An Argus Specialist Publication

Computing today

JANUARY 1984

85p

MACHINE CODE MUSIC ON THE APPLE:

6502 routines can be adapted for other systems

In-depth review of the Electron — a quantum jump for Acorn?

Hi-res graphics on the Commodore 64 explained

Find your files quickly with our Atari Disc Library

Full screen editor for the TRS-80

Plus programs for the Spectrum, Nascom

ADVENTURE COMPETITION

Pick your way through our questions and win the hardware of your choice!

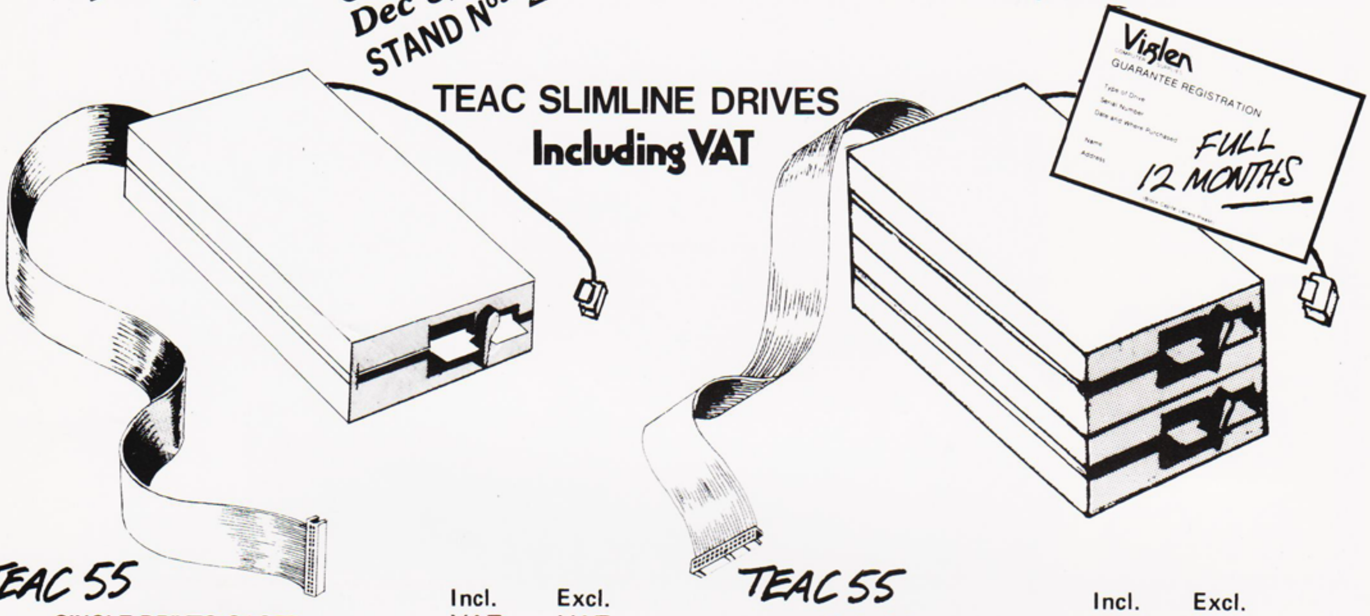
YOU CAN ALSO DO IT IN 5 SECONDS!

INVITATION TO MICRO USER SHOW
Westminster Exhibition Centre
Greycoat Street, London, SW1
Dec 8, 9, 10, 11, 1983
STAND N^os. 28·29·30

Replacing your Cassette with a Disk Drive, means DATA ACCESS in less than 5 seconds!

If you are looking for a guaranteed Disk Drive, fully BBC compatable, at a price that includes VAT, formating disk, user manual, cases and leads, then just look at our all-inclusive prices!

TEAC SLIMLINE DRIVES
Including VAT



TEAC 55

SINGLE DRIVES CASED	Incl. VAT	Excl. VAT
40 TRACK 100K	£166	£144.35
40 TRACK 200K	£230	£200.00
40/80 TRACK SWITCHABLE 200K	£199	£173.05
40/80 TRACK SWITCHABLE 400K	£257	£223.48

TEAC 55

DUAL DRIVES CASED	Incl. VAT	Excl. VAT
40 TRACK 200K	£320	£278.26
40/80 TRACK SWITCHABLE 400K	£402	£359.59
40/80 TRACK SWITCHABLE 800K	£520	£452.18
DFS KIT	£75	£65.22
P.S.U.	£32.20	£28

MITSUBISHI DRIVES	Excl. VAT	Incl. VAT
400K SINGLE DRIVE	£213.04	£245
800K DUAL	£379	£435.85
800K & POWER SUPPLY	£399	£458.85

*These drives are fully compatable with other Computers and can be used should you change your computer.

*Check our special terms for Educational Establishments and Government Departments

DEALER ENQUIRIES WELCOME

SPECIAL OFFER!
FULL HEIGHT DRIVE
£145 inc. VAT
READY TO PLUG IN

HOW TO ORDER

BY POST
To purchase any of the items simply fill in your requirements listed above, in the coupon
Enclose your cheque/P.O.
or use your Access or Barclaycard.
Please make cheque payable to;
VIGLEN COMPUTER SUPPLIES
and post to address below,
allow 7 to 14 days for delivery
& add £ 8—00 postage,
package & insurance on all prices.

BY TELEPHONE
-(01) 843 9903
Credit Card holders
(Access/Barclay Card only)
can purchase by telephone.
Please give Card No, Name,
Address and the item required.

ENQUIRE ABOUT OUR 48-HOUR GUARANTEED DELIVERY SERVICE!

Post to; VIGLEN COMPUTER SUPPLIES UNIT 7
TRUMPERS WAY · HANWELL · W7 2QA

Please send me the following items :

I enclose Cheque/P.O. for £ ______

I prefer to pay Access/Barclaycard
(Delete whichever not applicable)

CARD No ______
SIGNATURE ______
NAME ______
ADDRESS ______

Tel (Day) ______ Tel (Eve) ______

CREDIT CARDS VALID IF SIGNED BY CARD HOLDER.
ADDRESS ABOVE MUST BE THE SAME AS CARD HOLDER.

CTy·1

Viglen
COMPUTER SUPPLIES
UNIT 7 TRUMPERS WAY
HANWELL W7 2QA
Tel: (01) 843 9903

EMS

Acting Editor:
Peter Green
Advertisement Manager
Malcolm Wynd
Sales Executive
David Poulter
Advertisement Copy Control:
Sue Couchman,
Ann McDermott
Managing Editor:
Ron Harris BSc
Chief Executive:
T.J. Connell

Origination and design by MM Design & Print.

ABC Member of the Audit Bureau of Circulation

ISSN 0142-7210

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

Subscription Rates : UK £13.30 including postage. Airmail and other rates upon application to Computing Today Subscriptions Department, 513 London Rd, Thornton Heath, Surrey CR4 6AR.

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

All material should be typed. Any programs submitted must be listed (cassette tapes and discs will not be accepted) and should be accompanied by sufficient documentation to enable their implementation. Please enclose an SAE if you want your manuscript returned, all submissions will be acknowledged. Any published work will be paid for.

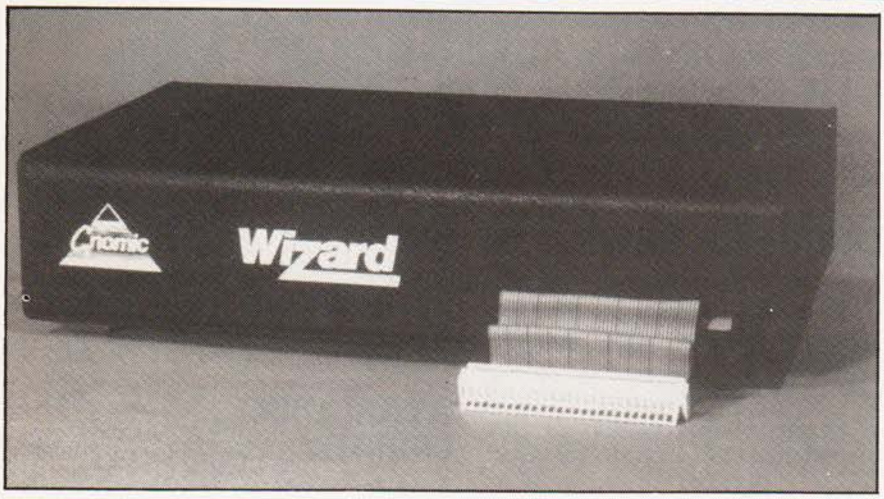
All work for consideration should be sent to the Editor at our Charing Cross Road address.

CONTENTS

VOL 5 NO 11 JANUARY 1984

EDITORIAL & ADVERTISEMENT OFFICE
145 Charing Cross Road, London WC2H 0EE.
Telephone 01-437 1002-7. Telex 8811896.

CONSUMER NEWS



A WIZARD WHEEZE ▲

General Northern Microcomputers Ltd (GNOMIC) of Peterlee, Co Durham have designed the first disc interface for the Colour Genie. The interface, called the Wizard, is now rolling off the production line. Priced at £99.00, the Wizard incorporates both disc and parallel printer interfaces. The GNOMIC software team have also written a Disc Operating System, called QDOS, for the Colour Genie with Wizard interface. With a retail price of only £99.00 GNOMIC are expecting substantial sales for the Wizard, and indeed have already received export orders. For further details contact General Northern Microcomputers Ltd, 8, Whitworth Road, South West Industrial Estate, Peterlee, Co. Durham SR8 2JJ (telephone: Peterlee (0783) 860314).

AT THE ZENITH

Zenith Data Systems has launched two high quality monochrome video monitors which complement its range of desktop microcomputers and peripherals. A choice is available between the ZVM-122 amber display or the ZVM-123 green phospor display. The new monitors offer crisp character definition and are compatible with most business microcomputers and video games, such as the Apple II and III, IBM Personal Computer, Commodore 64 and VIC 20, Texas Instruments TI-99/4A and the Atari 800 and 1200, as well as Zenith's own Z100 desktop computer.

Designed with a bandwidth of 15 MHz that gives the ability to display more than 800 lines horizontally, the ZVM-122/123 monitors have a rise time of only 30 nanoseconds. This results in noticeably improved character definition. Special DC-coupling circuitry permits the video displays to retain their brightness even when a screen is full of information.

A special premium deflection system helps to provide a display that is easy to read. The 12 inch diagonal screen offers a large capacity display with up to 25 lines containing 80 characters each. To optimize the character presentation on the CRT, each monitor is also equipped with a 40-to-80-character switch. This enables the character size to be increased to accommodate a video format when the computer has a 40-column display.

The monitors have optional tilt-bases to permit the user to position the monitor for optimum viewing comfort and the lightweight-style makes the monitor easily portable.

The Zenith ZVM-122/123 monochrome monitors are available for a retail price of £100 for the ZVM-122 and £95 for the ZVM-123, plus VAT, per unit. Zenith Data Systems Ltd is at Bristol Road, Gloucester GL2 6EE. Telephone: 0452 29451.

HIGH STREET DISCS ▶

The UK floppy disc drive distributor, Cumana Ltd, are now marketing disc drive units through well-known High Street retail outlets. This is the first time that a computer peripheral has entered the rapidy-expanding consumer marketplace through the High Street, with appropriate packaging, point-of-sale material and full support from advertising and associated literature.

Cumana's 5¼" Japanese slimline disc drives have many attractive features to tempt the BBC and Dragon Micro users, for which they are specifically designed. Available in single-sided 40 and 80 track, and double-sided 80 track formats, the disc drives are fully assembled and tested before packaging, have a 12 month warranty, and are attractively finished in hard-wearing and robust beige steel cabinets. Each disc drive has an easy-to-use manual door mechanism, and heat dissipation without ventilation holes to prevent any risk of inquisitive young fingers poking screwdrivers into the electronics.

Cumana's design includes an independent power supply — complete with mains power supply lead and plug — enabling up to two and four disc drives to be added to the BBC and Dragon Microcomputers respectively, without any modifications to the computer or risk of it overheating.

The first Cumana disc drive purchased for the BBC Micro — addressed as drive 0 — is supplied complete with comprehensive disc user manual, two-drive connecting cable and formatting diskette. For the Dragon Micro the first Cumana disc drive purchased — addressed as drive A — is supplied complete with the disc user manual, drive connecting cable, demonstration diskette and 'Delta' ROM-based cartridge adaptor.

Packaging for the disc drives has been designed for the shelves of well-known High Street retailers, among them W.H. Smith and Spectrum UK. Full support for the new Cumana disc drives is also being given through advertising and associated literature, as well as

point-of-sale material. Point-of-sale and support literature is presented in a straightforward and easy-to-understand manner, giving the prospective purchaser a clear view of what he or she is buying without all of the usual tedious and complicated technical jargon; although, of course, full technical details are available.

Cumana is already developing dual slimline disc drives, and two versions will shortly be available to supplement the single drive market. These will be slimline disc drives placed side-by-side or 'piggy-back', in a single casing. The company plans a similar marketing strategy for these new dual disc drives, which will be in the High Street alongside the single disc drives in the near future.

OH BROTHER ▲

Lowe Computers of Derbyshire, distributors of the Colour Genie, have signed a major distribution deal with Brother Industries of Japan. At £169.95 (including VAT) from Lowe or one of their Genie Specialist dealers, the new machine will be well within reach for every type of micro user, from the computer whizz-kid to the accountant or businessman tired of juggling pen, paper and calculator on BR's crowded commuter lines.

The new machine is a dot matrix thermal or ribbon printer with a comprehensive QWERTY keyboard and 12-digit calculator. The keyboard features a versatile second shift for accents required in all Roman-script European languages and a range of signs for arithmetical or chemical formulae, including automatic superior and inferior numerals. It even makes sure you don't put an accent over the wrong letter.

Correction facilities include a 32-character buffer and a 16-character screen display with cursor-controlled insertion, deletion and overtype. A 2K continuous memory stores about an A4 page of text with displays for corrections and remaining memory and, most important for computer users, interfacing capability with a computer's disc or tape memory.

By integrating an RS-232 type serial interface into the EP-22, Brother have produced a sort of personal computer printer/typewriter. With any other comparable printer at more than twice the price, the EP-22's clear thermal or ribbon printing and interfacing memory will come as a boon to those who need an affordable first printer or a portable addition to either print or typewriting facilities.

More details can be obtained from Lowe Computers Ltd, Chesterfield Road, Bentley Bridge, Matlock, Derbyshire DE4 5LE (phone 0629 4995/4057).

MODEMS FOR NEWBRAINS

A characteristic of the Newbrain is that its video output turns off while accessing its communications port. This renders it next to useless for acting as a terminal in any application. The solution to this is an independent RS232C port on the main expansion bus. This leads on to the question of how to drive it and what to do with it. A package called "Comm Software" has been developed by Kuma to enable the new RS232C port to couple up to a Modem or Acoustic Coupler and access any bureau computer or electronic mail service.

To date this combination has been proven on Telecom Gold, Comet and many other 300 baud services over BT telephone lines. Work is taking place on providing the viewdata facility on Prestel.

The cost of the device is £69.50 plus VAT. It allows software selection of any baud rate and comes with all the handshakes necessary for communications work. An external power supply for the interface is £6.95 plus VAT.

Details of the Comm Software are as follows. The software is written in machine code for maximum performance (made possible by Kuma's Zen Editor/Assember) and allows the user to define all the usual parameters such as baud rate, parity, stop bits etc. It has various additional features such as split baud rate (vital for Prestel) and a self-defining baud rate for non-standard applications. Prompts and a status line giving useful information make this program very easy to use. The superb typeface of the Newbrain makes it a very attractive terminal. The cost of this cassette-based software is £29.50 plus VAT. For more details contact Kuma Computers Ltd, 11 York Road, Maidenhead, Berks SL6 1SQ (phone 0628 71778).

DISC DRIVE MARK II ▼

Having shown how floppy disc storage cases can withstand motor vehicles (see News last month), other people are now getting in an the act. AMS graphically demonstrated the durability of their own 3" discs by — surprise! — driving a motor car over them. The new discs are encased in plastic sleeves holding 100K per side. Nick Pearson, marketing director of AMS (Advanced Memory Systems Ltd) commented. "We knew these new discs were really strong so we put a couple in the road and drove over them at various speeds up to 60 mph. Not only were they not cracked or anything but worked perfectly when we used them in the drives. We don't recommend everyone doing it but if they can stand up to that sort of punishment they'll withstand anything that schools, companies or the general public hand out".

The 3" disc also features an automatically retracting steel shutter to protect the disc surface, and an overwrite switch protection. The drives (made by Hitachi and housed by AMS in rigid steel, so they probably stand up well to heavy traffic too), come in two configurations: a £225 single drive 100K or a £399 dual drive 200K including cables, VAT and delivery. They use the BBC Micro power supply. The 3" Hitachi drives have an industry-standard interface, so they are electronically identical to 5¼" drives but of course a quarter of the size and with a superior disc system. The drives are 40 track with a track-to-track access time of 3 mS and average access time of 55 mS.

SOFT WARES

EPSON QX-10 GRAPHICS

A business graphics program has been launched for the Epson QX-10 desk top micro which offers the extent of versatility previously only available on mainframes. Known as Dataplot III, and developed by Grafox Limited of Oxford, the package uses data from standard format files, such as those produced by the Supercalc or Peachcalc spreadsheet programs, to produce graphs of any scaling, size, style and type including bar charts, histograms, scattergrams, line graphs and pie diagrams. Up to seven different sets of data may be illustrated at any one time and each of these may employ a different graph type. Multiple graphs can also be shown simultaneously and two independent Y-axes are available to allow sets of data with totally different scales to be placed on the same screen.

Specially-featured in the package is a unique 'parameter assignment' facility which provides for extensive manipulation of data by allowing the user to design statistical and scientific formulae which can be applied to the data before it is graphed.
Further details are available from Grafox Ltd, 35 St Clements, Oxford OX4 1AB (telephone 0865 242597), or from Epson (UK) Ltd, Dorland House, 388 High Road, Wembley, Middlesex HA9 6UH. Telephone: 01-902 8892. Telex: 8814169.

NEW DOS FOR MSX

Microsoft Corp, developers of the 16-bit MS-DOS and IBM Personal Computer DOS operating system, has announced a new eight-bit operating system for MSX microcomputers. Called MSX-DOS, the operating system is designed for MSX microcomputers and will be available to the 14 Japanese and one US microcomputer manufacturer now committed to the MSX standard. The first machines with MSX-DOS will appear in January 1984.

MSX-DOS is CP/M-80 2.2 compatible and runs all Microsoft's eight-bit software including the languages MBASIC, COBOL-80 and FORTRAN-80. Microsoft's Multiplan also runs on MSX microcomputers under MSX-DOS. MSX-DOS supports all MSX hardware including 32, 40 and 80 column text modes as well as MSX-BASIC, an enhanced version of Microsoft's GW-BASIC. MSX-DOS occupies 8K of memory, and includes Microsoft's popular M-80 assembler.

"MSX-DOS was developed in response to the tremendous demand for a Disk Operating System (DOS) from the Japanese MSX manufactuers, who wanted a DOS with the standard MS-DOS disc format and user interface." said David Fraser, Microsoft Ltd's General Manager. "As well as supporting the popular disc drive sizes (5¼" and below) MSX-DOS specifies a common disc format across all machines — something CP/M-80 never did," he added. MSX-DOS supports any type of size of diskette including 3", 3½", 5¼" and 8", with the ability to read discs created by MS-DOS.

This compatibility with MS-DOS means two things. First, users familiar with MS-DOS machines will be instantly able to use a MSX micro with MSX-DOS. The user interface, command structure and command syntax are identical. Second, data is transferrable between MS-DOS and MSX machines without the need for translation utilities or other extras. Multiplan, for example, runs under both MSX-DOS and MS-DOS and data discs can be transferred and updated between machines running these operating systems as required.

One of the grave shortcomings of CP/M-80 was that a program had to be produced in a different format for each microcomputer. MSX-DOS specifies one disc format, common across all microcomputers, thus reducing costs in producing software, and passing these savings on to the user.

The MSX specification was first introduced by Microsoft in June 1983. It provides a standard hardware and software specification for low-end eight-bit microcomputers that will make a range of software compatible with low-cost computers from different manufacturers.

XENIX FOR ACORN

Logica UK Ltd has signed a contract with Acorn Computers Ltd to port the XENIX operating system to a new processor due from Acorn in the Spring of 1984 (now where have we heard that one before?). The 32-bit second processor is designed as an add-on processor for the BBC microcomputer, designed and manufactured by Acorn, and is based on National Semiconductor's 32-bit 16032 microprocessor.

TAKE IT APART

Henry's of Edgware Road, the long-established specialist hardware supplier and main distributor for the Gemini Galaxy range of computers, has entered the software market with the introduction of MDIS, an intelligent dis-assembler. Developed for all CP/M based machines, MDIS is a low-cost package which can be used to take apart CP/M machine code programs for examination and/or modification. Using the correct mnemonics in each case, it will, for example, dis-assemble both the Z80 and 8080 machine codes.

Labelling is automatic and tables are dealt with in an intelligent fashion by reporting data areas in 'defined byte' form with an ASCII printout alongside. As a result, the programmer is given the opportunity to decide whether the table is a text string or access table. The dis-assembled output is entirely compatible with the Microsoft M80 assembler and L80 linker and as the input command syntax has been designed to be similar to Microsoft utilities, MDIS is easy to use. In addition, the MDIS output may be either to disc or hard copy and can be edited without difficulty into forms suitable for assemblers other than Microsoft types.

MDIS is available at £50 plus VAT from Henry's, the sole worldwide distributor who would be pleased to arrange a demonstration to show the program's capabilities. Henry's is at 404-406 Edgware Road, London W2 1ED. Telephone: 01-402 6822.

COMPUTER COURSES

Four new low-cost courses in computer studies are being introduced by the Open University, as part of its non-degree studies. The courses feature programming, systems analysis and design, and the principles of data analysis. All four courses are suitable for study at home, or use by educational establishments, business groups and user groups, and include exercises and self-assessment questions to monitor students' progress.

Structured Programming with UCSD Pascal is intended for the many people who require training in problem-solving techniques enabling them to design and implement software. Also suitable for home micro users who have a UCSD Pascal compiler, the course costs £65.

Introduction to Commercial Data Processing with COBOL uses the most popular business language and is aimed at those with little computing background. Ideal for in-house training as well as for the home-user the course gives a complete introduction to the subject. On-line access to the Open University's Academic Computing Service through a local study centre is available for those without computing facilities and the course fee in this instance is £175; for micro users with a COBOL compiler the cost is £130. Both the UCSD Pascal and the COBOL courses involve approximately 100 hours' study time.

