ø31Ø  HPLOT 3,3 TO 277,3 TO 277,152 TO 3,152 TO 3,3
1Ø32Ø  RETURN
2ØØØØ  POKE 232,4Ø: POKE 233,117
2ØØØ1  POKE 3ØØØØ,2: POKE 3ØØØ1,Ø
2ØØØ5  POKE 3ØØØ2,15Ø: POKE 3ØØØ3,Ø
2ØØ1Ø  POKE 3ØØØ4,21: POKE 3ØØØ5,3: REM [illegible]
2ØØ15  POKE 3Ø159,146: POKE 3Ø16Ø,27
2ØØ2Ø  POKE 3Ø161,41: POKE 3Ø162,36
2ØØ25  POKE 3Ø163,54: POKE 3Ø164,33
2ØØ3Ø  POKE 3Ø165,99: POKE 3Ø166,99
2ØØ35  POKE 3Ø167,49: POKE 3Ø168,54
2ØØ4Ø  POKE 3Ø169,6: POKE 3Ø17Ø,36
2ØØ45  POKE 3Ø171,45: POKE 3Ø172,63
2ØØ47  POKE 3Ø173,Ø
2ØØ5Ø  POKE 3Ø789,146: POKE 3Ø79Ø,36
2ØØ55  POKE 3Ø791,39: POKE 3Ø792,6Ø
2ØØ6Ø  POKE 3Ø793,54: POKE 3Ø794,18Ø
2ØØ65  POKE 3Ø795,73: POKE 3Ø796,54
2ØØ7Ø  POKE 3Ø797,35: POKE 3Ø798,42
2ØØ75  POKE 3Ø799,Ø
2ØØ8Ø  RETURN
25ØØØ SP = SP + 1: IF SP > 4Ø THEN 25Ø4Ø
25Ø1Ø  IF SP > 8Ø THEN 25Ø3Ø
25Ø2Ø  FOR I = 1 TO R: NEXT
25Ø3Ø  FOR I = 1 TO S: NEXT
25Ø4Ø  FOR I = 1 TO T: NEXT
25Ø5Ø  RETURN
26ØØØ  HOME : FOR I = 1 TO 7: PRINT  CHR$ (7);: NEXT : VTAB 23: PRINT  SPC( 11);: INVERSE : PRINT
     "****  SCORE  ****": NORMAL : PRINT SA%; TAB( 41 -  LEN ( STR$ (SB%)));SB%
26Ø1Ø  INPUT "HIT RETURN FOR NEXT GAME OR N TO EXIT";K$
26Ø2Ø  VTAB 22: CALL  - 958
26Ø3Ø  IF K$ = "" THEN  GOSUB 1Ø2ØØ: GOTO 51Ø
26Ø4Ø L =  LEN (K$): FOR I = 1 TO L: IF  MID$ (K$,I,1) <  > "N" THEN  GOSUB 1Ø2ØØ: GOTO 51Ø
26Ø5Ø  NEXT I
26Ø6Ø  END
```

VARIABLE MAP

By David Eyles

The following routine for a TRS-80 can be used as a debugging aid in the development of programs. When called upon, it displays all single letter variables and their values.

Integer, single precision, double precision or string variables will be displayed depending on the definition character following the variable "Z", (using the listing shown below, single precision variables would be displayed). To change this character use EDIT Ø.

The routine can be accessed from the command mode or program statements by GOSUB Ø or GOTO Ø. If accessed from the command mode, an RG error will result after execution of the routine leaving the computer back in the command mode.

Take care when keying in the routine that no spaces are inserted or deleted — results can be disastrous! Each time RUN is entered the routine will be executed, so to run the program properly use RUN line number.

```
ØZX=1Ø:ZW=17194:POKE16396,
23:CLS:FORZV=65TO9Ø:POKE
ZX,ZV:?@ZX,CHR$(ZV);" = ";
Z!;:ZX=ZX+32:NEXTZV:POKE
16396,2Ø1:RETURN
```

ACRONYMS

Submitted by Richard Forsyth

This program was designed by Richard Forsyth to convert any word (up to 25 characters) into any acronym. Suitable catch phrases are generated using a buzzword directory. In order to make the program more useful, an entire selling blurb (containing more vital buzzwords) is produced. A useful feature of this program is that you can request any number of alternative blurbs for each acronym entered. Here is an example:

```
What is BRUCE?
---------------------

   BRUCE is a dramatic new development. It stands for Binary Record Unbundled Centralized Encoding a totally
new concept in Encoding.
BRUCE is not only Reactive
but also Normally Acoustic.

   It has the following attractive features:

      **Structure
      **Multiprogramming
      **Relational Formulation
      **Access Switching
      **Standardization

BRUCE is Forward
because of its Computational Facility
Y-axis.
It also has Networking Hash-coding
which makes it Modular.
It is in a class of its own when it comes
to the Layout Servo-mechanism,
and above all it is Environmental.
   Can you afford to do without
BRUCE (Binary Record Unbundled
Centralized Encoding)?
```

We had to remove the contents of line 9000 as it comprised eight variable-length anglo-saxon words. These words were incorporated to enable the program to display the phrase in line 1560 (also deleted) to the rude user. You will, no doubt, want to create your own vocabulary.
Each record comprises two words W$ and T$ where W$ is the word and T$ is the description A,AA,N or NN. A describes an adjective, AA is an adverb, N a general noun and NN a specific noun.
The first few words in Richard's vocabulary are listed at the end of the program to guide you. You will notice that it is alphabetic on the initial letter only.

```
1       REM* PROGRAM TO MAKE UP CATCH
        PHRASES FROM BUZZ-WORDS:
2       REM* BY RICHARD FORSYTH, 1978.
10      DIM W$(1500),W(3),G(3,27)
20      DIM P$(25),P(25),S$(8)
25      MARGIN 80
50      REM* TABLE ADDRESSING FUNCTION:
55      DEF FNW(I,J)=366*I+J
60      RESTORE
75      RANDOMIZE
77      REM* READ IN SWEARWORDS:
80      READ S$(Z) FOR Z=1 TO 8
99
100     REM* MAIN LINE:
110     GOSUB 1000        ' INITIALIZE
115     GOSUB 1500        ' USER INPUT
120     IF LEN(I$)=0 OR B<1 THEN 160
122     PRINT
125     LET B=B-1
130     GOSUB 2000        ' GENERATE AND STORE
140     GOSUB 3000        ' EXPRESS
150     IF B>0 THEN 125
155     PRINT \ GOTO 115
160     PRINT "VOCABULARY LISTING (Y=YES)";
165     INPUT Y$
170     IF Y$="Y" THEN GOSUB 4000
175     GOTO 9999
999
1000    REM* DATA INPUT ROUTINE:
1001    FOR J=0 TO 3
1002    LET W(J)=0
1005    FOR I=0 TO 27
1010    LET G(J,I)=0
1012    NEXT I \ NEXT J
1015    LET H=0
1020    OPEN "WORD.FIL" FOR INPUT AS FILE #1
1022    IF END #1 THEN 1180
1023    INPUT #1, W$,T$
```