Programmers and prospective systems analysts will find **Introduction to Systems Analysis and Design** an ideal course to take them a step further in their careers but it would be equally useful for managers and administrators wishing to gain a greater insight into the subject. The fee is just £50 and unlike the other three courses, this course has no practical computing element. Material for the 40-hour course includes video and audio cassettes, a study guide and reference handbook.

Data Analysis for Information Systems Design will appeal to data base systems managers and to analysts, in fact anyone requiring a grounding in the principles and skills of data analysis. This course, like all the three other courses, can be followed at a pace to suit the student and involves between 40 and 50 hours of study. If required, on-line access of up to five hours is available via OU study centres, in which case the course fee is £110 (£95 if using your own terminal).

For more detailed information on all these courses, or to enrol, contact the Student Enquiry Office, The Open University, PO Box 71, Walton Hall, Milton Keynes MK7 6AG.

ABERSOFT GOES AUSSIE

All publishing rights to Abersoft FORTH, the version we are using in our teach-yourself series, have been secured by publishers Melbourne House who will be producing new documentation on this high level language in the next few weeks.

Already hailed as being the package which is by far the best implementation of FORTH for the ZX Spectrum (that's why we picked it), Abersoft FORTH makes it possible to run programs between 10 and 50 times faster than those written in BASIC, without the tedious programming requirements of machine language. It is even possible to re-write the language to suit individual programming applications. It is, say Melbourne House, **the** tool for programmers who want to squeeze maximum power from their Sinclair Spectrum. Available early November, Melbourne House, Abersoft Forth costs £14.95. The Melbourne House sales office is at 224 Stanley Road, Teddington, Middlesex TW11 8VE. Telephone: 01-977 9160.

GAMES WITHOUT FRONTIERS

With well over 1,000 titles of games software constantly in stock in their nine (soon to be 10) branches, Games Centre are now established as Britain's largest supplier of games software to the general public. Not satisfied with coping with the demand in this manner, Games Centre also offer their entire stock through their mail order department. Their 1983/84 Mail Order Catalogue is now ready and obtainable from the Mail Order Department, Games Centre, 22 Oxford Street, London W1A 2LS.

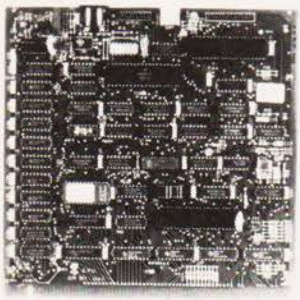
SPECTRUM PASCAL COURSE

Owners of 48K Spectrum owners who are interested in Pascal may wish to attend a course of fourteen weeks duration to be held on Monday, January 9, 1984 at the East Ham College of Technology, High Street South London E6 4ER. The course is intended to introduce Spectrum owners to programming in Pascal using the Hisoft Pascal 4T-version 1.5 compiler.

Students will use their own Spectrums and tape recorders but monitors (TV sets) are provided. The course fee will include the compiler tape and programmers manual. The internal course of study is as follows, week by week: Introduction to the Compiler and Editor; Pascal Syntax and Data Types; Control Structures; Procedures and Functions; Parameters: Formatting; 'Case' Statements; Arrays and Strings; Records; Pointers and Linked Lists; The GOTO statement and use of ZX Printers; Team Programming Project — 1; Team Programming Project — 2; Evaluation and Discussion.

Further details can be obtained from B.F. Boakes, Course Tutor. Telephone: 01-472 1480 ext 249.

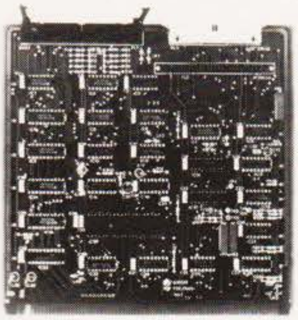
MicroValue 80-BUS MULTIBOARDS



GM813—CPU/64K RAM Board

* 4MHz Z80A C.P.U. * 64K Dynamic RAM
* RS232 Interface * Two 8-Bit I/O Ports
* Cassette Interface
* Extended & Page Addressing Modes
* CP/M Compatible Monitor

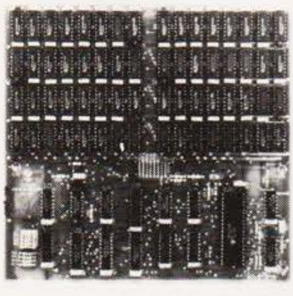
£225



GM829—Disk Controller Board

* Up To 4 Mixed 5.25 & 8" Drives
* SASI Hard Disk Interface
* Single & Double Density Operation
* Single & Double Sided Drive Support
* Supports 48 and 96 TPI Drives

£145

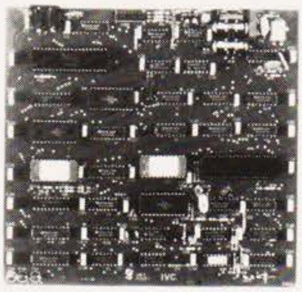


GM833—RAM-DISK Board

* Virtual Disk Operation * 512K Dynamic RAM
* Port Mapped For Easy Interface Software
* Over 10 Times Faster Than a Floppy Disk

£450

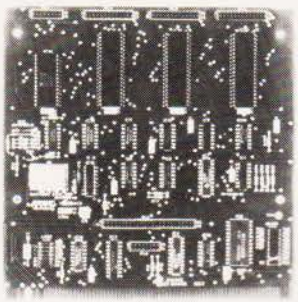
Please note: This board cannot be used as a conventional RAM board



GM812—Video Controller Board

* 80 Characters×25 Line Display Format
* On-board Z80A Microprocessor
* Buffered Keyboard Input
* Programmable Character Generator
* 160×75 Pixel Graphics
* Light Pen Input

£125



GM816—Multi I/O Board

* 6 I/O Ports
* 4 Counter/Timer Channels
* On-Board Real Time Clock
* Battery Backup
* Further Expansion Capability

£125

PLUTO—Colour Graphics Board

* On-Board 16 Bit Microprocessor
* 640×576 Bit Mapped Display
* 192K Of Dual Ported RAM
* Comprehensive On-Board Software

£399

CLIMAX—Colour Graphics Board

* 256×256 Pixel Display
* 16 Colours
* Ultra-fast Vector & Character Generation
* Light Pen Input
* UHF or RGB Outputs

UHF Version **£199**

UHF & RGB Version **£220**

GM811—CPU Board

* 4MHz Z80A CPU
* 4 'Bytewide' Memory Sockets
* 2×8-Bit Input/Output Ports
* 8 Bit Input Port
* RS232 Serial Interface
* Cassette Recorder Interface

£125

GM803—EPROM/ROM Board

* Up to 40K of Firmware
* 2708 or 2716 EPROMS
* Page Mode Operations

£65

GM802—64K RAM Board

* 64K Dynamic RAM
* 4MHz Operation
* RAM Disable Function
* Page Mode Operation

£125

MP826—Static RAM Board

* 32K Static RAM
* Battery Backup
* Page Mode Operation

£185

EV814—IEEE488 (GPIB) Controller

* Cost Effective Controller
* Comprehensive Software Supplied
* Full Implementation
* Easy To Use

£140

GM827—87 Key Keyboard

* User Definable Function Keys
* Numeric Keypad
* Cursor Control Keys

£85

GM839—Prototyping Board

* Fibreglass P.C.B.
* 80-BUS Signal Identification
* High Density IC Capability

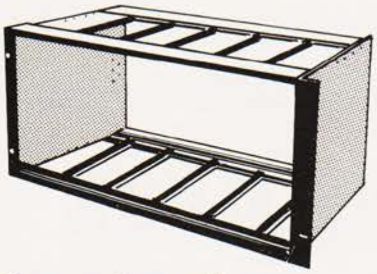
£12.50

MULTIBOARD COMPUTERS

Gemini Microcomputers

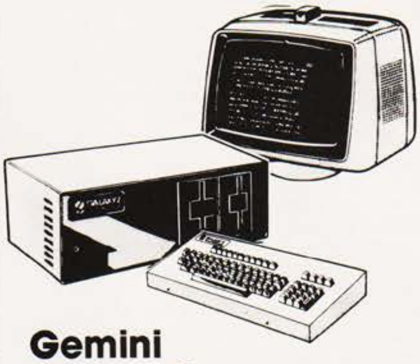
All the boards and components in the 80-BUS range are fully compatible and offer a very flexible and cost effective solution to your computer needs. For further information about the 80-BUS range contact your nearest MICROVALUE dealer.

MicroValue · MicroValue



Power Supplies, Mother Boards & Frames

* GM807	3A Power Supply	£40.00
* GM817	6A Switch Mode P.S.U.	£75.00
* GM843	10A Switch Mode P.S.U.	£95.00
* GM656	3 Slot Motherboard	£5.00
* GM654	5 Slot Motherboard	£6.00
* GM655	8 Slot Motherboard	£10.00
* MP840	14 Slot Motherboard	£47.00
* GM662	5 Board Frame	£50.00
* GM610	19" Frame	£37.50



Gemini Galaxy 2

"I would place the Galaxy at the top of my list" (Computing Today, April 1983)

* Twin Z80A Processors
* CP/M 2.2 Operating System
* 80×25 Video Display
* 64K Dynamic Ram
* Light Pen Interface
* Up to 1.6Mhz Disk Capacity
* Serial RS232 Interface
* Parallel Interface
* Numeric Keypad
* Definable Function Keys
* Cassette Interface
* 12" Monitor Included

from £1495

Quantum System 2000

Computerise Without Compromise

* 80-BUS Construction
* Serial & Parallel Interface
* Stylish Design
* Up To 2.4Mhz Disk Capacity
* Up To Three 5.25" Drives
* Fully Expandable
* Twin Z80A Processors
* CP/M Operating System
* 64K Dynamic Ram
* Definable Function Keys

Two-Drive Quantum £1910

Gemini Multinet

The Gemini Multinet enables as many people as possible to have access to their own microcomputer with mass storage and printer facilities for the lowest possible cost. This is achieved by providing a central 'fileserver' fitted with a Winchester hard disk unit and printer interfaces, in conjunction with a method of interconnecting up to thirty-one workstations to the fileserver. The fileserver and each station are fitted with the Gemini GM836 network interface board. A Micropolis 800K floppy disk drive is incorporated in the fileserver providing backup for the hard disk.

GM910 Galaxy 4 Multinet 5.4 M/byte fileserver	£2600
GM912 Galaxy 4 Multinet 10.8 M/byte fileserver	£2850
GM909 Galaxy 4 Multinet workstation	£650

Both fileservers and workstations are supplied complete with VDU's; the operating software is suppliedwith the fileserver.

Phoenix P12 Monitor

A high quality 12" data display monitor, ideal for Gemini systems. The P12 is available in both green and amber phosphor versions and has a resolution of 20Mhz. **£95**

BUY FROM THE COMPUTER PROFESSIONALS

MICROVALUE DEALERS:

AMERSHAM, BUCKS
Amersham Computer Centre,
18 Woodside Road,
Tel: (02403) 22307

BRISTOL
Target Electronics Ltd., 16 Cherry Lane.
Tel: (0272) 421196

EGHAM, SURREY
Electrovalue Ltd.,
28 St. Judes Road, Englefield Green.
Tel: (07843) 3603

LEEDS
Leeds Computer Centre,
55 Wade Lane, Merrion Centre.
Tel: (0532) 458877

LONDON W2
Henry's Radio, 404 Edgware Road.
Tel: 01-402 6822

LONDON SW11
OFF Records,
Computer House, 58 Battersea Rise,
Clapham Junction.
Tel: 01-223 7730

MANCHESTER M19
EV Computing, 700 Burnage Lane.
Tel: 061-431 4866

NOTTINGHAM
Computerama, (Skytronics Ltd.)
357 Derby Road.
Tel: (0602) 781742

Telephone orders welcome

MicroValue

REAL value – from the Professionals

NEXT MONTH

Computing Today

FEBRUARY ISSUE
ON SALE
JANUARY 13th

SPECTRUM MACHINE CODE

Would you look forward to assembling a few thousand bytes of Z80 machine code using paper and pencil only? No, neither did one of our contributors, so he started looking around for books and software tools to help him in his task. Next month we'll be telling you some of the products he tried and what he finally settled on to do the job. If you want to get to know the language at the heart of your Spectrum, it's required reading.

MULTI-TASKING ZX81-FORTH

Still on the subject of mucking about with Sinclair products, our second feature explains how to give your ZX81 a heart transplant. Out with the old ROM, in with the new — and you have a cheap system that runs multi-tasking FORTH. Apart from speeding up the old machine out of all recognition, this new FORTH ROM provides a multi-tasking facility that allows up to 10 jobs to be run simultaneously. Sounds too good to be true? Read our review in the February **Computing Today**.

SOUNDING OFF ON THE COMMODORE 64

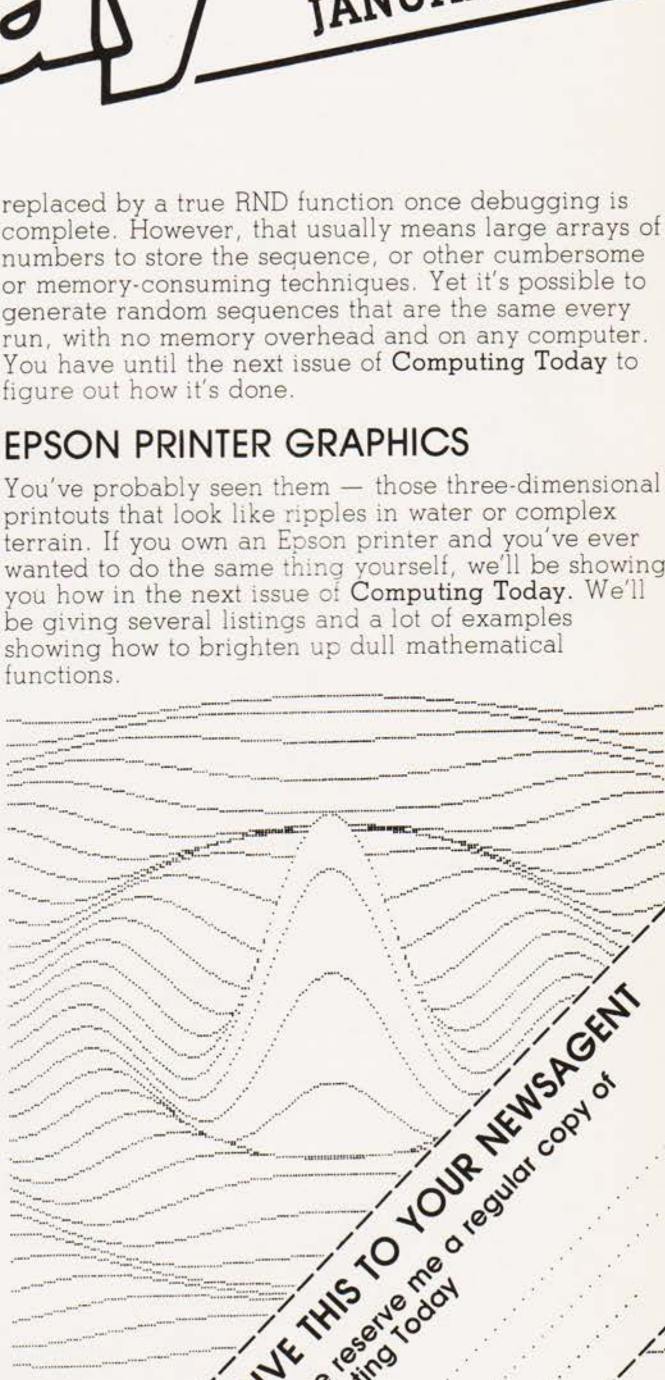
Our third and final part of the 'Getting more . . .' series examines the sound capabilities of the Commodore 64 and delves into the mysteries of the control registers. Not only do we tell you what's going on, we correct some of the errors in the User Manual. A must for all Commodore 64 programmers.

NON-RANDOM RANDOMS

Sounds like a contradiction in terms, doesn't it? But when you're writing a games program that relies heavily on random numbers to control the program operation, it can be very tricky trying to debug the thing. After all, if the program is doing something different every time you run it, how do you check your corrections are having the desired effect? What is needed is a sequence of random numbers that remains the same every time the program runs, which can be replaced by a true RND function once debugging is complete. However, that usually means large arrays of numbers to store the sequence, or other cumbersome or memory-consuming techniques. Yet it's possible to generate random sequences that are the same every run, with no memory overhead and on any computer. You have until the next issue of **Computing Today** to figure out how it's done.

EPSON PRINTER GRAPHICS

You've probably seen them — those three-dimensional printouts that look like ripples in water or complex terrain. If you own an Epson printer and you've ever wanted to do the same thing yourself, we'll be showing you how in the next issue of **Computing Today.** We'll be giving several listings and a lot of examples showing how to brighten up dull mathematical functions.



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

Garry Marshall

BOOK PAGE

This month we look ahead to some of the exciting developments that lie in the future of computing.

This month's book review pages are devoted to only two books, but as they are both about matters of some importance to the future development of computing in general and, in at least one case, to microcomputing in particular, they do deserve extended examination. One of the books deals with a language that is intended to make the full potential of personal computers readily available to all their users. The other is about the way that it is planned to make the next generation of computers more friendly and also capable of doing more of the things that we want them to. If the books are more about computing tomorrow than computing today, we all know that, with computing, tomorrow can come more quickly than we expect.

SMALLTALK

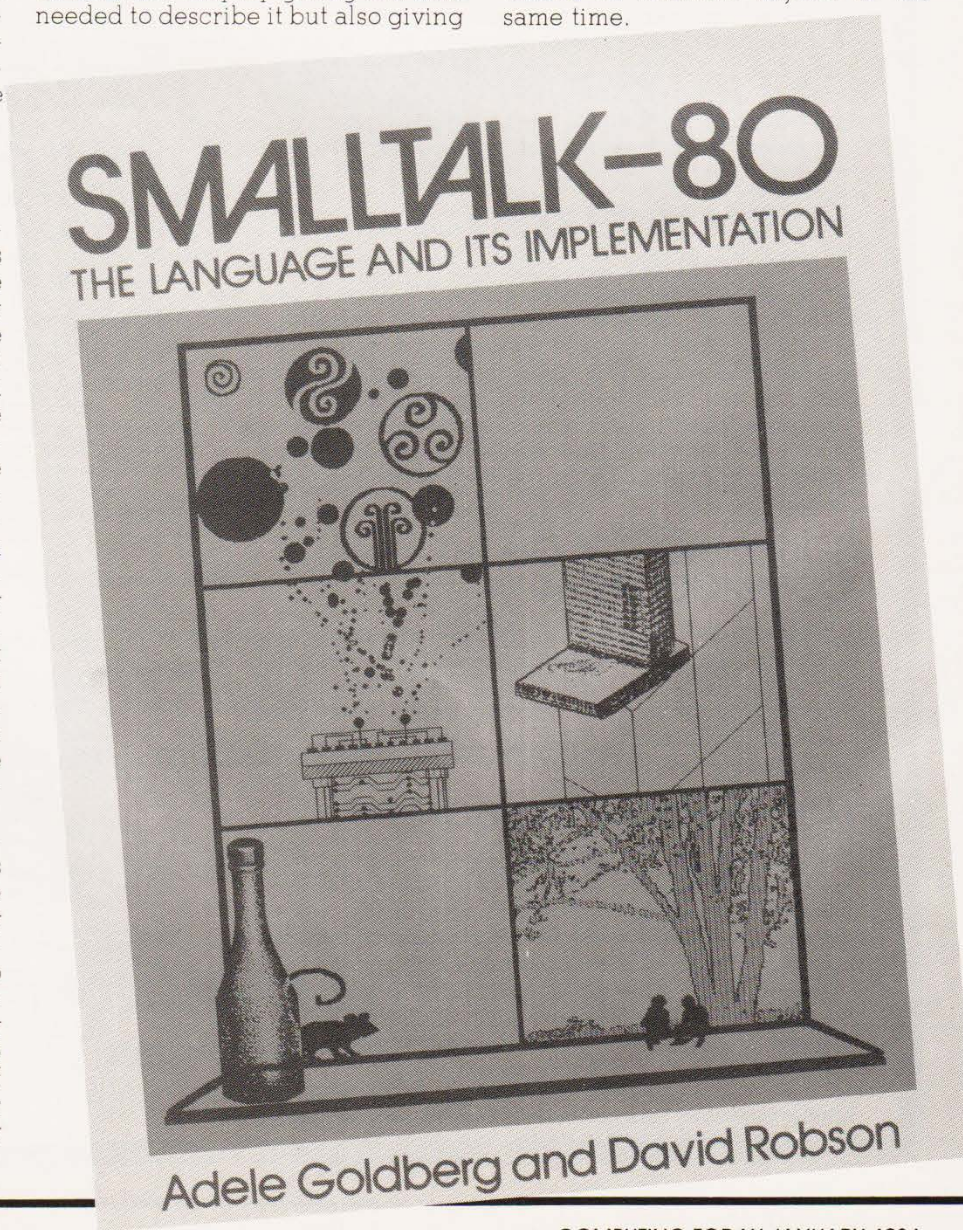
The Xerox Palo Alto Research Centre (PARC) is where the mouse was devised, and it is also where Smalltalk comes from. The Learning Research Group, now renamed the Software Concepts Group, and in particular one of its members, Alan Kay, conceived the idea of the 'Dynabook'. This is a personal computer about the size of a large notebook that should handle practically all of one's information-related needs. It was conceived as a hand-held high-performance computer with a high-resolution display and input and output capabilities. These should allow it to support visual and audio communication and connect to communication networks to access shared information resources. The intention was for the Dynabook to be available as a personal possession in the 1980s.

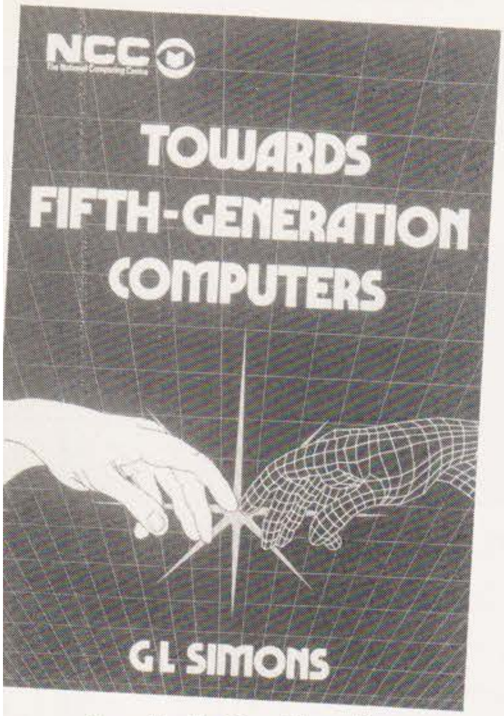
Smalltalk is the software component of the Dynabook project. It is designed to allow the owners of the Dynabook to use it creatively by providing a programming environment that allows programmers to create what they need. A programmer can modify Smalltalk by building his own version of the language, either as the only part of the language that he wants to use, or by extending it in a specialised way that meets his needs.