```
1024    IF H>1200 THEN 1180
1025    IF W$="**" THEN 1180
1030    LET T=0
1032    LET A=ASCII(W$)-64
1035    IF T$="A" THEN T=1
1040    IF T$="AA" THEN T=2
1045    IF T$="N" THEN T=3
1050    IF T$="NN" THEN T=4
1055    IF T>0 THEN 1070
1060    PRINT "INPUT ERROR: ",W$;" , ";T$
1065    GOTO 1022
1070    LET W=T-1
1075    LET W(W)=W(W)+1
1077    LET W$(FNW(W,W(W)))=W$+""
1080    IF G(W,A)=0 THEN G(W,A)=W(W)
1085    REM* NOTES START OF EACH LETTER.
1088    LET H=H+1 \ GOTO 1022
1100    REM* END OF FILE:
1180    CLOSE 1
1182    PRINT H;"WORDS."
1185    FOR J=0 TO 3
1188    LET G(J,27)=W(J)          ' LAST
1190    NEXT J
1195    PRINT W$
1199    RETURN
1499
1500    REM* USER INPUT ROUTINE:
1505    PRINT "GIVE YOUR WORD:"
1510    INPUT I$
1515    IF LEN(I$)>25 THEN PRINT "TOO LONG!"
        \ GOTO 1510
1520    FOR Z=1 TO 8
1522    IF INSTR(1,I$,S$(Z))>0 THEN 1555
1525    NEXT Z
1530    PRINT "HOW MANY BLURBS (0 TO HALT)";
1535    INPUT B
```

LOOKY VIDEO

INDEPENDANT IMPORTER/MANUFACTURER

OHIO SCIENTIFIC COMPUTERS

HARDWARE:

1. MEMORY EXPANSION BOARD, 8K. The Double sided Plated through the Board has 8K of wired sockets with driver and buffer I.C.'s and R.F. filters. Add 2114's as you require RAM . KIT . $120.00 Fully assembled and tested, extra $15.00
2. 4K RAM Memory Expansion Chips, 2114, 450nS. . . . . $ 52.00
3. R.F. Modulator, 5 to 9 Volt, all channels . . . . $ 19.95
4. SUPERBOARD II Cover/VDU Stand . . . . $ 18.95
5. SUPERBOARD II COMPUTER 8K ROM, 4K RAM $389.00
6. CHALLENGER C1P COMPUTER 8K ROM, 4K RAM $489.00
7. CHALLENGER C4P COMPUTER, Sound/Colour $946.20

SOFTWARE:
We regret, PRICE RISE, due to shipping costs. All previous (54) advertised Cassette Software, up $2.00. Instructions (14 sets), are still the same price. O1 is now replaced by an updated, and enlarged C1 at $2.95

NEW PROGRAMMES
for OHIO – SOFTWARE:

UTILITIES:

U.15. Dissassembler – 4K	$10.95
U.16. Filename, for cassettes . .	$ 8.95
U.17. Trace	$12.95
U.18. Packer – (K saver)	$14.95
U.19.Cursor – C2/4	$13.95

INSTRUCTIONS:

I.11. Dissassembled ROM, with Comments	$12.95
I.12. 32x64 Character Display . .	$12.95
I.13. Wp 6502 Word Processor Book	$ 3.95
I.14. Dissassembled ROM, with Inbuilt References	$13.95
I.15. Sound – S II/CIP	$ 6.95
I.16. G.T. Conversion, X2	$ 2.95
I.17. Saving Data on Tape	$ 6.95
I.18. CIP Tape Control	$ 6.95
I.19. CIP Beeper.	$ 5.95
I.20. CIP/Base 2 Printer	$ 6.95
I.21. Morse Code Converter and Tape	$16.95
I.22. 32x64 Character – 100% Hardware.	$13.95

BUSINESS:

B.4. Savings and Loan Package .	$14.95
D.2. Flashboard, your advert . .	$10.95

GAMES:

G.25. Alien Invaders/Space Invaders	$10.95
G.26. Orbital Lander	$11.95
G.27. Escape from Mars, 2 tapes	$18.95
G.28. Death Ship, 2 tapes	$18.95
G.29. Startrek	$ 7.95
G.30. Air – Sea Battle	$ 7.95

BOOKS/MAGAZINES/CATALOGUE

T.1. The First Book of OSI . . .	$19.95
T.2. Aardvark Journal (6 issue) .	$12.95
T.3. Basic Handbook, David Lien .	$15.95
T.4. Personal Computing Monthly	$ 2.20
C.1. Catalogue describes all software and hints and free progs	$ 2.95

Postage:

1 or 2	$1.00
3 – 5	$1.50
6 – 9	$2.00
10 or more	$2.50

All prices are subject to change without notice.

All prices INCLUDE Sales Tax.

MAIL ORDER: Check or Bankcard Looky Video, P.O. Box 347, Richmond, 3121. Victoria.
Shop: 418 Bridge Road, Richmond, Phone (03) 429 5674

bankcard welcome here