Conventional languages were found to have severe shortcomings in meeting the needs of the Dynabook. One aspect of their shortcomings lies in their separation of data (the items that programs deal with) and procedures (for manipulating the data). This is apparent in languages such as BASIC and Pascal, as is the serial mode of operation that they impose because their program statements are followed in a strictly sequential manner. By contrast, Smalltalk is based on the idea of the 'object'. An 'object' is a package of information describing an item not only by giving the data needed to describe it but also giving the procedures that determine how the item may be manipulated. Thus, an 'object' consists of data and procedures, all in a single entity. An object is manipulated by sending it a message to carry out one of its procedures to manipulate itself.

To illustrate this, if we have an object for plotting a coloured square on the display screen and moving it around, then it could be named SQUARE, have properties with the self-explanatory names POSITION, SIZE, COLOUR and ORIENTATION, and procedures for manipulating itself called MOVE, ROTATE, ERASE and MAGNIFY. Sending the message SQUARE ROTATE 45 might be expected to cause the object named SQUARE to rotate through 45 degrees, while the message SQUARE ERASE might cause the same object to erase itself from the screen.

One aspect of the object-message approach is that activities can take place in parallel, as they do in real life, because different parts of the computer can manage the activities of different objects at the same time.





Smalltalk-80: The Language and its Implementation is written by two members of the Software Concepts Group at PARC. It describes and explains virtually everything that one might want to know about Smalltalk, but it undoubtedly comes in at rather a high level. Smalltalk is constantly evolving, but the coverage is complete as at the time the book was written. The four parts of the book give an overwview of the underlying ideas of Smalltalk and of its specification, an extended example written in Smalltalk, and a full account of how Smalltalk is implemented.

A less demanding introduction to Smalltalk, in case you are interested or feel the need of it, is provided by the articles in the August 1981 issue of ***Byte,*** which was entirely devoted to Smalltalk, and by an article by Alan Kay in the classic September 1977 issue of ***Scientific American*** devoted to microelectronics.

FIFTH GENERATION COMPUTERS

The evolution of computers to the present is usually seen in terms of the technology used for their hardware. Broadly speaking, computers of the first generation were constructed using valves, the second generation used individual transistors, the third generation makes use of integrated circuits and the fourth generation uses large scale integration (LSI) circuits. Now, for the first time, there is a concerted attempt to design the next generation of comupters. It takes into account both their hardware and software in trying to arrive at a coherent overall design. In this way the next generation should to some extent conform to a definite plan that takes account of what we need of it rather than consisting of barely considered products whose fabrication is determined by a rapidly evolving technology. **Towards Fifth Generation Computers** aims to explain the main trends and developments that will contribute to the fifth generation of computers.

Although work is being carried out already in most of the areas that fifth generation computers will draw on, the book is inevitably about the future (as its title clearly indicates). The fifth generation computer is a target to aim for rather than a reality. A major consideration in developing it is to ensure that it is suitable for people to use, particularly people with no knowledge whatever of computers. The introduction of menus, icons and mice has already made computers much easier to use than before, while developments such as speech input and output will take the process further. The book's cover captures absolutely the idea of man creating computers in his own image.

However, after its very suitable title and imaginative cover, the contents of the book are a disappointment. It is a 'vacuum cleaner' of a book in that a large number of threads from different sources have been sucked into it, but they have not been integrated or digested. In a chapter on the feature that fifth generation computers should have there is coverage of circuit design, memories, architectures, software, languages and knowledge information processing. Material on each topic is presented at a high level, drawn from recent conference, research reports and journals. It can be difficult to understand when lifted too brutally from its full context or when it is too loaded with jargon.

The jargon is a particularly severe problem, since each topic has its own and we are moving fairly quickly from one topic to another. However, a good deal of information on each topic is presented and the sources are listed in an appendix. What is lacking is any indication of how the different topics may interact with each other. There is no indication, for example, of how knowledge will be processed by using particular software, much less of how the designs of hardware and software will affect each other. Thus, no feel is obtained for fifth generation computers as ***systems.*** They appear as collections of parts that may or may not connect with each other.

Chapters on artificial intelligence, expert systems and man-machine communications from the heart of the book, perhaps tipping its balance too heavily towards the software side. Again, no relationships between the different topics are presented, but up-to-date explanations are given in each chapter.

The penultimate chapter on 'The Response of Fifth Generation' finds the author in much better form and apparently much surer of himself. It gives a very readable account of the politics, on the Japanese side, of the introduction of the concept and, in the US, UK and Europe, of responding to it. The Americans, by this account, are not unduly put out by the Japanese initiative, holding a pretty low opinion of their ability for innovation of any kind, let alone that required in this case. The British response is contained in the Alvey report, which recommends that research and development should take place in the four key areas of software engineering, the man/machine interface, intelligent knowledge-based systems and VLSI. The government's response to the report is beginning to become apparent, one aspect being its Information Technology initiative as part of which new courses and new lecturing posts in Information Technology have been established in universities and polytechnics.

The final chapter gives us a glimpse of the sixth generation, with biochips that may be able to rearrange themselves and can be arranged in three dimensions rather than the two dimensions of current chip technologies. The use of biochips for computers is obvious (once you have thought of it!), for computers, in the form of brains, have been generated biologically for a long time.

As the book makes clear, the significance of the fifth generation concept probably lies more in providing a clear strategy for developing the next computers than in providing precise blueprints for actual computers. Some aspects of the fifth generation are probably with us now, and when it can be recognised as such it may well contain what now seems to be sixth generation technology.

This month's books are:

Smalltalk 80: The Language and its Implementation by A. Goldberg and D. Robson (Addison — Wesley), 714 pages, £24.95.

Towards Fifth Generation Computers by G. L. Simons (NCC Wiley), 226 pages, £10.50.

Tony Cross and Phil Cornes

GETTING MORE FROM THE 64 PART 2

Following our look at sprites on the Commodore 64 last month, we examine the high resolution graphics screen. There are also several useful machine code routines that Commodore forgot to include.

A high resolution graphics screen is now an essential part of any self-respecting micro. The extra resolution means that each screenfull contains more information. Diagrams and graphs can be made more detailed and pictures become more realistic. On the Commodore 64 the 320 by 200 dots high resolution screen is more than adequate for most purposes. However, accessing and using the screen is by no means straightforward . . .

SCREEN ORGANISATION

Let's begin by looking at the way the high resolution screen is organised. The high resolution screen is 'bit mapped' into memory. This means that each dot on the screen is controlled by a bit in memory. If the bit is a 1 then the dot will be lit, if it is a 0 then the dot will not be lit.

Armed with this information we can work out how much memory will be needed to store the high resolution screen. The number of dots on the screen is 320 x 200 = 64000. Each byte can control eight dots, so we will need 6400/8 = 8000 bytes to store the screen. In the 64K of RAM available there are eight different places where the screen could be located (65535/8000 = 8.192). However, the VIC chip can only 'see' 16K of RAM at a time and in the standard configuration it is looking at the first 16K. (It is possible to make the VIC chip look at the other 16K banks of memory, but that's beyond the scope of this month's article.)

This means that there are only two possible locations for the screen in the standard configuration. These are at location 0 and at location 8192 decimal. (8192 is used because this is 8K or 2^13, which is a much more convenient number to a computer than 8000.)

We mentioned last month that the first 1K bytes of memory are fully used by the Commodore 64 'operating system'. There are also many BASIC pointers and variables stored in other low memory locations. All this means that the high resolution screen cannot be stored at location 0. So, in the standard configuration, the only place where the high resolution screen can be located is at locations 8192 decimal to 16191 decimal.

LOCATING THE HIGH RESOLUTION SCREEN

To find out where the high resolution screen is located, the VIC chip looks at bit 3 of the VIC memory control register, which is at location 53272 decimal. If the bit is a 0 then the screen is at location 0 (within the current 16K 'bank'). If the bit is a 1 then the screen is at location 8192 (within the current 16K 'bank'). For example, to locate the high resolution screen at 8192 decimal, type POKE 53272, PEEK (53272) OR 8.

Just as with sprite data, locating the high resolution screen at 8192 takes up BASIC text space, and since BASIC starts at location 2048 it is quite likely that BASIC programs will overwrite the high resolution screen. This can be avoided by using a modified version of the MEMSHIFT routine presented last month. MEMSHIFT currently moves the bottom of BASIC up to location 2560 decimal to give room for sprite data. By changing a couple of bytes in the routine it can be made to move BASIC up to location 16384 decimal. (This is the closest to location 16191 decimal that you can get, because the bottom of BASIC can only be shifted in multiples of 256 bytes.)

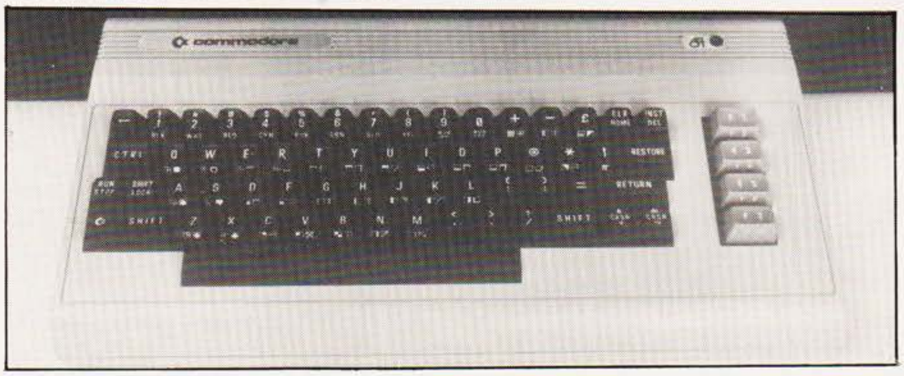
The BASIC program to load this modified version of MEMSHIFT is included at the end of this article. The modified MEMSHIFT is executed in just the same way as the old version, by using the SYS 49329 command from BASIC.

The net result of moving the bottom of BASIC to location 16384 decimal is that we have 'lost' 14K of BASIC space (although we still have 24K left). On the 'plus' side, however, we have protected the high resolution screen and we have enough room between locations 2048 decimal and 4095 decimal for 32 sprites. (Remember from last month's article that sprites cannot be stored between locations 4096 decimal and 8191 decimal.)

USING THE HIGH RESOLUTION SCREEN

Having located the high resolution screen in memory we can now enter high resolution mode. This is done by setting bit 5 of the VIC control register, which is at location 53265 decimal. For example, to enter high resolution mode, use POKE 53265, PEEK(53265) OR 32.

The VIC chip will now start displaying the contents of the high resolution screen. Initially the screen is full of garbage and this comes from two sources. First, the contents of memory locations 8192 to 16191 are probably full of rubbish and this will be displayed as red dots on a black background. In addition, any text characters that were being displayed on the normal text screen are now displayed as different coloured squares on the high resolution



```
Ø . . . . . . . . . . . . . X . . . . . . . . . . . . 319
.
.
.
.
.
.
.
.
.
Y
.
.
.
.
.
.
.
.
.
199
```

Fig. 1 The logical layout of the high resolution screen.

```
         byte Ø    byte 8    byte 16  . . . .  byte 312
         byte 1    byte 9    byte 17  . . . .  byte 313
         byte 2    byte 1Ø   byte 18  . . . .  byte 314
TOP      byte 3    byte 11   byte 19  . . . .  byte 315
LINE     byte 4    byte 12   byte 2Ø  . . . .  byte 316
         byte 5    byte 13   byte 21  . . . .  byte 317
         byte 6    byte 14   byte 22  . . . .  byte 318
         byte 7    byte 15   byte 23  . . . .  byte 319

         byte 32Ø  byte 328  byte 336  . . . . byte 632
         byte 321  etc
         byte 322  etc
NEXT     byte 323  etc
LINE     byte 324  etc
         byte 325  etc
         byte 326  etc
         byte 327  etc

         etc etc
```

Fig. 2 The physical organisation of the high-res screen.

screen. This happens because the normal text screen is used to store the high resolution colour information (more on this later).

The first thing to be done, then, is to clear the high resolution screen. The coloured squares can be removed by printing a 'clear screen' character (Shift/Clr), although they will reappear if you type anything on the keyboard. Clearing the high resolution screen memory can be done from BASIC as follows:

```
100 FOR I=0 to 7999: POKE
    8192+I,0: NEXT I
```

Since this is a very slow process (it takes about 15 seconds to clear the screen), we have also included a machine code clear screen routine. The routine is called CLEAR and it is called by a SYS 49378 command from BASIC. After executing CLEAR, all memory locations from 8192 to 16191 will be set to zero.

PLOTTING POINTS ON THE SCREEN

Just before we actually start drawing on the high resolution screen we need to look at the way in which the screen is laid out. There are two ways of visualising the layout of the high resolution screen. These are:

- The logical view, which is the way you would like the screen to be organised, as shown in Fig. 1.
- The physical view, which is the way the screen is really organised, as shown in Fig. 2.

Obviously the logical view is the one we want to be using when we refer to screen locations, so a conversion formula to convert from the logical to the physical view is needed. There isn't enough space to be able to describe how the formula is arrived at, but here it is anyway. (Assume that the x and y coordinates are given by X and Y respectively.) The byte which must be changed is given by:

```
BYTE = 8192 + INT(Y/8)*320 +
       INT(X/8)*8 + (Y AND 7)
```

The bit within that byte which must be set is given by:

```
POKE BYTE, PEEK(BYTE) OR
     2^(7 - (X AND 7))
```

In BASIC this tends to be fairly slow, particularly if you use it as part of a line draw routine. To overcome this problem we have included another machine code routine which does the conversion and lights the appropriate dot for you.

The routine uses the value in the integer variable X% as the x coordinate and the value in the integer variable Y% as the y coordinate. There are two 'entry points' to this routine: one called PLOT, which turns on the dot at X%, Y%, and one called UNPLOT which turns off the dot at X%, Y%.

To use the routine, first set up X% and Y% to contain the desired coordinate, then use the SYS 49475 command for the PLOT entry point and the SYS 49488 command for the UNPLOT entry point. No range checking is done on either of the two coordinates although the Y coordinate is always treated as a one byte number, which means that it can never be greater than 255. If either X% or Y% does not exist then a SYNTAX ERROR message will be returned.

USING COLOUR

As we mentioned earlier, the colour information for the high resolution screen comes from the normal text screen at locations 1024 decimal to 2023 decimal. Each text screen location holds the colour for a square eight dots wide by eight dots high on the high resolution screen. Location 1024 controls the colour for screen bytes 0 to 7, location 1025 controls the colour for screen bytes 8 to 15 and so on (see Fig. 2).

Like sprites, the high resolution screen can operate in two colour modes. These are:

- Standard colour mode where each dot on the screen can be one of two colours.
- Extended colour mode where each dot can be in one of four colours.

In 'standard colour' mode the high four bits of the appropriate text screen location hold the colour for a dot which is lit (1), while the low four bits of the text screen location hold the colour for a dot which is not lit (0). Usually the whole screen will be using the same two colours which means that the appropriate composite colour value must be written into the whole text screen. (This is why the screen was initially red dots on a black background. A 'cleared' text screen is really full of 'space' characters which are code 32 decimal. This translates to a red foreground and a black background!)

You can fill the screen with any combination of foreground and background colours by using the following line of code where the variable C contains the composite colour:

```
110 FOR I=1 TO 999: POKE 1024+I,C:
    NEXT I
```

Like the BASIC clear screen routine this process is also fairly slow. In addition, sorting out the composite colour value can be a bit awkward. To overcome these problems we have included another machine code routine (last one this month — honest!) to set up the colours for you.

The routine is called COLOUR ▶

DOT PAIR	DOT COLOUR
00	Background colour register 0
01	High four bits of the text screen memory
10	Low four bits of the text screen memory
11	Normal colour memory

Fig. 3 Dot pair coding for the extended colour mode.

and it reads the values in the BASIC integer variables FC% and BC% to get the foreground and background colours respectively. To use COLOUR, first set up FC% and BC% and then call COLOUR with the SYS 49401 command from BASIC. The whole of the text screen memory will then be filled with the appropriate composite colour value.

No range checking is done on either of the colours but only the low four bits of each value is 'read' which limits their range to 0-15. If a SYNTAX ERROR message is returned by COLOUR then either FC% or BC% did not exist.

In 'extended colour' mode you sacrifice horizontal resolution for increased colours. Instead of every dot being in one of two colours, the horizontal dots are now 'read' in pairs so that each pair of dots can be in one of four colours. Figure 3 shows how the dot pairs are coded into colours.

The background colour register 0 is located at 53281 decimal and it normally holds the text screen background colour. The colour value in this register is the same for the whole high resolution screen.

The normal colour memory runs from 55296 decimal to 56295 decimal. As with the text screen memory, each colour memory location controls the colour for an eight-by-eight square.

To select extended colour mode for the high resolution screen you must set bit 4 of the second VIC control register at location 53270 decimal. For example, to select extended colour mode POKE 53270, PEEK(53270) OR 16.

DRAWING SHAPES ON THE SCREEN

Unfortunately there isn't space this month to look at line and circle drawing routines. We hope to produce another article on this subject at a later date. In any case, lines and circles can be drawn by repeated calls to the PLOT routine with some other control routine, in BASIC or machine code, to set up the X% and Y% coordinates.

THE MACHINE CODE ROUTINES

Finally for this month, here is the BASIC program to load the modified version of MEMSHIFT and the CLEAR, COLOUR, PLOT and UNPLOT routines. Before loading these routines you must first load the set we presented last month. This is because this month's routines call several subroutines which were in last month's listing.

That's all for this month — next momth we will look at the sound and music facilities on the Commodore 64.

```
10 REM Routine to load MEMSHIFT, CLEAR,
   COLOUR, PLOT & UNPLOT
20 FOR A=0 TO 273
30 READ Q : POKE 49329+A, Q
40 NEXT A
100 DATA 162, 64, 160, 0, 24, 32, 156
110 DATA 255, 200, 132, 43, 134, 44, 169, 0
120 DATA 141, 0, 64, 76, 66, 166, 165, 168
130 DATA 197, 170, 208, 7, 165, 167, 197, 169
140 DATA 208, 1, 96, 160, 0, 165, 171, 145
150 DATA 167, 230, 167, 208, 233, 230, 168, 76
160 DATA 198, 192, 169, 0, 133, 167, 169, 32
170 DATA 133, 168, 169, 64, 133, 169, 169, 63
180 DATA 133, 170, 169, 0, 133, 171, 76, 198
190 DATA 192, 169, 195, 133, 251, 169, 194, 133
200 DATA 252, 32, 0, 192, 176, 46, 152, 41
210 DATA 15, 133, 171, 169, 195, 133, 251, 169
220 DATA 198, 133, 252, 32, 0, 192, 176, 28
230 DATA 152, 10, 10, 10, 10, 5, 171, 133
240 DATA 171, 169, 4, 133, 168, 169, 0, 133
250 DATA 167, 169, 7, 133, 170, 169, 232, 133
260 DATA 169, 76, 198, 192, 169, 27, 141, 17
270 DATA 208, 169, 21, 141, 24, 208, 162, 11
280 DATA 108, 0, 3, 32, 95, 193, 32, 128
290 DATA 193, 160, 0, 17, 180, 145, 180, 96
300 DATA 32, 95, 193, 32, 128, 193, 160, 0
310 DATA 73, 255, 49, 180, 145, 180, 96, 169
320 DATA 216, 133, 252, 169, 128, 133, 251, 32
330 DATA 0, 192, 176, 200, 134, 181, 132, 180
340 DATA 169, 217, 133, 252, 169, 128, 133, 251
350 DATA 32, 0, 192, 176, 183, 132, 182, 96
360 DATA 165, 180, 72, 41, 7, 133, 183, 104
370 DATA 41, 248, 133, 180, 165, 182, 72, 41
380 DATA 7, 133, 184, 104, 41, 248, 72, 74
390 DATA 74, 74, 133, 182, 74, 74, 24, 101
400 DATA 182, 133, 182, 104, 10, 10, 10, 5
410 DATA 184, 24, 101, 180, 133, 180, 165, 182
420 DATA 101, 181, 24, 105, 32, 133, 181, 169
430 DATA 128, 166, 183, 232, 202, 240, 3, 74
440 DATA 16, 250, 96
```

Listing 1. Routine to load MEMSHIFT, CLEAR, COLOUR, PLOT and UNPLOT.

PLAY ALL DAY WITH THE LATEST AND GREATEST SOFTWARE TO BE SEEN ON SCREEN!

PLAY ALL DAY!

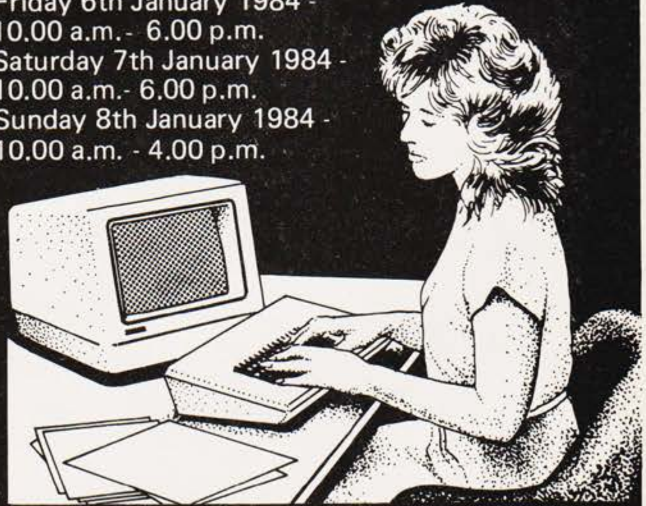
JANUARY 6th 7th 8th

January 6th 7th 8th are the dates when you can play all day with the very latest in software on the very best computers, including Atari, Vic 20 and 64, ZX plus Spectrum, Oric, Dragon and BBC A & B. All will be on display at **The London Home Computer Show!**

Once again the emphasis will be on the lower end of the price bracket, with computers from £50 up to £300.

Play all day, try or buy it's all there at the greatest home computer show on earth!