```
1550    RETURN
1555    REM* DIRTY WORD:
1560    PRINT
1570    REM* KEEP IT CLEAN!
1575    STOP
1999
2000    REM* PHRASE GENERATOR ROUTINE:
2005    LET L=LEN(I$)
2010    MAT P=ZER \ LET P$=""
2015    REM* FIRST WORD FIRST:
2020    LET T=INT(RND+1.5)        ' ADJ/ADV
2022    LET P(1)=T
2025    LET A=ASCII(I$)-64
2030    GOSUB 2200        ' PICK WORD OF TYPE T
2032    LET P$(1)=W$+""
2035    LET P$=P$+W$+" "
2040    FOR J=2 TO L
2045    LET A=ASCII(MID$(I$,J,1))-64
2050    IF J=2 AND P(1)=2 THEN T=1 \ GOTO 2065
2055    LET R=J/L
2060    IF R>=RND THEN T=INT(3.5+RND) ELSE T=1
2065    LET P(J)=T        ' TYPE OF JTH WORD
2070    GOSUB 2200        ' PICK WORD
2075    LET P$(J)=W$+"" \ P$=P$+W$+" "
2077    NEXT J
2080    LET P$=LEFT$(P$,LEN(P$)-1)
2088    RETURN
2199
2200    REM* WORD SELCTION ROUTINE:
2201    LET W=T-1
2202    IF A=0 THEN 2300
2205    IF A<1 OR A>26 THEN W$="" \ RETURN
2210    LET G0=G(W,A)    ' START OF THAT LETTER
2220    LET G=G(W,A+1)-G0
2255    IF G<=0 THEN W$="!?" \ RETURN
2260    LET I=INT(G*RND)
2265    LET W$=W$(FNW(W,G0+I))+""
2275    RETURN
2300    REM* NO CHOSEN INITIAL:
2305    LET I=INT(RND*W(W)+1)
2310    LET W$=W$(FNW(W,I))+""
2325    RETURN
2999
3000    REM* OUTPUT ROUTINE:
3003    LET T1=0 \ A=0
3005    PRINT \ PRINT "What is ";I$;"?"
3010    PRINT "-"; FOR Z=1 TO L+9
3015    PRINT \ PRINT
3020    PRINT TAB(4);I$;" is a dramatic new development. It stands for"
3022    PRINT P$
3025    IF P(L)=3 THEN 3040
3030    PRINT "a completely new kind of ";P$(L);"."
3035    GOTO 3045
3040    PRINT "a totally new concept in ";P$(L);"."
3045    PRINT I$;" is not only";
3048    GOSUB 3500        ' ADJ PHRASE
3050    PRINT \ PRINT "but also";
3052    GOSUB 3500 \ PRINT "."
3055    PRINT TAB(4);"It has the following attractive features:"
3060    PRINT \ LET N=INT(RND*4.4+2)
3065    FOR Z=1 TO N
3070    PRINT TAB(8);"** ";
3075    GOSUB 3600        ' NOUN PHRASE
3077    PRINT \ NEXT Z
3080    PRINT
3085    PRINT I$;" is";
3088    GOSUB 3500 \ PRINT
3090    PRINT "because of its";
3095    LET T1=4 \ GOSUB 3600
3099    PRINT "."
3100    PRINT "It also has";
3110    LET T1=3 \ GOSUB 3600
3115    PRINT
3120    PRINT "which makes it";
3125    GOSUB 3500 \ PRINT "."
3130    LET T1=INT(RND+3.33)
3132    PRINT "It is in a class of its own when it comes"
3133    PRINT "to";
3135    IF T1=4 THEN PRINT " the";
3136    GOSUB 3600 \ PRINT ","
3140    PRINT "and above all it is";
3145    GOSUB 3500 \ PRINT "."
3150    PRINT TAB(4);"Can you afford to do without ";I$
3155    PRINT "(";P$;")?"
3160    PRINT
3225    RETURN
```

```
3499
3500    REM* ADJECTIVAL PHRASE ROUTINE:
3505    IF RND>0.44 THEN 3520
3510    LET T=2 ' ADVB
3512    GOSUB 2200
3515    PRINT " ";W$;
3520    LET T=1
3522    GOSUB 2200
3525    PRINT " ";W$;
3530    IF RND<0.8 THEN RETURN
3535    PRINT " and";
3540    GOTO 3522        ' REPEAT
3599
3600    REM* NOUN PHRASE ROUTINE:
3605    IF RND>0.5 THEN GOSUB 3500
3610    LET T=INT(RND+3.33)
3615    GOSUB 2200
3620    PRINT " ";W$;
3625    IF RND>0.6 THEN 3610
3630    REM* LOOP AD LIB.
3635    IF T1=0 THEN RETURN
3640    LET T=T1 \ GOSUB 2200
3645    PRINT " ";W$;
3650    RETURN
3999
4000    REM* WORDLIST ROUTINE:
4005    PRINT
4010    FOR W=0 TO 3
4015    READ T$
4020    PRINT \ PRINT T$
4022    PRINT W(W)
4025    FOR J=1 TO W(W)
4030    LET Z=FNW(W,J) \ PRINT W$(Z),
4040    NEXT J \ PRINT
4050    NEXT W \ PRINT
4075    RETURN
8999
9000    DATA
9005    DATA ADJ,ADV,NOUN-GENERAL,NOUN-SPECIFIC
9999    END
```

```
Administration,N
Administrative,A
Always,AA
Algebra,N
Accumulator,NN
Axiom,NN
Abnormally,AA
Accounting,N
Address,NN
Array,NN
Audit,NN
Automation,N
Available,A
Approved,A
Asynchronous,A
```

ALIEN ATTACK

by Peter Wright

This program, written for PET will work with or without a sound box. Full instructions (with one spelling mistake) are included in the program.