Friday 6th January 1984 -
10.00 a.m.- 6.00 p.m.
Saturday 7th January 1984 -
10.00 a.m.- 6.00 p.m.
Sunday 8th January 1984 -
10.00 a.m. - 4.00 p.m.



The London Home Computer Show,
Royal Horticultural Society's New Hall,
Greycoat Street,
London S.W.1.

Admission Adults £2.50
Children under 16 £1.50

THE LONDON HOME COMPUTER SHOW

Peter Hewitt

THE PROGRAMMER'S AID

Single key entry of BASIC keywords is made possible on the TRS-80 by this simple program. You also get keyboard entry of graphics characters, and a free giveaway program to test the routine.

A thing I personally find offensive about keying in a BASIC program is that after having taken the trouble to type in, for example, the six characters of the keyword 'RETURN', the computer has the audacity to convert it into a single character (146) before storing it!! It's enough to make you think that the ZX81 has it right!!!

So you can go out and buy a package which gives you a single-keystroke entry facility. This allows you, by hitting a shifted key, to produce one of 26 standard keywords, which are preset for the particular package. The trouble is that it's a safe bet that a keyword you particularly need is missing, and there is very little that you can do about it.

The TRS-80 interpreter has the capacity to convert 123 keywords into 'tokens', in the range 128 to 251. These are shown in the table, together with their start locations in memory, given by the first value times 256 plus the second.

The first program, The Programmer's Aid, pokes into high memory a machine-code routine which allows the user to get into this section of memory and pull out the whole individual keyword with a single keystroke. This is achieved by hitting any of the shifted 'alpha' characters. For example, shifted 'R' (see line 110) accesses the location given by the pair 23/157 — the keyword 'AND'. By altering the values in this line, it can be made to produce AUTO (23/60), ABS (23/171), ATN (23/204), ASC (24/10) or anything else. It depends very much on the sort of program you are developing or copying.

Key the program in carefully, but please save it before attempting to RUN it! When you are satisfied that all is well then RUN it. It will return to READY. Now hit any shifted letter, and you will see that the full keyword appears instantaneously. If you think that you would have preferred to have had a different keyword from that key, then you will find that the reference pairs are stored between locations 32703 and 32754. You can do a little PEEKing and POKEing to make any necessary corrections. DO NOT attempt to amend the program and re-RUN it. This will lock the whole system up, since it won't know what it has done with its 'Keyboard Driver'. If you have suffered from the same thing, you will know how painful ***that*** is! If you want to start again, you must switch off, then re-load the program from tape (or disc of course). To be entirely safe, you should remember to enter NEW immediately after running the program. Note that the program has two lines numbered 50. You will only need one of them, depending on your operating system: simply omit the other. If you are using DOS, then be warned that you will have as little as 5.3K of memory left, and you may have to look into re-locating the program, which is not straightforward.

GRAPHICS TOO

Got it de-bugged and working? OK, now hit Shift and Right-arrow: gosh, a flashing cursor! Now type in a few characters, and you will get what appears to be garbage. Well, have faith. What you are getting is the graphic character corresponding to the ASCII value of the key you hit, plus 100. Thus key 'P' produces character 180. There are five exceptions, and these are shown at the end of the program listing. To return to 'normal' operation, simply hit Shift and Right-arrow again. The cursor becomes static again, confirming that your message has been received and understood.

There is little terribly significant about the program itself. Note that it sets its own memory limit in line 30, thus saving you the chore. This is preset to 32552, which means that the first few bytes of the routine are available for overwriting, since they are redundant after the keyboard driver has been relocated.

WINDOW DRESSING

In the hope of demonstrating the use of this new tool, the second program listing is of a fairly well-known game, which I have chosen to call 'Window'. The idea is simple enough: a 'ball' is dispatched towards a window, and half-way there, suddenly disappears. You have got to hit the space bar to stop it before it hits the window, and the closer to the pane, the higher the score — up to 20 for a 'toucher'. Simple enough, but the graphics for the window breaking are a little bit fussy, or would be if the facility to enter graphics direct were not available.

Start entering the program as normal. You should first set AUTO 10, on the basis that you will (or should) be avoiding the 'Remark' lines. When you come to a 'graphic' section, enter the quotes, then hit Shift and Right-arrow. Now hit the letter or key shown on the listing, and, all being well, you will get the requisite graphic character. The first occurs in line 20 — the 'L'. The remainder should be fairly apparent, but note carefully the abreviations used, and hit the correct keys. Remember to return to 'normal' operation to input the trailing quotes. You should be making good use of your single keystroke facility. You may stop at line 420, since this is the end of the game.

When you list the program, you will probably be stunned by what the machine has done to your lovely graphics. It's gone and changed them all into keywords. You can't trust anything these days. However, don't worry, this is simply one of the machine's little foibles, and can be ignored, because the graphics will be all right.

Well, when I say all right, there is one small problem. You can't edit a line which includes graphic characters. Our friendly BASIC interpreter unjumbles all the codes when you are editing them, and assumes you don't want them back! But fear not, help is at hand. At lines 430 to 500 is a whole additional program — a BASIC line searching system. In order to look at the innards of any line in your program, RUN 430. You will be asked for 'LINE NUMBER', which you should enter. It skips through all of the lines sequentially, until it comes to the one requested.

The display is quite interesting. It shows the memory location, together with its contents both in ASCII and character codes. You can move through by hitting 'Newline' ('Enter'), enter any replacement value, if required, or terminate by entering '999', or breaking. The first two items make up the memory address of the next line. The third and fourth are the line number; and you can alter this if you have need to. (How else did I get two line 50's on my listing ???). As you move through, you will see both 'tokens' and the character contents of the line, any of which you alter by simply inputting a new value.

Apart from having the facility to 'fix' lines which include graphics, you will no doubt have spotted that graphic characters can be inserted directly into a BASIC line, providing that the space has been made available. It is even practicable to put in 'control' characters. Take, for example, line 30. Input the line as:

```
30 X = "[6 SPC]"
(in CT Standard notation)
```

Set the line-searcher to find line 30, and step through to the character after the first quote (Char 34). Now change the next six spaces (Char 32) to 24,24,24,24, 24,26 and enter 999 to exit. When you LIST the line, you will see that it's a bit of an oddity. The LISTing process has taken each character literally, moving the pointer back (Char 24) five times, then down (Char 26). However, the string 'X' is exactly the same as that which would have been produced by using CHR$ and STRING$.

If you are heavily into machine code, you will find the line-searcher a most convenient way of inserting the characters into an appropriate REM line. All in all, it is an extremely useful facility, and well worth tagging onto the end of any BASIC program under development, in order to give additional flexibility.

HOW WINDOW WORKS

The string array V (lines 40 to 130) holds 'big' numbers, based on a 10 by 9 pixel matrix. The array X represents 11 'frames' of the window breaking, which, when displayed consecutively, give a very realistic effect.

Line 300 flashes the ball, until the space bar is hit to send it on its way. It moves forward one pixel at a time, which means that it must be consecutively represented by two pairs of characters (line 320). Note that the delay, such as it is, is achieved by calling line 390. The process of actually finding this line is sufficiently long-winded to give an adequate delay, and the Return is immediate.

The last graphic character occurs in line 340, and lines 350 and 360 give special displays for zero, or maximum scores. The remaining lines display 'big' numbers, and even allow for some eggheaded genius who might hit a maximum 100 with five shots. In fact, scores over 90 would be considered outstandingly good.

THE LINE-SEARCHING FACILITY

Sixteen-bit numbers are stored with the least significant byte first, so that two consecutive eight-bit numbers A and B represent a 16-bit number A + 256 *B. Lines 490 and 500 pick up two bytes from a specified location, and carry out

Table 1. The TRS-80 keywords and their start locations (see text).

22	80	END	22	83	FOR	22	86	RESET	22	91	SET	22	94	CLS
22	97	CMD	22	100	RANDOM	22	106	NEXT	22	110	DATA	22	114	INPUT
22	119	DIM	22	122	READ	22	126	LET	22	129	GOTO	22	133	RUN
22	136	IF	22	138	RESTORE	22	145	GOSUB	22	150	RETURN	22	156	REM
22	159	STOP	22	163	ELSE	22	167	TRON	22	171	TROFF	22	176	DEFSTR
22	182	DEFINT	22	188	DEFSNG	22	194	DEFDBL	22	200	LINE	22	204	EDIT
22	208	ERROR	22	213	RESUME	22	219	OUT	22	222	ON	22	224	OPEN
22	228	FIELD	22	233	GET	22	236	PUT	22	239	CLOSE	22	244	LOAD
22	248	MERGE	22	253	NAME	23	1	KILL	23	5	LSET	23	9	RSET
23	13	SAVE	23	17	SYSTEM	23	23	LPRINT	23	29	DEF	23	32	POKE
23	36	PRINT	23	41	CONT	23	45	LIST	23	49	LLIST	23	54	DELETE
23	60	AUTO	23	64	CLEAR	23	69	CLOAD	23	74	CSAVE	23	79	NEW
23	82	TAB(	23	86	TO	23	88	FN	23	90	USING	23	95	VARPTR
23	101	USR	23	104	ERL	23	107	ERR	23	110	STRING$	23	117	INSTR
23	122	POINT	23	127	TIME$	23	132	MEM	23	135	INKEY$	23	141	THEN
23	145	NOT	23	148	STEP	23	152	+	23	153	-	23	154	*
23	155	/	23	156	[	23	157	AND	23	160	OR	23	162	>
23	163	=	23	164	<	23	165	SGN	23	168	INT	23	171	ABS
23	174	FRE	23	177	INP	23	180	POS	23	183	SQR	23	186	RND
23	189	LOG	23	192	EXP	23	195	COS	23	198	SIN	23	201	TAN
23	204	ATN	23	207	PEEK	23	211	CVI	23	214	CVS	23	217	CVD
23	220	EOF	23	223	LOC	23	226	LOF	23	229	MKI$	23	233	MKS$
23	237	MKD$	23	241	CINT	23	245	CSNG	23	249	CDBL	23	253	FIX
24	0	LEN	24	3	STR$	24	7	VAL	24	10	ASC	24	13	CHR$
24	17	LEFT$	24	22	RIGHT$	24	28	MID$						

this conversion, making due allowance for machines with more than 16 K. The first location searched (from line 430) is 16548, which is where the BASIC start-line pointer is stored.

As it reaches each line, it looks to see if it is the required number (line 440), and, if not, makes use of the stored pointer to go to the next line, to continue the search.

Once the required line is found, it displays the location character value (putting "." if it is unprintable), and the ASCII code. Hitting 'Enter' moves to the next byte, without amendment (line 480). Alternatively, a new value can be entered.

If you are among the select band of those with a machine possessing more than 16 K, you will note that location numbers go negative after 32767 — which is the standard TRS-80 method of handling PEEKs and POKEs in that range.

Listing 1. The Programmer's Aid.

```
20 REM ---    ********************
              * PROGRAMMERS' AID *
              *        BY        *
              *  PETER   HEWITT  *
              ********************
30 POKE 16561,40: POKE 16562,127: CLEAR 50: RUN 40
40 FOR I = 32538 TO 32754: READ A: POKE I ,A: NEXT
50 DEFUSR0 = 32538:                  '---- FOR DOS USERS
50 POKE 16526,26: POKE 16527,127:    '---- FOR TAPE USERS
60 A = USR(0): NEW
70 DATA 42,22,64,34,179,127,33,41,127,34,22,64,195,204,6,58,
        186,127,183,40,46,58,187,127,60,50,187,127,32,7,42,32,
        64,126,238,127,119,205,178,127,254,33,48,88,254,25,40,
        91,33,181,127,6,5,79,126,185,40,5,35,16,249,121,201,62
80 DATA 133,144,201,58,188,127,183,32,38,205,178,127,254,25,40,
        47,254,97,248,254,123,240,33,191,127,214,97,203,39,95,
        22,0,25,86,35,94,235,126,230,127,50,188,127,34,189,127,
        201,42,189,127,35,34,189,127,126,203,127,200,175,50,188
90 DATA 127,201,50,186,127,24,139,254,92,48,135,198,100,201,
        175,50,186,127,62,95,42,32,64,119,195,41,127,195,227,
        3,32,10,31,1,9,0,0,0,0
100 REM       DATA PAIR FOR SHIFTED :-
110 DATA 23,157         :'A
120 DATA 22,145         :'B
130 DATA 24,13          :'C
140 DATA 22,110         :'D
150 DATA 22,163         :'E
160 DATA 22,83          :'F
170 DATA 22,129         :'G
180 DATA 23,17          :'H
190 DATA 23,135         :'I
200 DATA 22,114         :'J
210 DATA 24,0           :'K
220 DATA 24,17          :'L
230 DATA 24,28          :'M
240 DATA 22,106         :'N
250 DATA 23,32          :'O
260 DATA 23,207         :'P
270 DATA 24,22          :'Q
280 DATA 22,150         :'R
290 DATA 23,110         :'S
300 DATA 23,141         :'T
310 DATA 23,101         :'U
320 DATA 22,204         :'V
330 DATA 23,45          :'W
340 DATA 22,133         :'X
350 DATA 23,74          :'Y
360 DATA 23,69          :'Z
370 '    ***********************
         * SPECIAL CODES ARE:  *
         * 128 = <SPACE>       *
         * 129 = <DOWN-ARROW>  *
         * 130 = <CLEAR>       *
         * 131 = <BREAK>       *
         * 132 = <RIGHT-ARROW> *
         ***********************
```

Listing 2. The Window game.

```
9 REM   *********************
        * W  I  N  D  O  W  *
        *                   *
        *       b  y        *
        *                   *
        *   P  E  T  E  R   *
        * H  E  W  I  T  T  *
        *********************

10  CLEAR 1000: DEFINT A-T: DEFSTR V-Z: DIM X(11)
20  W = "WOWWWEEEEEEEEEEEEEEEE!!!!": WW = "":FOR I = 1 TO 25:
    WW = WW + MID$(W,I,1) + "L": NEXT
30  X = STRING$(5,24) + CHR$(26)

39  REM -- Big numbers !!
    Abbreviations used are:
     ^ = <SPACE>          (94=128)
     b = <BREAK>          (98=131)
     c = <CLEAR>          (99=130)
     d = <DOWN-ARROW>    (100=129)
     r = <RIGHT-ARROW>   (114=132)
     [ = <UP-ARROW>       (91=191)
40  V(0) = "ZbbbY" + X + "[^^^[" + X + "KLLL;"
50  V(1) = "^$[^^" + X + "^^[^^" + X + "<L[L,"
60  V(2) = "*bbbY" + X + "T(((#" + X + "[LLLL"
70  V(3) = "*bbbY" + X + "^$((S" + X + "HLLL;"
80  V(4) = "^(:d^" + X + "TSL[L" + X + "^^^[^"
90  V(5) = "[bbbd" + X + "+(((P" + X + "HLLL;"
100 V(6) = "Zbbb)" + X + "[(((P" + X + "KLLL;"
110 V(7) = "bbbW#" + X + "^(:d^" + X + "T#^^^"
120 V(8) = "ZbbbY" + X + "W(((S" + X + "KLLL;"
130 V(9) = "ZbbbY" + X + "'((([" + X + "HLLL;"
140 ZZ  =  "^^^^^" + X + "^^^^^" + X + "^^^^^": X = X + CHR$(24)

149 REM -- Window-breaking graphics

150 X(1)  = "^1^^^^" + X + "c3^^^^"
160 X(2)  = "^A^^^^" + X + "^3d^^^"
170 X(3)  = "^%,^^^" + X + "^4br^^"
180 X(4)  = "^c0^^^" + X + "^<"%^^"
190 X(5)  = "^$$^^^" + X + "^^Nb,^"
200 X(6)  = "^^,$^^" + X + "^^<2-^"
210 X(7)  = "^^^<^^" + X + "^^>L/^"
220 X(8)  = "^^^^^^" + X + "^^<P/d"
230 X(9)  = "^^^^^^" + X + "^^<P0b"
240 X(10) = "^^^^^^" + X + "^^<PLc"
250 X(11) = "^^^^^^" + X + "^^<PL^"

259 REM -- The actual window

260 X = X + CHR$(24): Y = "^^1^^^^":
    W = "[[[[[[[" + X + "+K[++++" +X+Y+X+Y+X+Y+X+Y+X+ "[bbbbb["
270 CLS:PRINT@6,
    "TOTAL SCORE^^^^^^THIS^^TIME^^^^^^LEFT^^^^^^^^^^HIGH SCORE";
279 REM -- Game start

280 PRINT @166, V(5);: PRINT @128,ZZ;: AA = 0: NN = 156:
    GOSUB 420: NN = 140: GOSUB 420: NL = 5: TS = 0
290 Y = INKEY$: PRINT @632,W;: XX = " ":PRINT @896,
    "^^^------------ HIT <SPACE> TO START AND STOP -----------^^^";
300 Y = INKEY$: IF Y<> " " THEN IF XX = " " THEN XX = "b"
    ELSE XX = " " ELSE PRINT @897,STRING$(56,32);: GOTO 320
310 PRINT @896,XX;: GOTO 300

319 REM -- Ball visible

320 FOR I = 896 TO 921: PRINT @I, "cd";: GOSUB 390:
    PRINT @I, "^b";: GOSUB 390:NEXT

329 REM -- Ball becomes invisible

330 PRINT @921, "  ";: FOR I = 922 TO 952:
    IF INKEY$ <> " " THEN A = 1:
    IF INKEY$ <> " " THEN A = 2: GOSUB 390: NEXT:
    FOR I = 1 TO 11: PRINT @889, X(I);: NEXT: SC = 0: GOTO 370

339 REM -- Show where the ball was

340 IFA = 1 THEN PRINT @I,"cd"; ELSE PRINT @I,"^b";

349 REM -- Calculate score, chicken if less than zero

350 SC = 18 - (952-I) * 2 + A: IF SC < 0 THEN SC = 0:
    PRINT @910,"---+++ C H I C K E N ! ! +++---";: GOSUB 400

359 REM -- Big deal for a 20 score !!

360 IF SC = 20 THEN N = 0: M = 30: FOR I = 1 TO 50:
    PRINT @896, LEFT$ (WW,I);: M = M - 2: FOR J = 1 TO M:
    NEXT J,I: GOSUB 400

369 REM -- Display score

370 GOSUB 410: NL = NL - 1: PRINT @166,V(NL);:
    IF NL > 0 THEN GOSUB 400: GOTO 290
    ELSE IF TS > HS THEN HS = TS: NN = 186:
    IF HS < 100 THEN AA = HS: FOR I = 1 TO 10: PRINT @180,ZZ;:
    PRINT @186,ZZ;: GOSUB 390: GOSUB 420: GOSUB 390: NEXT
380 GOSUB 400: GOSUB 400: GOTO 280

389 REM -- Delays

390 RETURN
400 FOR K = 1 TO 600: NEXT: RETURN

409 REM -- Score display routines

410 AA = SC: NN = 156: GOSUB 420: TS = TS + SC:
    IF TS < 100 THEN AA = TS: NN = 140: GOTO 420
    ELSE V = V(0): PRINT @128,V(1);: PRINT @134,V;:
    PRINT @140,V;: PRINT @174,V(1);: PRINT @180,V;:
    PRINT @186,V;: RETURN
420 PRINT @ NN,ZZ;: PRINT @ NN-6, ZZ;: A = INT(AA/10):
    B = AA - A * 10: PRINT @ NN, V(B);: IF A = 0 THEN RETURN
    ELSE PRINT @ NN - 6, V(A);: RETURN
429 REM --
         The BASIC line searching routine

430 J = 16548: GOSUB 490: I = L: INPUT "LINE NUMBER";N
440 J = I+2: GOSUB 490: PRINT L: IF L <> N THEN J = I:
    GOSUB 490: I = L: GOTO 440
450 J = I - 1
460 J = J + 1: I = J: IF I > 32767 THEN I = I - 65536
470 PRINT I;: L = PEEK(I):
    IF L > 31 AND L < 192 THEN PRINT " ";CHR$(L);" ";
    ELSE PRINT" . ";

479 REM -- Hit <NL> (<ENTER>) to move to next location
              and '999' to terminate

480 PRINT L;: A = 0: INPUT A:IF A THEN IF A = 999 THEN END
    ELSE POKE I,A: GOTO 460 ELSE 460
490 IF J > 32767 THEN J = J - 65536
500 L = PEEK (J) + 256 * PEEK (J+1): RETURN
```

There's one game you should play with a home computer before you buy it.

20 Questions.

If you're looking for your first home computer, may we make a suggestion.

Tear out this page.

And ask these questions of every computer you look at.

1) Is the basic price reasonable, say under £180?

2) Does it work with a monitor as well as with an ordinary UHF T.V.?

3) Does it include a basic training manual that doesn't require a science degree to understand?

4) Does it have colour?

5) And sound?

6) Does it take cassettes?

7) And cartridges?

8) And can you operate them with joysticks working directly off the basic unit?

9) Does it have a professional quality keyboard that's guaranteed for twenty million depressions (and no headaches)?

10) Does it have colour graphics that are capable of producing detailed pictures?

11) Is the case robust enough to stand up to the kids?

12) Does it use a standard language that's not a million miles away from English?

13) Is there an expanding range of high quality software available from the makers?

14) Does the software include educational programs for adults as well as the very young?

15) And is there software sophisticated enough to satisfy the experts?

16) Does it use the most up to date technology, for instance the 6809E microprocessor?

17) Will it drive a standard printer directly from the basic unit?

18) Is it made in the UK?

19) And for the basic price will I get an easily accessible, yet large memory of at least 32K?

20) Or will I have to add lots of extras (and lots of money) for that much?

After you've finished pestering every computer salesman in sight, compare your answers to these, courtesy of the Dragon 32 (this is, after all, an ad for the Dragon).

Questions 1-19. Yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes and yes. Question 20. No.

If you're thinking that that's a lot of computer for the money, you'd be right.

However, when you come to read the handbook you'll soon realise that something's missing.

The jargon usually associated with computers.

You see every part of the Dragon has been carefully designed to be easily understood, even by total beginners. Yet it has all the features an expert could want. Just ask it.

T.V. not included in price.

DRAGON 32

The first family computer.