```
2 GOSUB3300
5 J=50
6 R=INT((36)*RND(1))+1
10 PRINT"[illegible]"
20 X=32769:M=1:T=60
30 IFX<32769THENM=1
40 IFX>32806THENM=-1
41 IFR$="@"ORR$="*"THEN4
42 GOTO50
43 IFM=1THENM=-1:GOTO50
44 M=1
50 X=X+M:POKEX,83:POKEX-1,32:POKEX+1,32
52 IFPEEK(X+320)=160ORPEEK(X+320)=214THENGOTO1000
53 IFPEEK(X+640)=214THENR=X+600:GOTO1010
54 IFFF=0THEN60
55 IFFF=1THENGOSUB3004
56 IFFF=1THEN30
60 TT=(J-H):T=T+1:IFT<TTTHEN100
65 H=INT((3)*RND(1)):IFH=1THEN73
66 POKE0,20:POKE1,20:SYS(826)
70 PRINTTAB(R)"[illegible]":GOTO80
73 FORN1=0TO10:POKE0,20:POKE1,20:SYS(826):NEXT
75 PRINTTAB(R)"[illegible]"
80 T=0:R=INT((36)*RND(1))
```

```
100 GETA$
110 IFA$=" "ORA$="P"THENFF=1:F=X+40:POKE0,20:POKE0,10:SYS(826):GOSUB304
120 GOTO30
304 IFRI=1ORLE=1THEN3000
305 F=F+40
306 GETA$:IFA$="@"ORA$="="THEN307
307 GOTO310
308 IFM=1THENM=-1:GOTO310
309 M=1
310 IFPEEK(F)=160ORPEEK(F)=214THENH=H+1:GOTO500
311 IFPEEK(F)<>223THEN320
312 POKEF-40,93:POKEF-41,77:POKEF-1,60
313 FORO=0TO90:NEXT:POKEF-40,32:POKEF-41,32:POKEF-1,32
314 POKE0,20:POKE1,20:SYS(826):D=1:RI=1:F=F-1:GOTO327
320 IFPEEK(F)<>223THEN327
321 POKEF-40,93:POKEF-39,78:POKEF+1,62
322 FORO=0TO90:NEXT:POKEF-40,32:POKEF-39,32:POKEF+1,32
325 POKE0,20:POKE0,20:SYS(826):D=1:LE=1:F=F+1
327 IFF>33723THENFF=0:D=0:RI=0:LE=0:POKEF-40,32:RETURN
328 IFD=1THENRETURN
330 POKEF,46:POKEF-40,32
340 RETURN
500 POKEF-40,32:POKE0,225:POKE1,50:SYS(826)
501 POKEF-40,32:POKEF,32:POKEF-1,32:POKEF+1,32:POKEF+39,32:POKEF+41,32
510 FORO=0TO1
520 POKEF,127:POKEF+39,127:POKEF+40,127
525 POKEF,224:POKEF+39,224:POKEF+40,224
526 POKEF-1,42:POKEF+1,42
530 POKEF+41,127
540 POKEF,225:POKEF+39,255:POKEF+40,255
545 POKEF,102:POKEF+39,102:POKEF+40,102
546 POKEF-1,171:POKEF+1,171
547 POKEF-1,32:POKEF+1,32
550 POKEF+41,255
552 POKEF,32:POKEF+39,32:POKEF+40,32
553 POKEF+41,32
560 NEXT:FF=0
570 IFF=K-40THENPOKE0,255:POKE1,255:SYS(826):GOTO1500
590 D=0:RI=0:LE=0:RETURN
1000 K=X+200
1010 IFPEEK(K)=83THEN1110
1020 POKEK,224
1025 POKE0,100:POKE1,10:SYS(826)
1030 K=K-40:GOTO1010
1100 POKE0,100:POKE1,10:SYS(826)
1110 POKEK,211:POKEK-80,32:POKEK-40,224
1115 POKEK+39,78:POKEK+41,77
1116 POKEK+39,32:POKEK+41,32
1120 K=K+40:IFPEEK(K)=30THENF=K-40:GOSUB501
1130 GOTO1100
1500 FORX=0TO24
1550 FORZ=0TO20:NEXT
1600 PRINT:POKE0,50:POKE1,50:SYS(826)
1700 NEXT
2000 GA=GA+1
2001 IFB$="P"THENPRINT"]QQQQQQQQQ■■■■■I GOT "H"BUT I'M ONLY YOUR PET":GOTO2005
2002 PRINT"]QQQQQQQQQ■■■■■YOU DESTROYED"H"OF THE ALIENS"
2005 IFH2<HTHENH2=H:PRINT:PRINT"Q■■■■■IT IS THE BEST SCORE SO FAR":GOTO2010
2006 PRINT:PRINT"■■■■■THE BEST SCORE IS"H2
2010 FORA=0TO10
2020 GETA$:NEXT
2024 PRINT:PRINT"■■■■■TO PLAY AGAIN PUSH ANY KEY"
2025 PRINT:PRINT"■■■AFTER 30 SECONDS I WILL PLAY ALONE"
2026 TI$="000000"
2027 IFTI$="000030"THENB$="P":H=0:J=5:D=0:GOTO6
2030 GETA$:IFA$=""THEN2027
2035 IFGA=22THENRUN5
2040 J=50:B$="":H=0:D=0:RI=0:LE=0:GOTO6
3000 IFLE=1THENK=41:GOTO3020
3010 K=39
3020 F=F+K:IFF>33723THENPOKEF-K,32:FF=0:D=0:RI=0:LE=0:RETURN
3025 IFPEEK(F)<>32THENPOKEF-K,32:H=H+1:GOTO3040
3030 POKEF,46:POKEF-K,32:RETURN
3040 IFPEEK(F+1)=160ORPEEK(F+1)=214THENF=F+1:GOTO500
3045 IFPEEK(F)=160ORPEEK(F)=214THEN500