ANIROG SOFTWARE
COMPUTER GAMES
OF TOMORROW
AVAILABLE
NOW!
AS SEEN ON T.V.!
HEXPERT
MOON BUGGY
SKRAMBLE
3D TIME TREK
COMMODORE 64
NEW
GALAXY
AVOID CAPTURE BY THE ALIEN MOTHER SHIPS TRACTOR BEAM AS THE FIGHTERS DIVE BOMB YOU 100 SCREENS WITH A TWO PLAYER OPTION K.B./J.S. £7.95
KONG K.B./J.S. £7.95
4 SCREENS WITH BRILLIANT ANIMATED GRAPHICS.
SKRAMBLE J.S. £7.95
6 SECTORS TO TEST YOUR SKILL.
HEXPERT J.S. £7.95
TRAIN BERT TO BE HEXPERT ON THIS 3D HEXAGONAL PYRAMID.
MOON BUGGY J.S. £7.95
MANOEUVRE YOUR PATROL CRAFT OVER GIANT POT HOLES AS YOU DEFEND THE MOON BUGGY FROM ALIEN ATTACK.
FROG RUN K.B./J.S. £5.95
A POPULAR ARCADE GAME. FUN FOR THE WHOLE FAMILY.
3D TIME TREK K.B./J.S. £5.95
SPECTACULAR 3D GRAPHICS STAR TREK GAME.
DUNGEONS K.B. £6.95
ENTER THE REALMS OF FANTASY IN THIS ROLE PLAYING GAME.
DARK DUNGEONS K.B. £6.95
2ND IN THE SERIES OF FOUR, DEFINITELY NOT FOR THE FAINT HEARTED.
NEW APPROVED GAMES FOR
THE STACK LIGHT RIFLE
INDIAN ATTACK £5.95
COSMIC COMMANDO £5.95
VENGEANCE OF ZENO £5.95
24 HR. CREDIT CARD SALES HORLEY (02934) 6083
PAYMENT BY CHEQUE, P.O., ACCESS/VISA
8 HIGH STREET HORLEY, SURREY.
Overseas 50p post & packaging
TRADE ENQUIRIES WELCOME
29, West Hill, Dartford, Kent. (0322) 92513/8

Dr. Barry Landsberg

SOME NOTES ON THE APPLE

Simple music routines for the Apple have several limitations, even given the speed of machine code. These new programs provide superior performance and are suitable for any 6502 machine which strobes a loudspeaker or cassette output.

Have you ever used an Apple II machine code subroutine to produce music, either as supplied in any Applesoft program that plays music or typed in from one of many articles on Apple music that have appeared in the past? If so, it will probably be something based on the program given in Listing 1, although it might be located at an address other than $0302. (Throughout this article, any number prefixed with a '$' refers to a hexadecimal number, while no such prefix means the number is in decimal notation — for example $0302 is the same as 770). This kind of routine may be used to make the Apple play useful, but somewhat limited, music.

In the example given, address $0300 contains a one-byte number which controls the pitch of the note produced, while $0301 contains a one-byte number which controls the duration of the note. The principle on which this routine works is that any access to the address $C030 (in Listing 1 by the command LDA $C030) changes the voltage on the Apple's loudspeaker from +5 V to 0 V, or from 0 V to +5 V, and doing this at regular intervals gives rise to the nearest

```
0302-   AD 30 C0    LDA  $C030
0305-   88          DEY
0306-   D0 05       BNE  $030D
0308-   CE 01 03    DEC  $0301
030B-   F0 09       BEQ  $0316
030D-   CA          DEX
030E-   D0 F5       BNE  $0305
0310-   AE 00 03    LDX  $0300
0313-   4C 02 03    JMP  $0302
0316-   60          RTS

PROGRAM I
```

approximation to a musical note that the Apple can perform without hardware modification — a square-wave! The rest of this short routine consists merely of timing loops. The first timing loop decrements the 6502 Y-register (which is not given an initial value by the routine!) and then whenever Y is zero, will decrement the number stored in $0301 until that is zero before exiting the routine. It can thus be seen that the value held in $0301 controls the duration of the note, and the routine will pass through the DEY instruction approximately 256 ★(T-1) times, where T represents the contents of address $0301. The exception to this is when T=0 which acts as though its value were 256.

The X register is loaded with the contents of $0300 (which we shall call P) and is decremented on each pass through the loop, and when X becomes zero, address $C030 is accessed (thus strobing the loudspeaker) and then X register is 'refreshed' with P. It is therefore evident that P controls the timing between each call to $C030, and thus controls the pitch of the note — lower values of P give rise to a higher pitch with the important exception that P=0 produce the lowest pitch possible using this routine.

This subroutine may be accessed from machine code using the following lines:

```
LDA #$00
STA $0300
LDA #$00
STA $0301
JSR $0302
  .
  .
  .
```

or alternatively from BASIC:

```
T = 0
P = 0
POKE 768,T
POKE 769,P
CALL 770
  .
  .
  .
```

Either of these two examples will produce the lowest, longest note available using this routine — approximately the G below middle C sounding for about five-eighths of a second.

To produce music from BASIC, all kinds of variations may be used to set up $0300 and $0301 with the desired values of T and P before calling $0302 — for example defining an ASCII string containing the musical information, reading DATA statements or numbering the notes and using a lookup table for the correct value of P. I prefer a combination of the last two as it makes programming and transportation (playing the tune a little higher or lower) far easier!

Having analysed the routine given in Listing 1, let us now list some of its limitations.

- The higher end of the range (ie values of P below 50) starts to become quite out of tune. For example, as the difference in frequency between P=39 and P=40 is about 2.5% in frequency while an internal of a semitone corresponds to a difference of about 5.9%, it is difficult to construct a semitone interval in this region.
- The lowest note playable is the G below middle C, which is by no means a low note.
- The longest note playable is about five-eighths of a second, and there is no satisfactory way of stringing two notes together to make a longer note (as can be done on the BBC microcomputer, for example) as even the shortest machine code routine to do this gives rise to a noticeable interruption.
- There is no natural provision for pauses, and to get them one may have to resort to tricks like ugly delay loops in BASIC, or a second machine code routine which is almost identical to that of Listing 1 except that LDA $C030 is replaced by an address which does not strobe the loudspeaker (perhaps LDA $0300) and can be called to give an accurately-timed interval of silence!

The first two limitations together mean that a tuneful range of less than three octaves can be produced, and the combination of drawbacks ensures that an ambitious desire to get the Apple to play something like the opening bars of 'Toccata and Fugue in D minor' by Bach is almost impossible. The remainder of this article will discuss a superior musical routine which overcomes the last three of these limitations, and which allows the Apple to give a rendering of the Toccata.

A NEW MUSIC SUBROUTINE

The fundamental idea of the new subroutine is simply to use the equivalent of two-byte numbers of control the pitch and timing instead of one-byte numbers as in Listing 1. Whereas Listing 1 uses essentially three counters for the timing loops (ie X, Y and address $0301), the new routine will have

to use five. Having chosen the X and Y registers to control the pitch of the note, the problem now is to choose the three other timing loop counters in such a way that no great loss of timing efficiency is effected, otherwise the usefulness of the routine will be curtailed. Furthermore, it is the first of these counters whose efficiency is the most important as it is decremented 255 times before the second counter is even accessed. In principle, use of the accumulator is by far the fastest method as commands such as ADC #$01 use only two of the microprocessor clock cycles, while the DEC (decrement) command uses five cycles even in the fastest (ie zero-page) mode! It is, in fact, possible to use the accumulator as a counter if the command LDA $C030 (which takes four clock cycles and destroys the contents of the accumulator) is replaced by BIT $C030 (which also takes four cycles, but does not affect the accumulator). I chose to use ADC #$01 as the counter step as it automatically clears the CARRY flag unless the accumulator contains the value #$FF. The zero-page addresses $FE and $FF, which are used by neither DOS or APPLESOFT BASIC, were chosen

```
0300-   A9 00      LDA  #$00
0302-   A6 FD      LDX  $FD
0304-   A4 FC      LDY  $FC
0306-   69 01      ADC  #$01
0308-   D0 08      BNE  $0312
030A-   C6 FF      DEC  $FF
030C-   D0 04      BNE  $0312
030E-   C6 FE      DEC  $FE
0310-   F0 0C      BEQ  $031E
0312-   CA         DEX
0313-   D0 F1      BNE  $0306
0315-   88         DEY
0316-   D0 EE      BNE  $0306
0318-   2C 30 C0   BIT  $C030
031B-   4C 02 03   JMP  $0302
031E-   60         RTS

PROGRAM II
```

as the final two counters, and the addresses $FC and $FD were chosen to 'refresh' the X and Y registers with the values to control the pitch as discussed earlier.

The final limitation to overcome is the inability to create a pause. The solution to this lies in calling BIT $C030 after one complete cycle of the pitch loop instead of before, as is done in Listing 1. Even for a low-pitched note at 100 Hz, this results in a delay of only 1/200 of a second as each cycle consists of two strobes to the loudspeaker; but as the routine is actually capable of going lower than 1 Hz if a value of zero is poked into addresses $FC and $FD, pauses of up to over half a second may be programmed.

The new routine is given in Listing 2, and the timing of the loops is such that it is as efficient as Listing 1 in spite of the fact that it has two-byte counters instead of one! The result of this is that the numbers needed to generate any note in Listing 1 are very close to (but not exactly the same as) those needed to generate the same note in Listing 2.

It is interesting to note that Listing 2 could have been made even more efficient, as after incrementing the accumulator we reach the command BNE $0313 and the program branches 254 times out of 253 (not 256 because of the way the CARRY flag is set). When any branch command is reached, it takes two cycles to complete if the branch is ignored, but three cycles if the branch is executed. Listing 2 was then rearranged in such a way that the BNE command was replaced by a BEQ command, either branching to the commands which decrement the zero-page addresses, or slipping through to the DEX command. This routine was indeed about 10% more efficient than Listing 2 and all other things being equal would increase the musical range by two semitones — but by decreasing the timing of the loop when it does not branch we increase the timing when it does branch. This results in the higher notes in the range sounding very rough indeed due to the larger timing inequality, and Listing 2 probably represents the best compromise between timing efficiency and musical acceptability.

INTERFACING WITH BASIC

In order to get this routine to play music from a BASIC program, it is

VALUES OF THE PITCH PARAMETER P NEEDED TO PRODUCE MUSICAL NOTES

NOTE VALUE	NOTE	VALUE OF P	NOTE VALUE	NOTE	VALUE OF P
1	C	768	25	MIDDLE C	191
2	C#	724	26	C#	180
3	D	686	27	D	170
4	D#	646	28	D#	160
5	E	610	29	E	151
6	F	576	30	F	143
7	F#	544	31	F#	134
8	G	514	32	G	126
9	G#	485	33	G#	119
10	A	458	34	A	113
11	A#	432	35	A#	106
12	B	408	36	B	100
13	C	384	37	C	95
14	C#	362	38	C#	89
15	D	342	39	D	84
16	D#	322	40	D#	80
17	E	304	41	E	75
18	F	287	42	F	71
19	F#	271	43	F#	67
20	G	256	44	G	63
21	G#	241	45	G#	60
22	A	228	46	A	56
23	A#	215	47	A#	53
24	B	203	48	B	50
			49	C	47
			50	C#	44

TABLE I

important to convert the frequency parameter P into a two-byte number in such a way that increasing P will always decrease the pitch of the note produced. To do this is not totally straightforward because the routine takes a value of #$00 in any of the counters as formally representing the number 256 and takes #$01 as formally representing zero due to the way the routine decrements the counter first and then asks if it is zero. It is lines 40 and 50 in Listing 3 which perform this conversion.

Next, we want to relate musical notes to values of P, and taking the lowest note playable as being two octaves below middle C, the values needed for each note were determined by measurements with a frequency counter and are listed in Table 1. The highest note listed is P = 44 which gives a rise to a top C#, but for the next note (top D), P = 42 would be about 0.2 semitones too low and P = 41 would be about 0.2 semitones too high. In all, a tuneful range of just over four octaves is represented. Naturally, still higher values of P give rise to even lower frequencies, but they tend to sound more like a harsh buzz than a musical note.

Once the program is written, the most tedious part is typing in the melodic data, and the most convenient way to do this is call the bottom C note number 1, the next, C#, note number 2 and so on, and to store the relevant values of P in an array. The element N(0) holds the highest number of P that can be converted to a two-byte number as described above (ie $FF00), and as this gives rise to a frequency of less than 1 Hz, asking for note zero may be used to program pauses. The only limitation to this is that values of T greater than 255 should not be used for pauses or else an audible click may be generated, but this value of T corresponds to about five-eighths of a second, and any number of consecutive pauses may be strung together! The DATA statements 2-5 in Listing 3 contain the information given in Table 1, which is read into the array N in the subroutine starting at line 1000. This way of arranging things has the advantage not only of easier programming from, say, sheet music, but also allows the melody to be played any number of semitones higher or lower (as long as all the relevant values are stored in the array) simply by inserting, for example, IF I THEN I = I + 4 at the end of line 30 in order to generate the same melody four semitones higher. The reason for the IF is that the pauses specified by I = 0 should not be transposed upwards into extraneous deep notes, or downwards to generate an "ILLEGAL QUANTITY ERROR".

Listing 3 shows a complete realisation of the last few bars of the melody 'Pizzicato' by Delibes. It is a good choice for this method of generating music because there are so many BASIC instructions between each note that the gaps between them are very noticeable. This is the first real disadvantage of this kind of routine as these gaps can be very irritating for certain pieces of music. There are two possible ways around this problem. Firstly, we could use an Applesoft compiler to speed up the execution of the program — it works very well but it is a sloppy method to use as it generates vast quantities of inflexible code. Secondly, we could write a simple machine code interface called from BASIC in order to achieve our final objective — a flexible, easy-to-install music routine. In fact, we are going to do just that!

PROGRAM III

```
2  DATA 65279,768,724,686,646,610,576,544,514,485,458,432,408
3  DATA 384,362,342,322,304,287,271,256,241,228,215,203
4  DATA 191,180,170,160,151,143,134,126,119,113,106,100
5  DATA 95,89,84,80,75,71,67,63,60,56,53,50,47,44
10  GOSUB 1000
20 FC = 252:FD = 253:FE = 254:FF = 255:TF = 256:UN = 1
30  READ I,T
35  IF T = 0 THEN  GOTO 100
40 P = N(I):P1 =  INT (P / TF) + UN:T1 =  INT (T / TF) + UN
50 P2 = P - TF *  INT ((P + UN) / TF) + UN:T2 = T - TF *  INT ((T + 1) / TF) + UN
60  POKE FC,P1: POKE FD,P2: POKE FE,T1: POKE FF,T2: CALL 768: GOTO 30
100  END
130  DATA 17,50,20,50,15,50,18,50,13,50,17,50,20,50,22,50,25,50
140  DATA 24,50,0,50,27,50,0,50,32,50,0,110
150  DATA 22,50,25,50,20,50,24,50,18,50,22,50,24,50,27,50,30,50
160  DATA 29,50,0,50,32,50,0,50,37,50,0,110
170  DATA 20,48,25,48,20,48,23,48,22,48,27,46,22,46,25,46,24,46
180  DATA 29,44,24,44,27,44,24,44,25,42,27,42,29,42,30,42
190  DATA 32,40,29,40,30,40,31,40,32,38,33,38,34,38,36,38,37,36,0,36,32,34,0,34,37,68
500  DATA  0,0
1000  DIM N(50)
1010  FOR I = 0 TO 50: READ N(I): NEXT I
1020  RETURN
```

```
PROGRAM IV
031F-  A9 40     LDA  #$40
0321-  85 FA     STA  $FA
0323-  A9 00     LDA  #$00
0325-  85 F9     STA  $F9
0327-  A0 00     LDY  #$00
0329-  B1 F9     LDA  (F9),Y
032B-  85 FC     STA  $FC
032D-  C8        INY
032E-  B1 F9     LDA  ($F9),Y
0330-  85 FD     STA  $FD
0332-  C8        INY
0333-  B1 F9     LDA  ($F9),Y
0335-  85 FE     STA  $FE
0337-  C8        INY
0338-  B1 F9     LDA  ($F9),Y
033A-  85 FF     STA  $FF
033C-  C9 01     CMP  #$01
033E-  D0 07     BNE  $0347
0340-  A5 FE     LDA  $FE
0342-  C9 01     CMP  #$01
0344-  D0 01     BNE  $0347
0346-  60        RTS
0347-  18        CLC
0348-  A5 F9     LDA  $F9
034A-  69 04     ADC  #$04
034C-  85 F9     STA  $F9
034E-  90 02     BCC  $0352
0350-  E6 FA     INC  $FA
0352-  20 00 03  JSR  $0300
0355-  4C 27 03  JMP  $0327
```

In order to generate smooth organ-like music on the Apple it is necessary to ensure that the time interval between the notes is as short as possible. However, if we want a system that can be programmed easily and is flexible, we really want to be in BASIC, slowed down even more by our reading DATA statements and our accessing an array! These conflicting demands can be met if we get BASIC to do all the work **before** it calls the music routine by storing the musical information in a reserved area of memory, and then calling a machine code routine to step quickly and efficiently through the melody. The machine code routine is necessary for this is shown in Listing 4, which is started at location $031F which is the address immediately after that of Listing 2. It takes the first four bytes from the buffer (which I chose to start at $4000), stores them in $FC-$FF and calls Listing 2, and will continue doing so until the bytes corresponding to T=0 are encountered. A word of warning here — if you wish to use this music package with a program that calls HGR2 (ie the second page of high-resolution graphics, which overwrites the area $4000-$5FFF), the music buffer should be relocated to $6000.

Once Listings 2 and 4 are in memory, Listing 5 may be run. The value of 16184 which the variable AD is initially given refers to the address $4000, and the program will convert the DATA statements in line 130 onwards into two-byte quantites and store them in memory. When this is done, the statement CALL 799 will initiate Listing 4. The DATA statements contain the opening bars of 'Toccata and Fugue in D minor' by Bach in as much glory as the Apple can possibly give to it!

Imagine we wanted to play the 'Pizzicato' keeping the staccato effect of the melody, but using the machine code routines as outlined above — surely we need to type in twice as much data to accommodate the pauses? Certainly not — if we substitute the DATA statements of Listing 3 into those of Listing 5 we get a somewhat faster 'Legato', but inserting the following line:

```
55 POKE AD,0: POKE AD+1,0: POKE
   AD+2,1: POKE AD+3,25: AD=AD+4
```

into Listing 5 will produce the desired effect.

The method I use to integrate all this into an easily used and coherent system is to set up the two machine-code routines (Listings 2 and 4) while in the HELLO program, either by POKEing the instructions one byte at a time or preferably by BLOADing them, and simply using a system based on Listing 5 with the relevant set of DATA statements for each different melody. The penalties to be paid for using this system are, first, that an area of memory from $4000 upwards (or from wherever you choose to start) is taken up with the musical information, and second, for a piece of music of any significant size, there is a considerable time delay between running the program and it actually playing the first note of the melody. This delay works out to be approximately 1/20 a second for each note or pause in the DATA statements. However, once the BASIC program has been run, any subsequent call to $031F (eg CALL 799) will initiate a replay of the melody immediately — provided, of course, that the area has not been overwritten by another part of the program. For example, having run Listing 5 once, we may load another BASIC program or even reboot by typing PR#6, and CALL 799 will still play the Toccata.

Finally, there is no reason why you should not BSAVE a very long piece of music (which might have taken the BASIC program minutes to set up) and store it on disc as part of a music library. The musical information can then be BLOADed (and even relocated if necessary), and as long as Listings 2 and 4 are in memory, any CALL 799 will immediately produce the melody. It is important that all of the melodic information is saved, as Listing 4 will not stop stepping through memory until it reaches the marker for T=0, that is two consecutive bytes each containing the value #$01.

Having demonstrated a way to string individual notes together to make a melody, next month's article develops a method of enriching the tone quality of each note and examines in more detail the strobe of the APPLE's loudspeaker.

```
PROGRAM V

2  DATA 65279,768,724,686,646,610,576.544,514,485,458,432.408
3  DATA 384,362,342,322.304.287,271.256,241.228,215.203
4  DATA 191,180,170,160,151,143,134,126,119,113.106,100
5  DATA 95,89,84,80,75,71,67,63,60,56.53,50,47,44
10  GOSUB 1000
20 AD = 16384:TF = 256:UN = 1
30  READ I,T
40 P = N(I):P1 =  INT (P / TF) + UN:T1 =  INT (T / TF) + UN
50 P2 = P - TF *  INT ((P + UN) / TF) + UN:T2 = T - TF *  INT ((T + UN) / TF) + UN
60  POKE AD,P1: POKE AD + 1.P2· POKE AD + 2.T1: POKE AD + 3.T2:AD = AD + 4
70  IF T THEN 30
100  CALL 799: END
130  DATA 46,24,44,24,46,672,0,240,0,240,44,240,42,60,41.60,39,60,38.120,39.480
140  DATA 0,240,0,240,34,24,32,24,34,672,0,240,0,240,29,120,30,120,26,120,27.480
150  DATA 0,240,0,240,22,24,20,24,22,672,0,240,0,240,20.60,18,60,17,60,15,60,14,120,15,480
160  DATA  0,240,0,240,3,960,14,240,17,240.20,240.23,240,26,240.29.480
170  DATA 0,240,22,240,17,240,19,480,0,240
180  DATA 26,72,27,48,29,48,26,48,27,48,29,48,26,48,27.48,29,48,26.48,27,72,29,72
190  DATA 30,48,32,48,29,48,30,48,32,48,29,48,30,48.32.48,29,48,30,72,32,72
200  DATA 34,48,35,48,32,48.34,48,35,48.32,48,34,48,35,48.32,48,34,72,0,240
210  DATA 38,72,39,48,41,48,38,48,39,48,41.48,38,48,39.48,41.48,38,48,39,72,41,72
220  DATA 42,48,44,48,41,48,42,48,44,48,41,48,42,48,44,48,41,48,42,72,44,72
230  DATA 46,48,47,48,44,48,46,48,47,48,44,48,46,48,47,48,44,48,46,48,0,240
240  DATA 46,48,44,48,47,48,41,48,44,48,47,48,41,48,42,48,46,48,39,48,42,48,46,48,39,48
250  DATA 41,48,44,48,37,48,41,48,44,48,37,48,39,48,42,48,35,48,39,48,42,48,35,48
260  DATA 37,48,41,48,34,48,37,48,41,48,34,48,35,48,39,48,32,48,35,48,39,48,32,48
270  DATA 34,48,37,48,30,48,34,48,37,48,30,48,32,48,35.48,29,48,32,48,35,48,29,48
280  DATA 30,48,34,48,27,48,30,48,34,48,27,48,29,48,32,48,26,48,29,48,32,48,26,28,0,240
290  DATA 3,660,14.240.17,240,20,240.23,240,26,240,29,240,32,240,35,528
300  DATA 34,48,32,48,30,48,29,48,27,48,26,48,24,48,26,96,22,96,26,96,29,48,32,48
310  DATA 30,24,32,24,30,24,32,24,30.24,32,24,30,24,32,24,30,96,29,96,27,48,0,48,27.256
500  DATA  0,0
1000  DIM N(50)
1010  FOR I = 0 TO 50: READ N(I): NEXT I
1020  RETURN
```

sinclair special

6

Inside...