3050 IFPEEK(F-1)=160ORPEEK(F-1)=214THENF=F-1:GOTO500
3060 IFPEEK(F-41)=160ORPEEK(F-41)=214THENF=F-41:GOTO500
3070 IFPEEK(F-39)=160ORPEEK(F-39)=214THENF=F-39:GOTO500
3080 IFPEEK(F+41)=160ORPEEK(F+41)=214THENF=F+41:GOTO500
3300 POKE59459,255
3310 FORHB=826TO870
3320 READB:POKEHB,B:NEXTHB
3330 DATA165,1,162,215,142,64,232,170,202,208,253,240,0,240,0,240,0,240,0,240
3340 DATA0,162,233,142,64,234,170,202,208,253,198,00,208,5,234,234,234,234
3350 DATA06,240,01,240,00,208,213
3500 PRINT"]QQQQQQQQQQ■■■■■■■■■■■■ALIEN ATTACK"
3520 PRINT"Q■■■■■■■■■■■■■BY PETER WRIGHT"
3540 FORO=0TO2000:NEXT
4000 PRINT"]QQQQQQQQQQ■■■■■■■■■■DO YOU NEED INSTRUCTIONS ?"
4010 GETA$:IFA$=""THEN4010
4020 IFA$="N"THENRETURN
4030 PRINT"]YOU ARE THE '●' AT THE TOP OF THE SCREEN"
4040 PRINT"THE OBJECT OF THE GAME IS"
4045 PRINT"QTO DESTROY THE ALIENS SHOWN BELOW."
4050 PRINT"Q1,HAS SHORT RANGE WEAPONS AND 2,LONG"
4060 PRINT"QNO1. ◤ ◥        NO2. ◤•◥
4070 PRINT"     •••            ◤•◥
4080 PRINT"THEY WILL APPEAR AT THE BOTTOM."
4090 PRINT"QIF THEY ARE NOT DESTROYED THEY WILL."
4100 PRINT"QMOVE UP AND DESTROY YOU."
4110 PRINT"QTO FIRE AT THEM PRESS THE 'SPACE' KEY."
4120 PRINT"QYOUR SHOT MUST HIT DEAD CENTRE."
4130 PRINT"QIF IT DOES NOT IT WILL BOUNCE OFF."
4140 PRINT"QQ          PUSH ANY KEY TO CONTINUE         "
4150 GETA$:IFA$=""THEN4150
4160 PRINT"]YOU MAY REVERSE THE DIRECTION."
4170 PRINT"QYOU ARE MOVING AT ANY TIME."
4180 PRINT"QWITH THE '=' KEY OR THE '@' KEY."
4190 PRINT"QTHE NUMBER YOU DESTROY IS SHOWN AT."
4200 PRINT"QTHE END OF EACH GAME."
4201 PRINT"QTHE ALIENS ARE SLOW TO ADVANCE AT FIRST."
4202 PRINT"BUT THE MORE YOU DESTROY THE FASTER."
4203 PRINT"QTHEY WILL MOVE."
4204 PRINT"QDON'T LET THEM GET TO CLOSE BECAUSE."
4205 PRINT"QWHEN THEY FIRE THEY NEVER MISS."
4207 PRINT"QPLUG IN A SOUND BOX IF YOU HAVE ONE!"
4210 PRINT"QQ           PUSH ANY KEY TO PLAY          "
4220 GETA$:IFA$=""THEN4220
4230 RETURN
```

Next Month

BENCHTEST
The TRS-80 Model II is an attractively packaged integrated unit with the sort of features that one would expect from a machine with a $5,000 plus price tag. How has Tandy survived the move upmarket . . . ? Stephen Withers reports.

PROGRAMMING – THE SIMPLE APPROACH
Mervyn Axson leads you gently through the minefield of writing your first "real" programs in BASIC.

COMPUTER GAMES
Part three of the series, David Levy introduces a minimax refinement known as the alpha-beta algorithm.

CHOOSING A SYSTEM
Dr Jon Patrick of Prahran C.A.E. presents guidelines for selecting a business system.

STRINGY FLOPPY
Thomas Murphy gives his personal impressions of a device that looks set to radicalize the concept of information storage for the small computer enthusiast. Next month's cover story.

THE COMPLETE PASCAL
Sue Eisenbach and Chris Sadler continue their series with Part 3.

PRACTISING A LITTLE MICRO-CONTROL
Z80 control signals explained by Mike Dennis.

BUZZWORDS
Our dictionary of computerese continued.

PLUS REGULAR FEATURES
Newsprint, Computer Answers, In Store, Systems, Leisure Lines, Interrupt and Programs.