Setting new standards in educational software with Sinclair-Macmillan

Plus six other learning programs

TODAY, LEARNING IS A NEW GAME

Subsidised microcomputers are now commonplace as teaching aids for the very youngest children and the ZX Spectrum is prominent amongst those micros at use in schools.

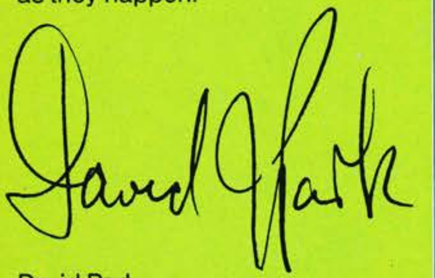
In the relatively short time that the Spectrum has been at work in the classroom, two questions have been answered. Yes: with the right software, the micro can and does teach effectively and thoroughly (and gives teachers more time to devote to individual pupils). Yes: young children think little of working rapidly and successfully, with a screen and keyboard, on even quite complex subjects.

In this Sinclair Special we reveal a range of educational software specifically designed to make full use of these advantages. The programs produced by Sinclair in collaboration with Macmillan Education are fascinating. They deal imaginatively and most effectively with early reading skills and take a truly refreshing approach to basic science.

In the Blackboard range we've programs which bring a light-hearted clarity to the tricky matters of spelling and punctuation.

These programs are designed for use both at home and in the classroom. Each program is accompanied by full documentation which gives parents helpful advice and guidance on the educational objectives.

The programs covered on these pages represent only a fraction of the full and fast-growing list of Spectrum software. Be assured we'll keep you in touch with new developments as they happen.



David Park
Education Marketing Manager

NEW WAYS TO LEARN WITH THE ZX SPECTRUM™

Programs from Blackboard Software

The new range of educational programs from Blackboard Software makes learning an enjoyable process by involving the child in a game which teaches as it entertains.

Each program has a step-by-step example section and gives correct answers after a number of attempts. Vocabulary changes can be made, allowing each program to keep pace with the child's development. This flexibility can also be used in the classroom to cater for children of differing ability.

The instructive and colourful games which follow the successful completion of each group of sentences provide useful practice in letter recognition and increase familiarity with the Spectrum keyboard.

All programs are written for the 48K RAM Spectrum.

Alphabet Games

Three games of letter recognition (using either upper or lower case) to help children learn the alphabet and find their way round the computer keyboard.

Alphagaps – The full alphabet is displayed, along with a second, incomplete version. The child must fill in the missing letters.

Random Rats – Press the letter key that is displayed on the gun to destroy the rats which have invaded the cellar!

Invaders – Stop little green men from landing on Earth by pressing the appropriate letter.

Early Punctuation

While an animated matchstick man marches above displayed sentences the child must decide which punctuation mark is missing and where to insert it. At the touch of a key the matchstick man drops the mark into place. After successful completion of every sentence in the exercise, light relief comes in the form of a bottle-shooting game!

The Apostrophe

As each sentence is displayed, a bird appears with a worm in its beak. The keyboard is used to move the bird and drop the worm into the correct place for the apostrophe. When ten sentences have been corrected, the Grub Game is displayed. Press the correct character to change the grub into a butterfly...before it munches through a flower!

Capital Letters

A program to teach the use of capital letters. Sentences incorporating proper nouns and sentences without opening capitals are displayed. The child inserts the correction by guiding an animated figure to the appropriate letter.

For each correct answer an apple grows on a tree. After ten correct answers the child's skills in recognising letters and using the Spectrum keyboard are needed to save the apples as they fall to the ground.

Speech Marks

A comprehensive program including sentences with one or two sets of speech marks ("inverted commas") and exercises in both direct and reported speech.

Using the Spectrum keyboard, a cursor is used to guide speech marks to the correct position. The program offers three levels of difficulty, with full examples for each section. Guide Max the mouse through a maze, after the correct completion of five sentences from each section, but beware of Persian cats!

Castle Spellerous

A spelling game with ten levels of vocabulary, including words with silent first letters, double letters and other difficult words. The Princess has been captured and carried off to Castle Spellerous. Helped by ten soldiers, the child can attempt a rescue by giving the right answers. Part of a siege tower is built for each correctly spelt word. Mistakes are costly – the wicked wizard appears as a vampire bat, turning the men into frogs, butterflies and bats!

When ten words are spelt correctly the rescue begins and the wizard takes flight.

SINCLAIR + MACMILLAN: A NEW DIMENSION IN EDUCATIONAL PROGRAMS

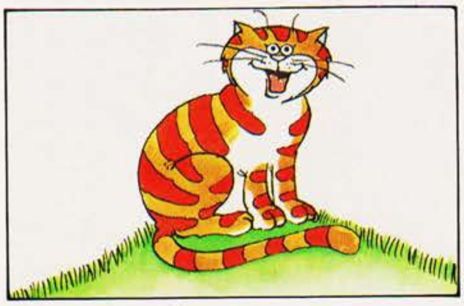
Sinclair have joined forces with Macmillan Education to produce a completely new and different range of educational software. The results so far can be seen in these exceptional programs.

The Learn to Read series is derived from Macmillan Education's best-selling primary school reading scheme, Gay Way. It offers a unique opportunity for parents and teachers to participate in the child's first experience in reading.

Macmillan Education's Science Horizons is one of Britain's most successful school science schemes. Each program concentrates on key scientific ideas and, through simulation of real life, makes the learning process entertaining and enjoyable.

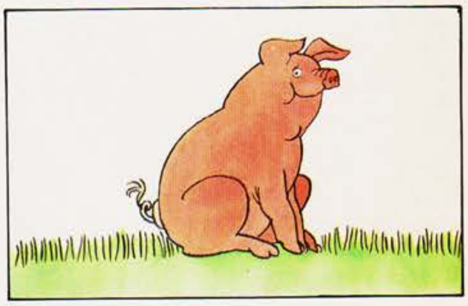
Learn to Read 1

Learn to Read 1 is designed for children who are just beginning to read. It is in four parts, each of which develops skills central to the reading process – letter recognition, sight vocabulary, early spelling and memory. The program is full of colour and fun and children will enjoy learning to read as they meet the animal characters – Ben the dog, Jip the cat and their friends.



Learn to Read 2

Learn to Read 2 extends the fundamental reading skills practised in the first program, as well as encouraging logical thinking. The child's vocabulary is gradually built up as new words such as "red," "green," "car," "ship" and "bus" are introduced. In addition, Learn to Read 2 features an attractive 'reward' system enabling children to see their achievements grow.



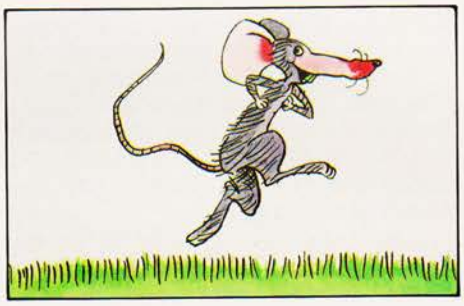
Learn to Read 3

Learn to Read 3 builds on the child's progress so far, so that he or she can gain the confidence to move on through the complex reading process. Learn to Read 3 features four different activities, all of which are colourful and lively. Further vocabulary is introduced until the child is reading more than 30 words.

Learn to Read 4

Learn to Read 4 is the alphabet program in the Learn to Read series.

Using various stimulating activities the program gives the child plenty of practice in working with the alphabet – matching initial letters to words and pictures and spotting missing letters. These exercises build familiarity with simple sequences within the alphabet.



Learn to Read 5

Learn to Read 5 teaches positional language – often difficult to understand and remember – by using words and phrases such as "behind" and "in front of," "inside" and "outside."

The program first demonstrates the meanings of the words using clear pictures. It then tests the child's understanding of the words in two lively games.

Cargo

Set sail around the world. Choose your ports of call – New York, Tokyo, Belem, Helsinki – then the real challenge begins! You must reach your destinations safely, weathering storms on the way. But first, load your cargo – using all your knowledge and skill. Poor loading can mean capsizing and sinking. Your rank, if not your life, is always at stake!

Glider

Be a glider pilot! The glider models real-life gliding conditions so that you can learn through experience. As the pilot you must consider the time of day, the amount of cloud cover and the kind of terrain below you in order to find the up-currents of air that will keep you airborne. Try to fly as far as possible and, when you are high enough, navigate your way back to your home airfield and land safely – if you can.

Survival

Discover what it is like to be an animal in the wild! Be a lion stalking your prey, escaping human hunters. Or be a hawk, mouse or even a butterfly, searching for food and avoiding predators.

Survival models the natural world and brings to life hazards that different creatures must face in their struggle to stay alive.

Magnets

With an army of small magnets you set out to conquer the powerful supermagnets of your opponent. You have one weapon – your forces of magnetic attraction and repulsion.

The strategy is simple: attract smaller magnets to build strength to repel the super-magnet. When cornered, just turn your poles on your enemy and see what happens!

Learn to Read and Science Horizons programs are written for a 48K RAM Spectrum, and are supplied complete with comprehensive supporting documentation for parents and teachers.

ZX INTERFACE 2
THE NEW ROM CARTRIDGE AND JOYSTICK INTERFACE

Now available in shops

Loads programs instantly
Takes two joysticks
Just plug-in and play

The ZX Interface 2 is the latest new peripheral for the ZX Spectrum system. It enables you to use new ZX ROM cartridge software: plug-in programs that load instantly. There are ten terrific games already available on cartridge. ZX Interface 2 also allows you to use one or two standard joysticks without the need for separate special interfaces.

To use new ZX ROM Cartridge programs, just connect Interface 2 to the rear of your Spectrum or Interface 1 and plug in the cartridge of your choice. Switch on and the program is then loaded, ready to run!

You can use any joystick that has a 9-way D plug. Use one or two of them for extra fun with suitable ZX ROM cartridge or Sinclair cassette programs – or with dozens of other Spectrum programs.

ZX MICRODRIVE/ ZX INTERFACE 1

The ZX Microdrive System is unique. This compact, expandable add-on system provides high-speed access to massive data storage. With just one Microdrive and a ZX Interface 1 you'll have at least 85K bytes of storage, the ability to LOAD and SAVE in a matter of seconds, the beginnings of a local area network of up to 64 Spectrums *and* a built-in RS232 interface. The cost? Less than £80.

How to get ZX Microdrive and ZX Interface™ 1

Spectrum owners who bought direct from us by mail order have been sent full details. Order forms are being mailed in strict rotation. If you haven't yet received your order form please bear with us. We're making good progress in meeting the huge demand.

If you didn't buy your Spectrum by mail order, send us the form at the bottom of this page and we'll add your name to the mailing list.

How to order

Complete the appropriate sections on the order form below. Note that there is no postage or packing to pay on software purchases. Orders may be sent FREEPOST (no stamp needed). Credit card holders may order by phone, calling 01-200 0200, 24 hours a day. Please allow 28 days for delivery.

™Sinclair, ZX Spectrum, ZX Microdrive, ZX, ZX Interface are trade marks of Sinclair Research Ltd.

sinclair™

Sinclair Research Ltd, Stanhope Road, Camberley, Surrey, GU15 3PS. Telephone: (0276) 685311.

To: Sinclair Research Ltd, FREEPOST, Camberley, Surrey, GU15 3BR.

ORDER FORM

Section A: Hardware

Qty	Item	Code	Item Price £	Total £
	ZX Interface 2	8501	19.95	
	ZX Spectrum – 48K	3000	129.95	
	ZX Spectrum – 16K	3002	99.95	
	Postage and packing: orders under £90	0028	2.95	
	orders over £90	0029	4.95	
			TOTAL £	

Section B: Software

Qty	Item	Code	Item Price £	Total £
	E10/S Learn to Read 1	4309	9.95	
	E11/S Learn to Read 2	4310	9.95	
	E12/S Learn to Read 3	4311	9.95	
	E13/S Learn to Read 4	4312	9.95	
	E14/S Learn to Read 5	4313	9.95	
	E15/S Cargo	4314	9.95	
	E16/S Glider	4315	9.95	
	E17/S Survival	4316	9.95	
	E18/S Magnets	4317	9.95	
	E19/S Early Punctuation	4318	7.95	
	E20/S Speech Marks	4319	7.95	
	E21/S Apostrophe	4320	7.95	
	E22/S Capital Letters	4321	7.95	
	E23/S Castle Spellerous	4322	7.95	
	E24/S Alphabet Games	4323	7.95	
			TOTAL £	

*I enclose a cheque/postal order made payable to Sinclair Research Ltd for £ ______

*Please charge my Access/Barclaycard/Trustcard account no:

*Delete/complete as applicable.

Signature

Mr/Mrs/Miss

Address

COT 401

(Please print)

ZX Microdrive™ information request

Please add my name to the Microdrive Mailing List *and* send me a colour brochure with full specifications of ZX Microdrive/Interface 1 ☐ *(tick here).*

Free Sinclair ZX™ Software Catalogue

Please reserve my priority copy of the January 1984 Sinclair ZX Spectrum Software Catalogue ☐ *(tick here).*

Please use the form above to send your name and address.

Don Thomasson

ELECTRON UNDER THE MICROSCOPE

Inevitably the Electron has been described as a cut-down BBC. We look at the latest price of Acorn hardware to see if this is a fair verdict.

A person who buys a racehorse want to know its pedigree, and if none is available the value of the animal will be considerably reduced. The Electron has a respectable family background, and that will enhance its chances of success, though it may not have all the advantages which were exhibited by its ancestors.

It is tempting to review it on a basis of comparison with the BBC computer, with which it is broadly compatible, but that would not be helpful to those unfamiliar with the earlier machine. On the other hand, some comparisons are unavoidable, because certain features have clearly been included for the sake of compatibility, and would be rather mystifying if treated solely in isolation.

THE EQUIPMENT

Packed in the familiar kind of polystyrene foam box, the Electron emerges sideways, rather than lifting out horizontally, as if it was asserting a claim to be different right from the start. Its removal reveals a little bag of moisture-absorbing pellets, an encouraging indication of attention to detail, perhaps more necessary in a device which has clearly travelled far and through moist climates ("Assembled in Malaysia" is embossed in the plastic). The main unit is chunky in appearance, its case made of hard and tough plastic, more robust than some cases and entirely practical. The maximum dimensions are roughly 13½" wide, 6¼" deep, and 2¼" high.

A second unit plugs directly into a three-pin mains socket, which may be inconvenient for sockets with limited clearance round them. This is not a power supply, merely an isolating transformer delivering 18-19 V AC at 14W. There is no on-off switch, the need for one being reduced by the fact that a single supply is involved, so there is no harm in unplugging the transformer output from the main unit. This connection is made at the right-hand end of the main unit.

At the left hand end are four connectors. There is the UHF output to a television set, and the DIN socket for the cassette recorder. More surprising are the video and RGB outputs — surprising because a colour monitor would probably cost more than the Electron itself. However, the quality of the display is enough to justify the use of a monitor, being slightly restricted by the performance of even a good television set.

Incidentally, the TV lead is supplied, the other leads are not. The cassette lead required is exactly like that used by the BBC computer, motor control being available if the recorder can use it.

At the back of the unit, protected in a recess and by a slip-on plastic cover, is the remaining connector, a 50-way double-sided edge connector formed by an extension of the main printed circuit board. This carries all address and data lines and the principal CPU control lines, but no details of the pin-outs were found in the manual. An interesting and useful provision was a pair of threaded inserts bonded into the case on either side of the connector. By allowing extension equipment to be secured firmly, these would ensure against uncertain connections.

The remaining contents of the box were an introductory tape and two manuals, one an introduction to BASIC. The tape contained good and bad items, ending with a close approximation to the 'Animation' program published in *Computing Today* for December 1982. The main manual was reasonably comprehensive, but unfortunately lacked a subject index, which is a pity, since there are 290 pages to search for a particular item. There was even a section on assembler programming, with a list of 6502 instructions.

THE KEYBOARD

Since it is the main interface with the user, the quality of a keyboard can have a considerable influence on the value of a machine. The Electron keyboard produced few complaints. The 54 keys, arranged mainly in a standard QWERTY layout, were as good as any yet seen in respect of touch: if anything a little light and over-ready to respond. There was the usual rattle in fast action, perhaps preferable to a 'beep' as an indication that a key had been pressed.

Most of the character keys have three meanings choice of a particular meaning depending on the use of the Shift, Control and Function keys. This provides for 94 characters, for function keys, and 29 keywords, plus the four arrow keys and Copy for editing. Single-function keys are Break, Escape, Delete, Return, the two Shift keys, Control and Function.

In this way, a four-row keyboard is given something near the maximum number of characters and functions which it could contain. The only problem arising from that is the closeness of the editing keys and Break, which caused occasional embarrassment. However, since Function/0 gives OLD with Return, recovery was simple, that combination restoring the less desirable effect of Break.

FIRST TESTS

The tape recorder still contained a BBC tape, and the temptation was

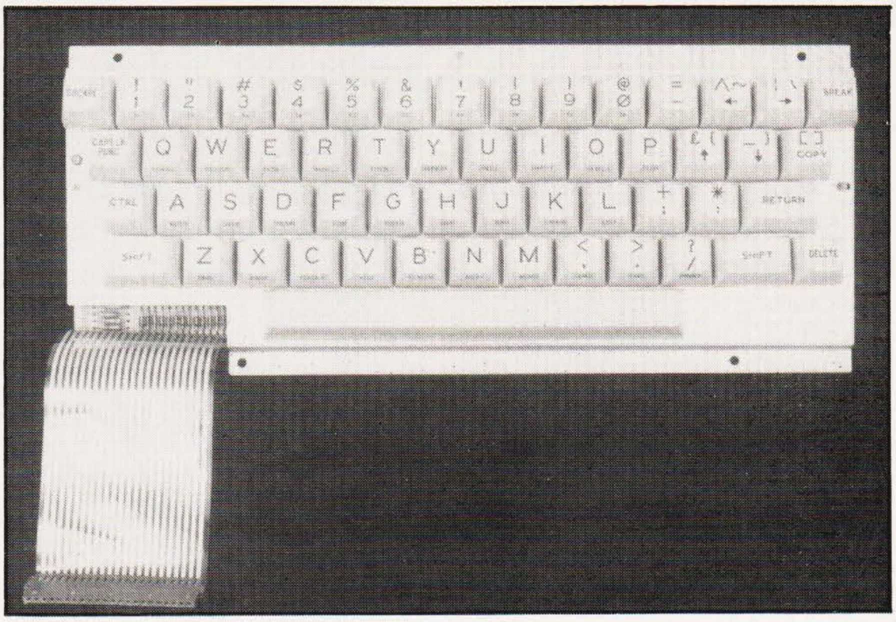
too much. Would the Electron load such a tape? It did, and the fairly simple programs on the tape were executed almost as on the BBC machine, though at a noticeably reduced speed and with a few differences.

This allowed a number of familiar programs to be run, which made it clear that the principal difference was the lower running speed, confirmed by the Bench Marks, as shown in the tabulation.

It was also noticed that graphic displays were clearer and sharper than they had been on the older machine, to the degree that vertical lines were noticeably thinner than horizontal lines — no doubt an effect of the reduced bandwith of the TV receiver.

It was time to dig a little deeper.

BENCHMARK	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	Average
TIME	1.6	8.1	18.1	24.3	25.4	36.9	57.3	153.2	40.6



THE INSIDE STORY

You've seen someone poking into a newly-opened envelope looking for something they had expected to find there? Looking inside the Electron produced the same sort of reaction. To the right hand side of the box there was a small power supply unit producing 5 V, apparently on a switched-mode basis. Then there was another printed circuit board measuring about 9½" × 5", which carried 18 integrated circuits. And that was the lot.

Perhaps there was no real reason for surprise. After all, the ZX81 manages with only four main components, but this was a rather more versatile machine, with 32K of store carried internally. Where was it?

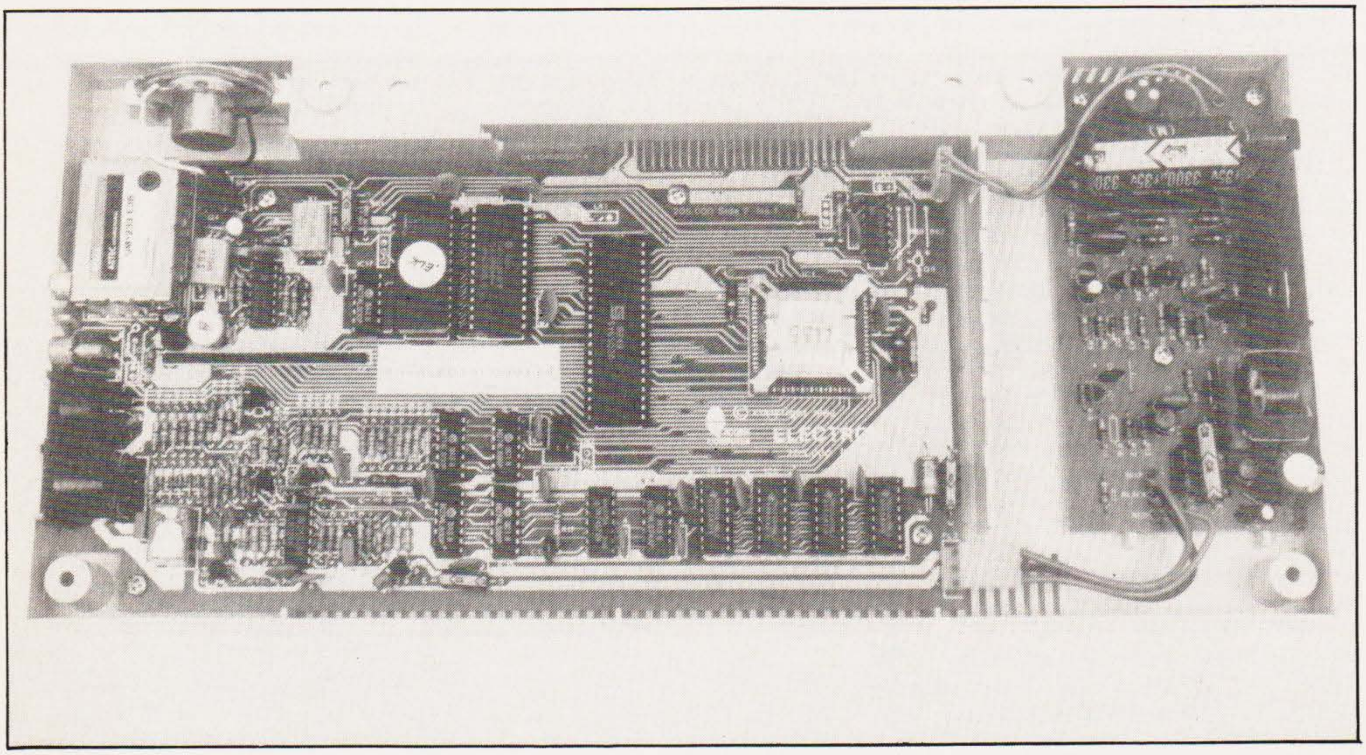
The answer was another surprise. There were four 4164 RAMs, each providing 64K x 1 bit storage. The lack of speed was immediately explained, because each byte access would need two accesses to the RAM, each picking up or depositing one nibble.