ADVERTISERS INDEX

Anderson Digital Equipment	17
BS Microcomp	29
CISA	42
Computerland (Melbourne)	IFC
Computerware	19, 28
Cottage Computers	14
Deforest Software	2, 29
GFS Electronic Imports	26
Hanimex	24
Informative Systems	5
The Logic Shop	36
Looky Video	51
MS Microsoftware	41, IBC
Ohio Scientific TCG	OBC
Seahorse Computer Services	4

People's Pascal

$29·95

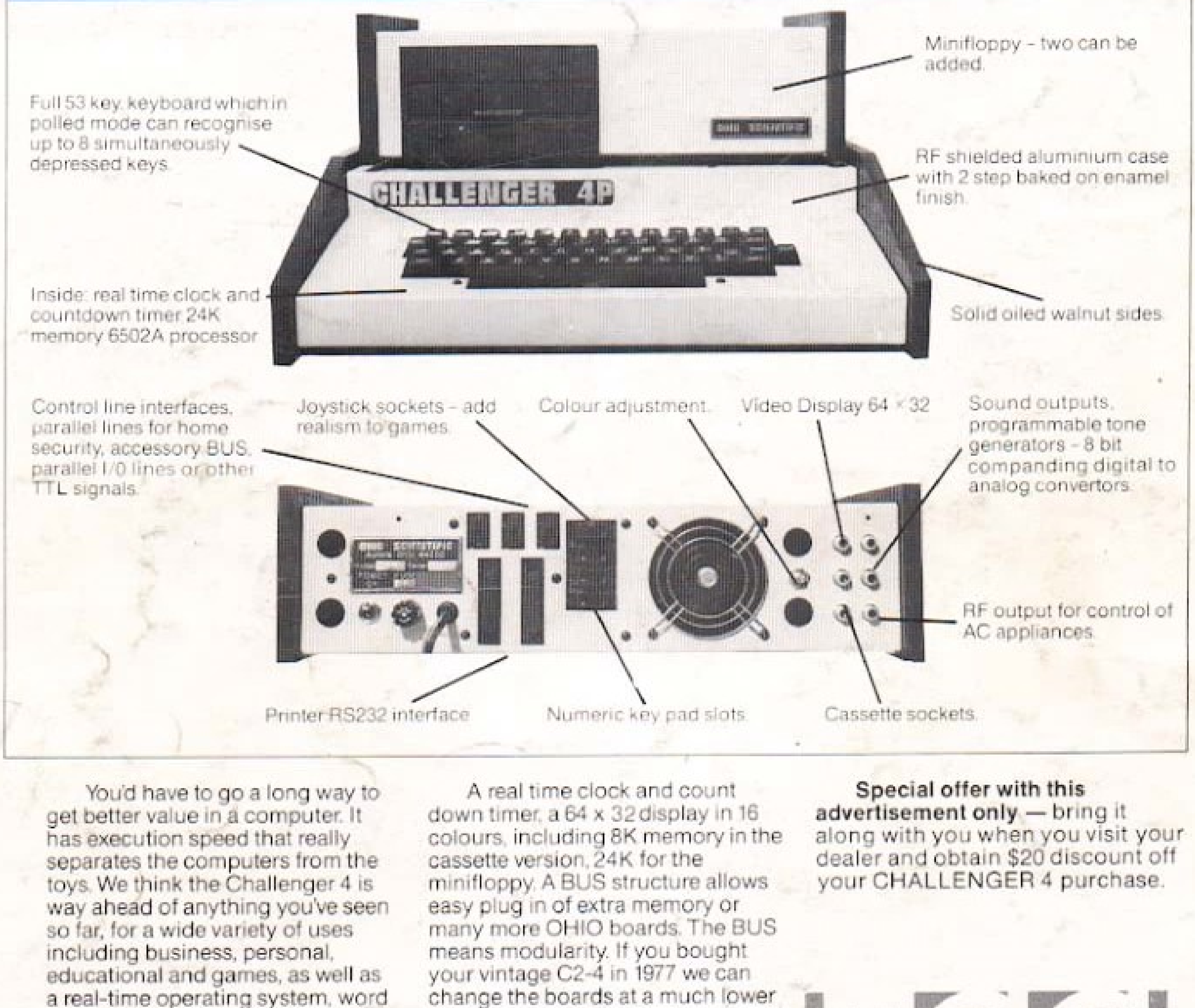
At last your TRS-80* can run Pascal too! The Chung/Yuen "tiny" Pascal is fully implemented for Level II TRS-80*, 16K and up. You no longer need to be left out of the growing group of Pascal users, because People's Pascal gives you everything you need to write structured Pascal programs:
• tiny Pascal compiler • complete text editor for writing your programs • complete tiny Pascal monitor • sample Pascal programs • users manual (TRS-80 Computing issue 1:4)
People's Pascal is both a powerful, structured language and "CPU expeditor". People's Pascal programs, execute at least four times faster than Basic, and often eight-times faster! Special functions open up the complete graphic capability of TRS-80*. You now have the means to write those dazzling, impressive, high-speed graphics programs that are great for games, plotting, statistics, etc.
For the serious computerist, side two of People's Pascal II (tape 6) contains a larger compiler and complete source to the compiler, written in Pascal. This means you can re-compile the compiler, making changes, adding features, etc. (but this will take at least 36K RAM and a solid knowledge of programming).
With the complete People's Pascal operating system, you can save and load both source (Pascal) programs, and compiled programs, to or from cassette tape. This means that once you have de-bugged a program, you can save the P-code (compiled program) and thereafter, to run the program, you need only load the super-fast P-code.
Here is a partial list of People's Pascal features:
recursive procedure/functions • for (loop) • case if/then/else • one-dimensional arrays • write • read constant • repeat/until(loop) • "peek & poke" • plot (graphics for TRS-80*).

MS MICROSOFTWARE

Please send mail order to P.O. Box 119, Essendon, Victoria, 3040.

The Challenger 4

Whichever way you look at it, no other computer offers so much for so little, and in colour!


Minifloppy - two can be added.
Full 53 key, keyboard which in polled mode can recognise up to 8 simultaneously depressed keys.
CHALLENGER 4P
OHIO SCIENTIFIC
RF shielded aluminium case with 2 step baked on enamel finish.
Inside: real time clock and countdown timer 24K memory 6502A processor
Solid oiled walnut sides.
Control line interfaces, parallel lines for home security, accessory BUS, parallel I/0 lines or other TTL signals.
Joystick sockets - add realism to games.
Colour adjustment.
Video Display 64 × 32
Sound outputs, programmable tone generators - 8 bit companding digital to analog convertors.
RF output for control of AC appliances.
Printer RS232 interface
Numeric key pad slots.
Cassette sockets.

You'd have to go a long way to get better value in a computer. It has execution speed that really separates the computers from the toys. We think the Challenger 4 is way ahead of anything you've seen so far, for a wide variety of uses including business, personal, educational and games, as well as a real-time operating system, word processor and a data base management system.

The Challenger 4 has a 2MHz 6502 processor, and if that's not fast enough we can supply the GT option with the 6502C processor, and 120 nanosecond memory which averages over one million instructions per second.

A real time clock and count down timer, a 64 x 32 display in 16 colours, including 8K memory in the cassette version, 24K for the minifloppy. A BUS structure allows easy plug in of extra memory or many more OHIO boards. The BUS means modularity. If you bought your vintage C2-4 in 1977 we can change the boards at a much lower cost than a new computer.

For the best surprise of all ask our opposition if they can provide all these facilities. When they can't, ask us!

Special offer with this advertisement only — bring it along with you when you visit your dealer and obtain $20 discount off your CHALLENGER 4 purchase.

TCG

TOMORROWS TECHNOLOGY TODAY

OHIO SCIENTIFIC

LOCKWOOD TCG 4528

For more information and advice call on your local dealer to help you select the best system for your needs

AUSTRALIAN DISTRIBUTOR-TCG,
31 Hume Street, Crows Nest, N.S.W. 2065

AUTHORISED AUSTRALIAN AGENTS

NEW SOUTH WALES

Bambach Electronics
NEWCASTLE 2-4996

Compuserve (Ncle) Pty. Ltd.
NEWCASTLE 61-2579

Dwell Electronics HORNSBY 487-3111

Hi-Fi Gallery TAMWORTH 66-2525

Macelec WOLLONGONG 29-1455

Manly Stat. Suppliers MANLY 977-2316

Micro Visions KINGSFORD 662-4063

Tel-Professional Consulting Services
FRENCHS FOREST 452-1540

Trevor Burton Pty. Ltd
BALGOWLAH 94-3861

Unique Electronics
MERRYLANDS 682-3325

AUSTRALIAN CAPITAL TERRITORY

MES CANBERRA 82-1774

QUEENSLAND

Dialog BRISBANE 221-4898

Johannes Systems MT. ISA 43-5582

SOUTH AUSTRALIA

Applied Data Control
FULLERTON 79-9211

Ktronics ADELAIDE 212-5505

TASMANIA

Aero Electronics HOBART 34-8232

Eastside Computing
EAST DEVENPORT 27-8121

J. Walsh & Sons HOBART 34-7511

VICTORIA

Comp Co-Ord. Services
BORONIA 762-5937

Computerware FOOTSCRAY 68-4200

WESTERN AUSTRALIA

Datas. Comp. Acc. PERTH 325-5191

Micro Data EAST PERTH 328-1179

Micro Solutions SUBIACO 381-8372

Compulator PERTH 321-6319

NEW ZEALAND DISTRIBUTOR

Computer Consultants, 3 Wolfe Street, Auckland, N.Z. Phone: 79-8345

Computer Consultants, Queens Road, Lower Hutt, N.Z. Phone: 69-4979