Further investigation identified nine ICs of small or medium scale integration types, the processor, two 16K ROM chips, an LM324 in the sound system, and one impressive 64-pin component of the type that sits in a square socket. This last item must be kept very busy, as it must handle keyboard and display transactions, cassette operations, and the clock, the graphics.

It is understood that even this is not the end of the story, as more components may yet be combined to reduce the chip count further.

FIRMWARE

The operating system identified itself as being of issue 1.00, and a quick scan through some of the routines revealed the expected similarity to the BBC operating system 1.20, even to the odd little quirk which omits the first byte of each page when clearing RAM dur-



ing initialisation. This preserves the Return from Interrupt that is set in &0D00 as the response to a non-maskable interrupt. It seems a pity that a tidier solution was not possible.

The data area from C000 upwards was a little smaller, but the overall 16K of operating system RAM seemed quite fully occupied, so it is possible that some hardware functions have been replaced by software, though it was not possible to check this.

The BASIC system also resembles that of the earlier machine, but with some functions omitted or modified.

Once again, pedigree counts, for here is a firmware system which has been developed progressively over a long period. It is not without fault, but the more serious bugs have been eliminated, and no evidence of significant problems was found during the tests.

THE BASIC

The Acorn version of BASIC is comprehensive in scope, and is broadly compatible with the Microsoft version except in respect of special extensions. If it has a fault, it lies in the need to set up long and complex strings of parameters, this being especially annoying when some of the parameters have no meaning, being 'reserved for future expansion'. However, if you want versatility, this is an inevitable consequence, since multiple options means multiple parameters.

The usual repertoire of familiar words and functions is offered, some doing more than the manual specifies. For example, AND can be used on a bit-by-bit basis, as well as for combining logic conditions, but only the latter use was mentioned. Useful tools include AUTO, RENUMBER and TRACE, while LISTO offers various listing formats. It is unfortunate that AUTO provides a space after the line number during line entry, but enters no space in the stored line, which led to a need for some patient editing to tidy up listings, but the fact that so minor a point was noticeable speaks volumes for the satisfactory nature of the system as a whole.

Some words may wrinkle a few brows. ADVAL, for example, makes sense as a means of reading the analogue-digital converter value on the BBC computer, but its function here is to report on the state of the sound channel buffers.

Perhaps the most obvious omission from the normal BASIC vocabulary is CONTINUE. Once a program has been stopped, it can be restarted by a GOTO with a carefully-chosen line number, but that is not quite the same thing.

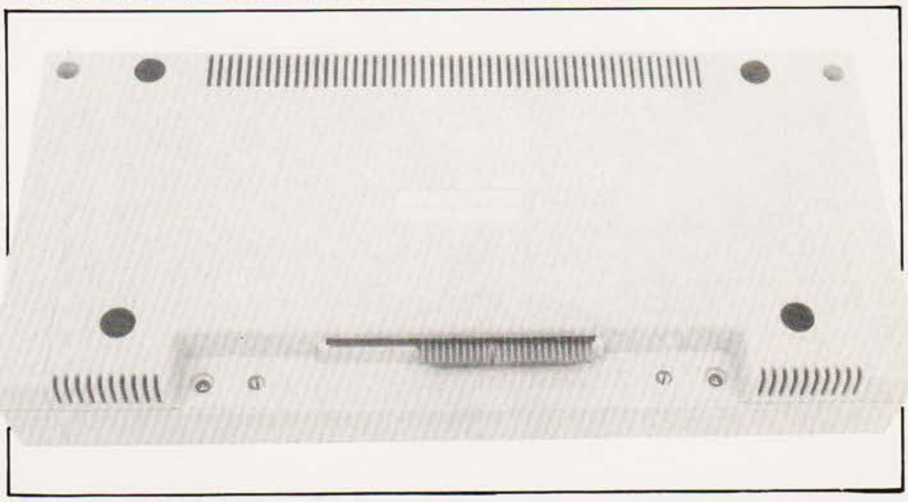
The most obvious added provision is the Assembler function, which is given particular attention in the Electron manual. The complete listing of 6502 instructions must be unique in any manual for a small computer. The facilities include some not specified for the BBC machine, and these allow a byte, word, double word or string to be set up. Some addressing modes, however, are simplified to match the Assembler characteristics.

GRAPHICS AND COLOUR

The Electron offers seven working modes, the teletext mode being omitted (it would need about five extra chips to be implemented). The remaining modes offer a wide choice of compromises between display complexity and space for programs, with 20, 40 or 80 column text on 25 or 32 lines, three levels of graphics definition, and 2, 4 or 16 colours. It is perhaps a pity that the corresponding data on available store space is not coupled with the specification of modes, but that has been corrected by the table provided here.

One point is clear. Without the very economical teletext mode, the maximum program space is less than the maximum available in the BBC machine. A thoughtless attempt to load 'Planetfall' resulted in the program beginning to appear on the screen when available space was exceeded, an interesting but not very useful effect. However, 20K is available in mode 6, which should be ample for most purposes. The 8K-odd left by modes 0-2 is more restricting.

The limitation is important, because it means that the most versatile colour/graphics combinations are only available for use with the



shorter programs.

Apart from the simplified MOVE and DRAW commands, there is a full repertoire of PLOT variants, 64 in all, and these include some fill functions not listed in the BBC manual. (They do work on the BBC machine, though! Try 72-79, which fill laterally.)

One exercise successfully carried through was the creation of a 'mimic diagram' for a model railway layout. The form of the diagram was specified by 56 sets of five parameters, some sets defining straight stretches and some defining curved segments. The trains were shown by colour contrast. Not surprisingly, the program was quite sizeable, with many lines of more than 80 characters, but it was stored in a sufficiently economical way to be compatible with mode 2.

It is truism to say that the capacity of the colour/graphics system is limited only by the user's imagination. One of the items on the demo tape was a picture of an island, complete with palm tree, and the surrounding sea was represented by moving waves . . .

SOUND

The sound system of the Electron was a trifle limited. Yes, there were three simultaneously available tone channels and a noise channel, but there were only two dynamic levels, on and off. There was an ENVELOPE command, so named for the sake of compatibility, but while it produced frequency modulation it had no effect on the sound envelope. The relevant parameters had to be set up, as zeroes, but they had no practical meaning.

Coupling these points with the use of a tiny internal loudspeaker and a slight uncertainty in some of the musical pitches, the sound system had to be regarded as a gimmick rather than as something which could be used seriously. It will make 'space ship noises', but only the tone deaf would accept its music. However, perhaps better systems have encouraged an expectation of better performance. Frankly, one machine which worked through the television set loudspeaker was so much better in a number of respects that it has set a subconscious standard that most other machines fail to attain.

Mode	Graphics	Colours	Text	Program store
0	640× 256	2	80× 32	&21F0= 8688 bytes
1	320× 256	4	40× 32	&21F0= 8688 bytes
2	160× 256	16	20× 32	&21F0= 8688 bytes
3	—	2	80× 25	&31F0= 12784 bytes
4	320× 256	2	40× 32	&49F0= 18928 bytes
5	160× 256	4	20× 32	&49F0= 18928 bytes
6	—	2	40× 25	&51F0= 20976 bytes

Table 1. Summary of Mode Characteristics

COMPARISONS

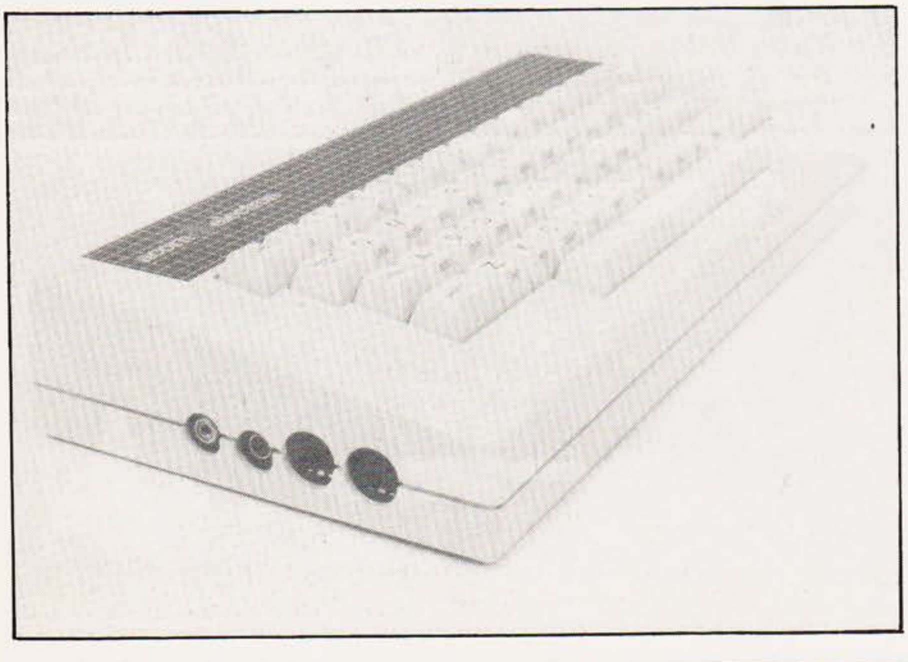
Odious though they may be, comparisons between the Electron and its forebears are both inevitable and necessary, if only to judge how the machines stand in relation to each other.

The Electron is a cut-down version of the BBC B machine, implemented with great hardware economy at the cost of some loss of operating speed and fringe facilities. It has twice the store of the A machine, but lacks the economical teletext mode, which would make a further 7K of store available in relation to mode 6. It cannot be expanded to B model standard as the A model can.

The key question is going to be the cost and capability of the extension system. In its basic form, the Electron is a completely viable machine, up to a point, but it lacks the wherewithal to drive a printer or to communicate with external devices directly. There are clear indications that it is seen as a stepping stone to the BBC machine. Users are urged to write their programs so that they are compatible with the more complex computer, which makes sense if there is a later transfer to the larger device. The possibility of exchanging programs between the different machines, subject to some limitations, is an interesting feature in itself. (Incidentally, no problems arose in loading programs saved on one machine and read into the other, but the Electron seemed to be slightly more reluctant to read its own output until the precisely correct volume level was found!)

In its general price bracket, the Electron has no competitors (yet — Ed.) It is a fully developed machine, which — unlike several other types — worked perfectly from first switch-on. It has an impressive performance, with virtually no evident shortcomings. Its pedigree is an added recommendation. From a purely personal point of view, there was a tinge of regret that it used the 6502, rather than the Z80, because the simpler processor tends to need rather more complex routines and to suffer some limitations: but the 6502 has its staunch adherents, who no doubt will read these words with fury. It would certainly be more difficult to incorporate an Assembler for the Z80 . . .

A key question regarding the merit of any computer is whether you would feel that you could recommend it to your friends. For many of the machines which have appeared during the past two years, it has been necessary to feel reservations. With the Electron, the only reservation in that its ultimate value must depend on the provision of extensions. For those who only want a minimum system, it seems almost ideal. It has already been recommended to several interested persons who come into that category.



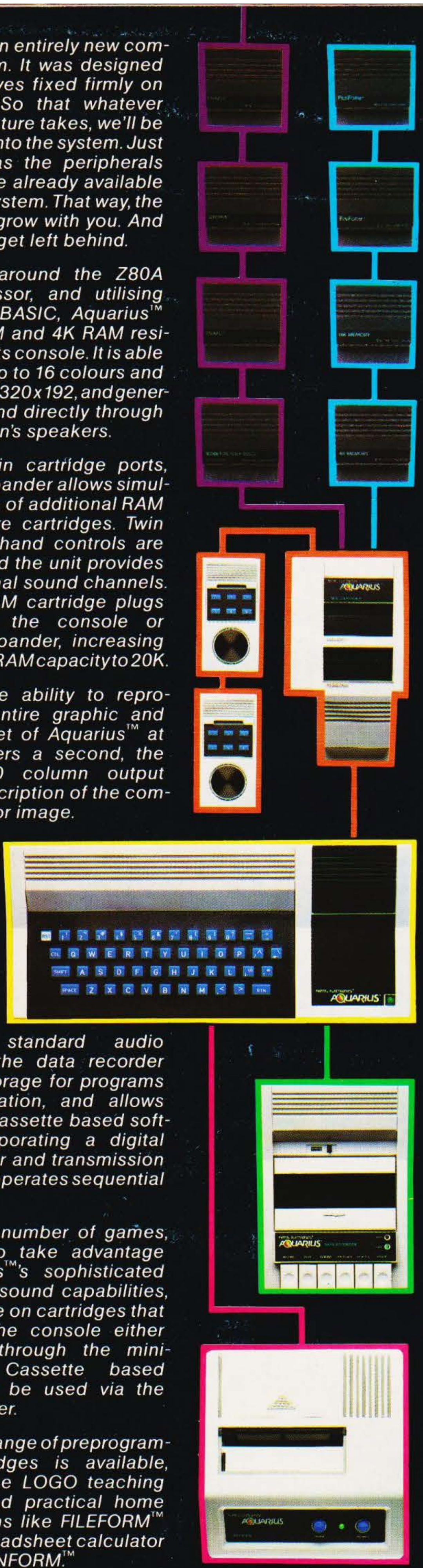

SPECTRAVIDEO
PERSONAL COMPUTER SV-318

- This is an entirely new computer system. It was designed with both eyes fixed firmly on the future. So that whatever shape the future takes, we'll be able to fit it into the system. Just as simply as the peripherals and software already available fit into the system. That way, the system will grow with you. And you'll never get left behind.

- Based around the Z80A microprocessor, and utilising Microsoft™ BASIC, Aquarius™ has 8K ROM and 4K RAM resident within its console. It is able to provide up to 16 colours and resolution of 320x192, and generates its sound directly through the television's speakers.

- With twin cartridge ports, the mini-expander allows simultaneous use of additional RAM and software cartridges. Twin disc game hand controls are included and the unit provides two additional sound channels. The 16K RAM cartridge plugs into either the console or the mini-expander, increasing Aquarius™'s RAM capacity to 20K.

- With the ability to reproduce the entire graphic and character set of Aquarius™ at 80 characters a second, the printer's 40 column output allows transcription of the complete monitor image.



- Using standard audio cassettes, the data recorder provides storage for programs and information, and allows the use of cassette based software. Incorporating a digital tape counter and transmission indicator, it operates sequential searching.

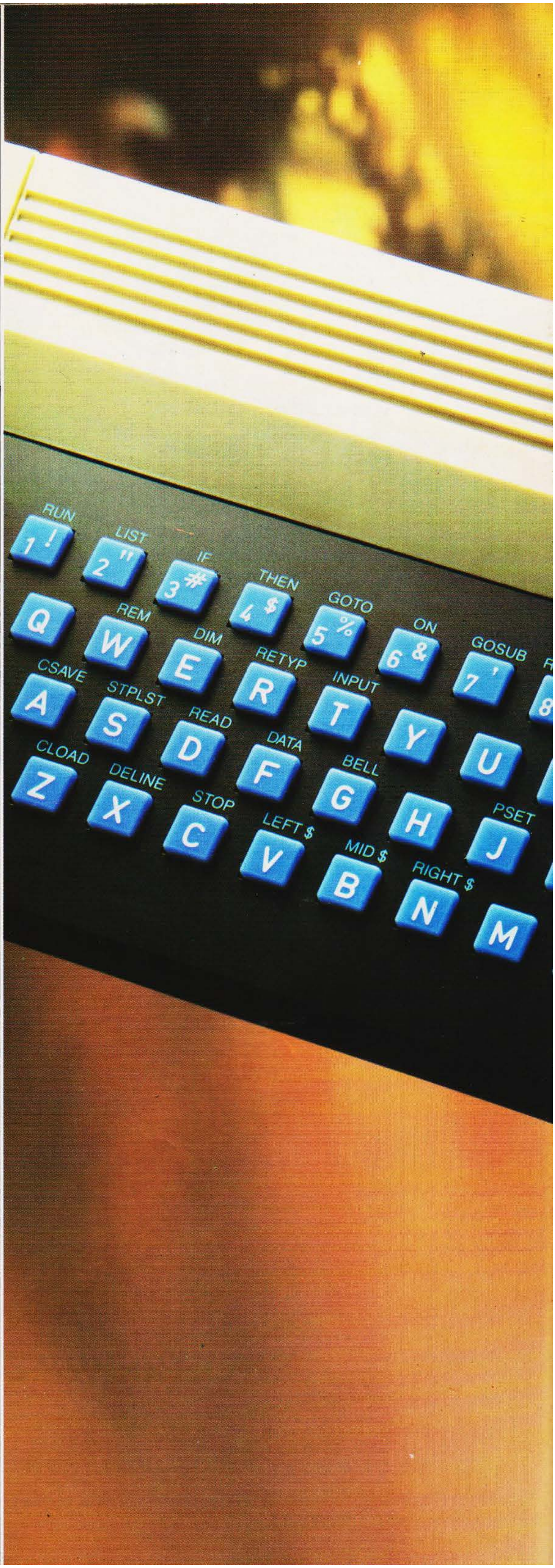
- A large number of games, designed to take advantage of Aquarius™'s sophisticated colour and sound capabilities, are available on cartridges that plug into the console either direct, or through the mini-expander. Cassette based games can be used via the data-recorder.

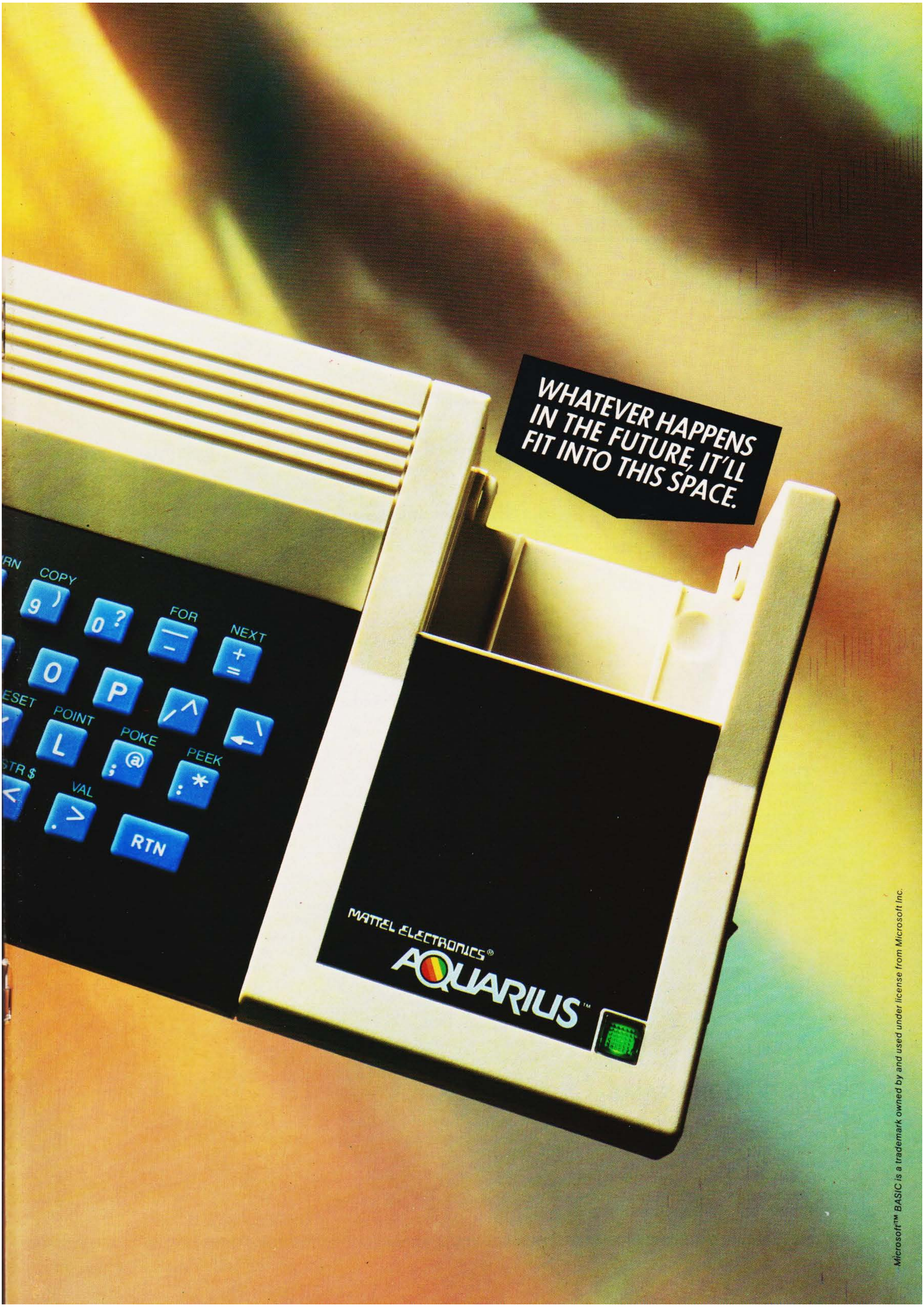
- A wide range of preprogrammed cartridges is available, including the LOGO teaching program and practical home data systems like FILEFORM™ and the spreadsheet calculator package, FINFORM.™

AQUARIUS™
HOME COMPUTER SYSTEM

WITH AQUARIUS, YOU WON'T GET LEFT BEHIND.

MATTEL ELECTRONICS®




WHATEVER HAPPENS IN THE FUTURE, IT'LL FIT INTO THIS SPACE.
COPY
FOR
NEXT
POINT
POKE
PEEK
VAL
RTN
MATTEL ELECTRONICS®
AQUARIUS™
Microsoft™ BASIC is a trademark owned by and used under license from Microsoft Inc.

A. C. Ellis

NAMED FILES FOR NAS-SYS

Anyone who has used the Nascom 2 will know that, although the BASIC interpreter supplies a single character field for all its files, NAS-SYS does not. Consequently if you have a utilities tape with many object or machine code programs, all small and on the same tape, it is hard to get the right one. The usual process is to 'verify away' the unwanted files, counting as you go, then read the correct one. Wouldn't it be simple if you could tell NAS-SYS to find a specific file?

This is now possible using the program given here. The user is supplied with a 16 letter field in which to name the file. All characters within the field are used and even spaces are significant; thus the two file headers below are different:

```
( SPACE   TEST  )
(SPACE TEST     )
```

If two files have the same file name then the monitor will load the first it sees.

There are two exceptions to this search for a specific file name. These are:
(1) If the first character in the name field of the command is * it will read or verify the next file it comes to.
(2) If during the search, a file with * as the first character in its name is encountered, it will be processed irrespective of what the monitor is looking for, so that:

```
VERIFY FILE (*               )
```

will load the next file it comes to. A program saved by

```
WRITE A FILE (*               )
```

will be loaded come what may.

These commands offer a form of priority level to the files they are used on.

WRITING TO THE NAME FIELD

As already stated, the name field is 16 letters long. Any character may be written into it with the exception of control codes other than backspace and carriage return. These are used for editing and field termination respectively. When the name field is full a CR will be inserted automatically.

The only other information needed is for the write command

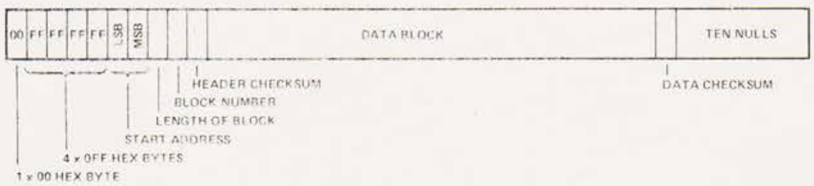


Fig. 1 Format of the data blocks

and that is the start and finish addresses for the write command. The data needed is in Hex format, and it is entered under NAS-SYS. Any non-valid number (not 0-9 or A-F or greater than FFFF) will result in an error message and another prompt.

SEARCHING

During a search for either verify or read the message "Files encountered" will be displayed. Any header encountered will then be written up on the screen; if it is the one required you will see the

Fig. 2 File structure of a headed program

normal 'NAS-SYS type' load commence.

If you wish to terminate the search before completion, turn off the tape and enter four escapes. That will return you to NAS-SYS command level.

As with the normal commands both the serial port and the keyboard are scanned, so avoid typing during tape operations as this will cause errors.

Named files may be read by unmodified commands, but not the other way around. This is because the program will be looking for a header before the data is read: with no header, it won't look for the data.

GENERAL INFORMATION

The GENERATE command has been suspended, as a part of it uses the READ command and since we have altered the operation of READ, using it will create a problem. I have not modified the program to enable GENERATE as I did not think it necessary for utilities handling. For those among you who wish it, though, try the following:
(i) Use one of the non-functional commands (D,F,P or Y) as the command letter for the READ operation. Set the pointer associated with the new letter to the start of the read command in ROM (changing NAS-SYS command table and STAB).
(ii) Then modify the string output used by GENERATE (bring GENERATE into RAM and then modify) to use your new routine letter (say F) for the new READ, then change the string (original address, 064C hex onwards) to

```
GDS     DFEB    CR,"E,"Ø,CR,"F,CR
GDSE    EQU  E
```

(iii) Remember that ARGX will have F instead of R so alter accordingly, otherwise the best you will get is a verify.

Although I have not made this modification outlined, I see no reason why it should not work.

USAGE

Since BASIC has its own file handling routine, using the new routines would only make things hang up. But using them in ZEAP is fine: in fact, since you have 16 letters to use, it is easy to store a program name and indicate whether it is source or object code.

For those with an assembler, it is possible to relocate the program by altering the ORG statement to a high area in RAM. This will allow it to be used as a utilities handler. Have it placed on a tape, as the first file, by the GENERATE command (since it has not yet been loaded, GENERATE has not been removed). Then follow it with all your utilities in the headed format.

The program is moved to high RAM because many utilities programs use the space that it occupied (0C80-0FC0 hex) when being used as a simple addition to the monitor.

Care should be taken if you save this program under its own operation. If you save the workspace, phantom headers may appear because of the way the program works. This is because the file will have its own header plus a valid header in the workspace.

HOW IT WORKS

Data is in 256 byte blocks, although the last block may have

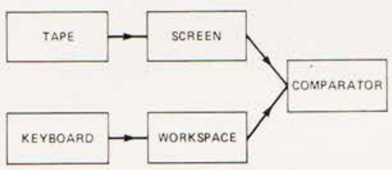


Fig. 3 Method used to compare files

less than that. The format of each block is shown in Fig. 1. All programs are output in a series of blocks, format as above, but the file structure of a headed program is shown in Fig. 2.

On loading the program in from tape type E 0D0C — this will alter the command table and set the new routines in NAS-SYS: if all is well it will return with the NAS-SYS-1 prompt. The programs themselves are more of a prefix to

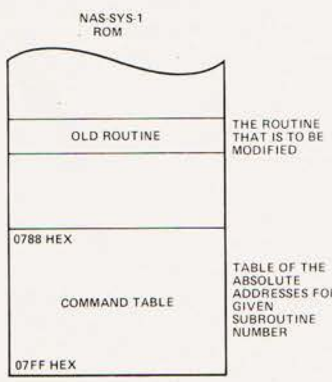


Fig. 4 The memory map before modificaiton

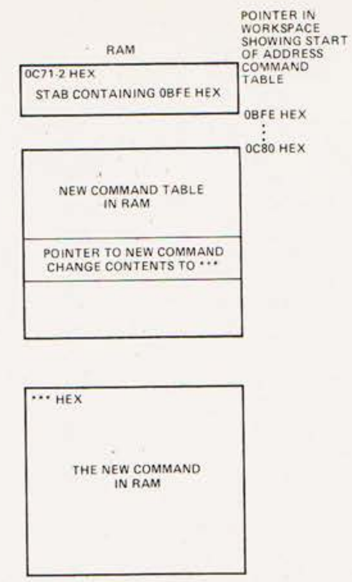


Fig. 5 The new memory map

the normal commands as opposed to a modification of them. Between the comments on the listing and the flowcharts all should be self-explanatory, but I have mentioned any points of interest below.

WRITE The SCAL 05D (one second delay) is to allow the tape drive to get up to speed. Since this has been done in the new addition to the program, we can re-enter the old routine past its own delay.

VERIFY and READ The comparison of required file to encountered files has been done as shown in Fig. 3, since it is easier to compare block for block than as it comes off the tape.

From the flowcharts it is not clear how NAS-SYS knows which is READ and which is VERIFY. It is done by loading the last command letter into ARGX before the routine is called, and then testing if ARGX contains the letter R.

NAS-SYS-1 treats its subroutines and commands much alike: for instance, if you enter values in the HL and DE registers and then SCAL 041 hex (DF 41) in a program, this is the same as using the arithmetic routine

A xxxx yyyy

People who use the Z80 CPU will recognise the instruction DF as RST 018 hex. The next byte in the program is the number of a particular command or subroutine within NAS-SYS (41 hex is ASCII A which means Arithmetic). To get the absolute address of the start of the routine, NAS-SYS uses a subroutine table.

So if we alter the values of the address in this table we can write our own routines. The only problem is that this table sits deep in the NAS-SYS-1 ROM (0788 to 07FF) so it cannot be altered.

The answer to our problem is in the fact that NAS-SYS holds a pointer to the start of the table in workspace RAM (0C71-0C72) called STAB. So if we alter STAB to point to the start of our new subroutine/command table in RAM then NAS-SYS will use this table instead of its ROM-based one.

There is one further step: since the table starts with the commands, the commands start with Arithmetic, and the table is ASCII-associated it will start at (STAB) + 2 *041 hex bytes.

The displacement of 82 hex bytes is to allow for the 41 hex address not being used.

To save space the table starts at 0C80 hex so the pointer is at 082 hex less than this (ie 0BFE). Since no access to the table is made below 'A' ASCII there is no risk of a false address being created by taking a value from NAS-SYS workspace.

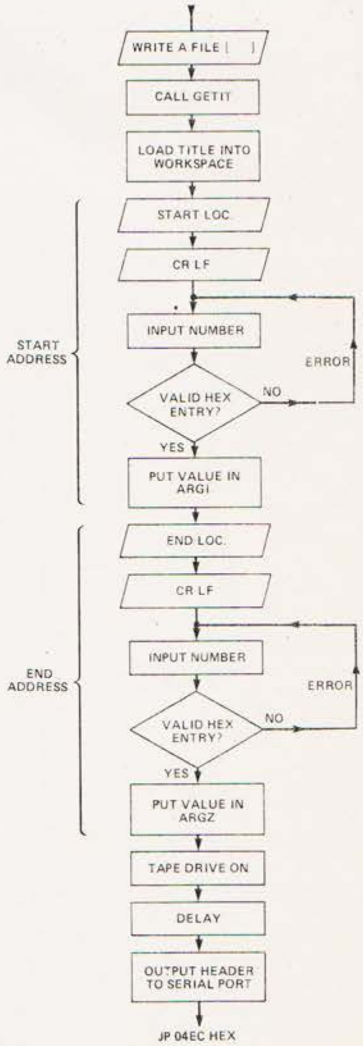


Fig. 6 The WRITE command flowchart

So now we have a copy of the command table in RAM, and the pointer is set to the start of it. Thus any alteration of the absolute address in the table is all that is needed to modify NAS-SYS-1 commands or subroutines, or even add commands to the ones available. I say add as the monitor commands D,F,P and Y do not exist, their table addresses pointing to ERRM (Error message). By altering these addresses the commands may be used to suit your own needs, for example debug routines and so on.

The only restraint that is placed upon the new subroutines is that they must obey the same functions and preserve the same registers as the routine that was replaced (since the monitor uses the same subroutines for its own operation).

Remember that on reset NAS-SYS re-initialises part of its workspace, STAB included. So any

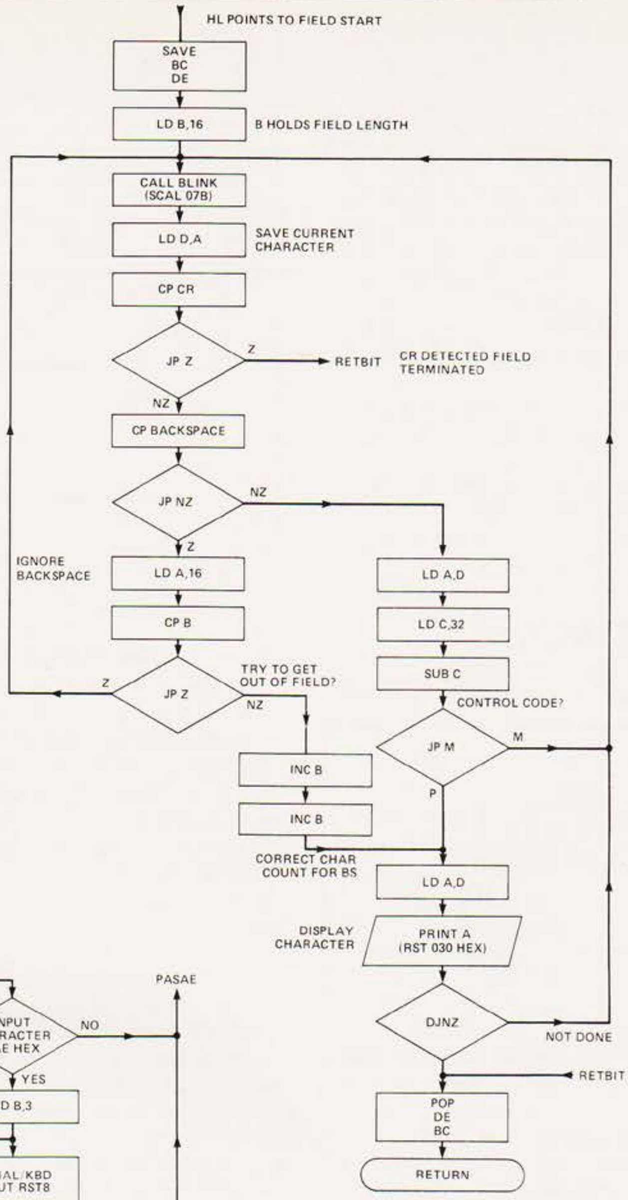


Fig. 8 The GETIT flowchart

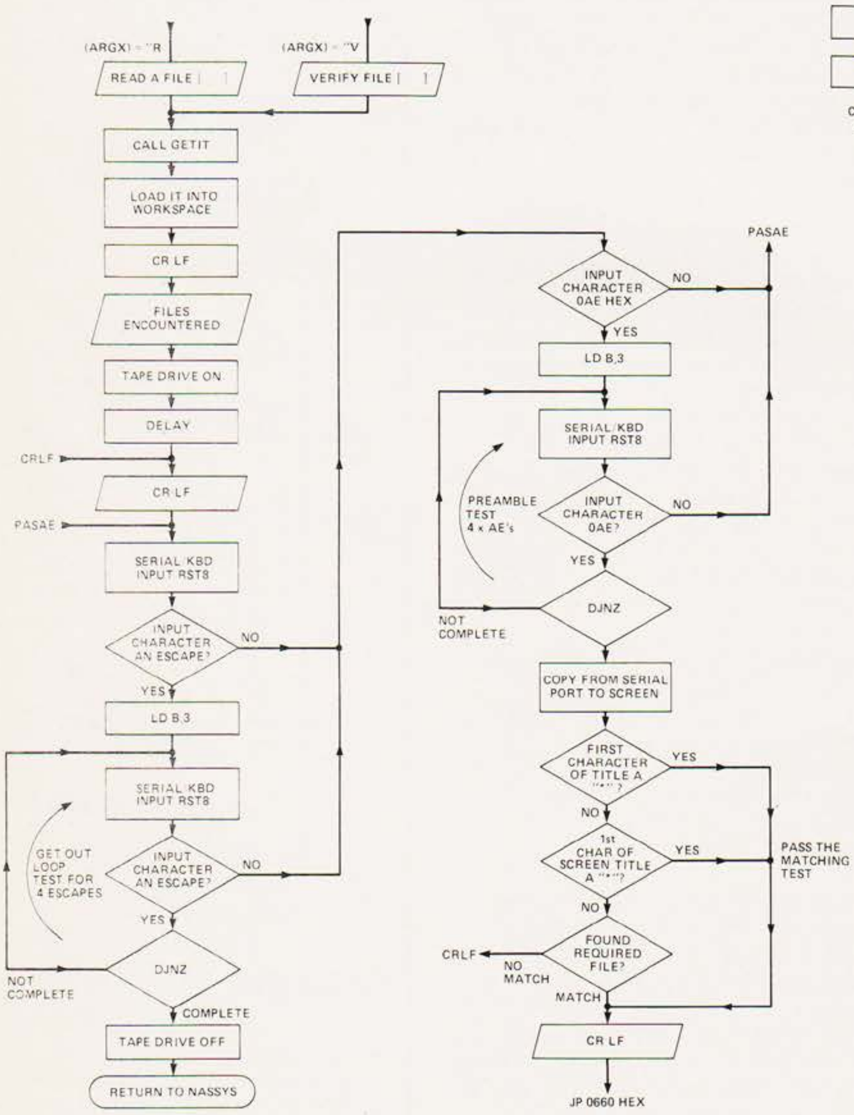


Fig. 7 The flowchart for READ and VERIFY

modification will be made ineffective until you run the program to change STAB again.

In general any modification to NAS-SYS will only be in a part of a subroutine or command, so if possible jump to the old routine at the relevant place after the modification. This will save you time and space by avoiding duplication.

Some modifications will not appear to work when in NAS-SYS (changing INLIN). This is because the monitor program does not call them from the table but runs into them from another routine.

When debugging a new modification remember that the B command will only work on a program under execution, and that your routine is a part of the monitor. So you will have to carefully set your breakpoint and then execute your routine.

```
               0001 ;*******************************************
               0002 ;MODIFIED W,R AND V COMMANDS (HEADED FILE)
               0003 ;*******************************************
0C80           0010 START  ORG  0C80H
0C80 0BFE      0020 TSTART EQU  START-"A-"A
0C80 0C29      0030 CURSOR EQU  0C29H
0C80 0C0C      0040 ARG1   EQU  0C0CH
0C80 0C0E      0050 ARG2   EQU  0C0EH
0C80 0C21      0060 NUMV   EQU  0C21H
0C80 0008      0070 BACSPC EQU  08H
0C80 000D      0080 CR     EQU  0DH
0C80 001B      0090 ESCAPE EQU  01BH
0C80 0C2B      0100 ARGX   EQU  0C2BH

               0120 ;Workspace
0C80 0078      0130 TABLE  DEFS 078H
0CF8 AE        0140 HEADER DEFB 0AEH
0CF9 AE        0150        DEFB 0AEH
0CFA AE        0160        DEFB 0AEH
0CFB AE        0170        DEFB 0AEH
0CFC 0010      0180 TITLE  DEFS 16

               0200 ;Move and alter command table
0D0C           0210        ENT
0D0C 21F80C    0220        LD   HL,HEADER
0D0F 0604      0230        LD   B,4
0D11 36AE      0240 HLP1   LD   (HL),0AEH
0D13 23        0250        INC  HL
0D14 10FB      0260        DJNZ HLP1 ;enshure tape header in wks.
0D16 218807    0270        LD   HL,0788H
0D19 11800C    0280        LD   DE,START
0D1C 017800    0290        LD   BC,078H
0D1F EDB0      0300        LDIR ;Table to ram
0D21 21710C    0310        LD   HL,0C71H
0D24 11FE0B    0320        LD   DE,TSTART
0D27 73        0330        LD   (HL),E
0D28 23        0340        INC  HL
0D29 72        0350        LD   (HL),D ;new $STAB

               0380 ;Load new V,R,W adds. in table
0D2A 21CF0D    0390        LD   HL,READ
0D2D 22A20C    0400        LD   (TSTART+"R+"R),HL
0D30 216D0E    0410        LD   HL,VERIFY
0D33 22AA0C    0420        LD   (TSTART+"V+"V),HL
0D36 21470D    0430        LD   HL,WRITE
0D39 22AC0C    0440        LD   (TSTART+"W+"W),HL
0D3C 21900E    0450        LD   HL,GENRAT
0D3F 228C0C    0460        LD   (TSTART+"G+"G),HL
0D42 3E0C      0470        LD   A,0CH
0D44 F7        0480        RST  030H ;clear screen
0D45 DF5B      0490        SCAL 05BH

0D47 EF        0510 WRITE  RST  028H
0D48 57726974  0520        DEFM /Write a file [                 ]/
     65206120
     66696C65
     205B2020
     20202020
     20202020
     20202020
     20205D
0D67 00        0530        DEFB 0
0D68 2A290C    0540        LD   HL,(CURSOR)
0D6B 11EFFF    0550        LD   DE,-17
0D6E 19        0560        ADD  HL,DE
0D6F 22290C    0570        LD   (CURSOR),HL
0D72 E5        0580        PUSH HL
0D73 CDBB0E    0590        CALL GETIT
0D76 E1        0600        POP  HL
0D77 11FC0C    0610        LD   DE,TITLE
0D7A 011000    0620        LD   BC,16
0D7D EDB0      0630        LDIR
0D7F DF6A      0640 REFROM SCAL 06AH
0D81 EF        0650        RST  028H
0D82 53746172  0660        DEFM /Start loc./
     74206C6F
     632E
0D8C 00        0670        DEFB 0
0D8D DF6A      0680        SCAL 06AH ;crlf
0D8F DF63      0690        SCAL 063H ;inlin
0D91 DF64      0700        SCAL 064H ;num
0D93 D29B0D    0710        JP   NC,PASERR
0D96 DF6B      0720        SCAL 06BH ;errm
0D98 C37F0D    0730        JP   REFROM
0D9B 2A210C    0740 PASERR LD   HL,(NUMV)
0D9E 220C0C    0750        LD   (ARG1),HL ;start value in ARG1
0DA1 DF6A      0760 RETO   SCAL 06AH
0DA3 EF        0770        RST  028H
0DA4 456E6420  0780        DEFM /End loc./
     6C6F632E
0DAC 00        0790        DEFB 0
0DAD DF6A      0800        SCAL 06AH
0DAF DF63      0810        SCAL 063H
0DB1 DF64      0820        SCAL 064H
0DB3 D2BB0D    0830        JP   NC,PASER1
0DB6 DF6B      0840        SCAL 06BH
0DB8 C3A10D    0850        JP   RETO
0DBB 2A210C    0860 PASER1 LD   HL,(NUMV)
0DBE 220E0C    0870        LD   (ARG2),HL
               0880 ;All data now ready
0DC1 DF5F      0890        SCAL 05FH ;tape drive on
0DC3 DF5D      0900        SCAL 05DH ;1sec.delay
0DC5 21F80C    0910        LD   HL,HEADER
0DC8 0614      0920        LD   B,20
0DCA DF6D      0930        SCAL 06DH ;Output header
0DCC C3EC04    0940        JP   04ECH ;Jump into W command

0DCF EF        0960 READ   RST  028H
0DD0 52656164  0970        DEFM /Read a file [                  ]/
     20612066
     696C6520
     5B202020
     20202020
     20202020
     20202020
     205D
0DEE 00        0980        DEFB 0
0DEF 2A290C    0990 GETIN  LD   HL,(CURSOR) ;Entry for V command
0DF2 11EFFF    1000        LD   DE,-17
0DF5 19        1010        ADD  HL,DE
0DF6 22290C    1020        LD   (CURSOR),HL
0DF9 E5        1030        PUSH HL
0DFA CDBB0E    1040        CALL GETIT
0DFD E1        1050        POP  HL
0DFE 11FC0C    1060        LD   DE,TITLE
0E01 011000    1070        LD   BC,16
0E04 EDB0      1080        LDIR
0E06 DF6A      1090        SCAL 06AH
0E08 EF        1100        RST  028H
0E09 46696C65  1110        DEFM /Files encountered /
     7320656E
     636F756E
     74657265
     6420
0E1B 00        1120        DEFB 0
0E1C DF5F      1130        SCAL 05FH
0E1E DF6A      1140 CRLF   SCAL 06AH
0E20 CF        1150 PASAE  RST  8
0E21 FE1B      1160        CP   ESCAPE
0E23 C2340E    1170        JP   NZ,PASESC
0E26 0603      1180        LD   B,3
0E28 CF        1190 ELP1   RST  8
0E29 FE1B      1200        CP   ESCAPE
0E2B C2340E    1210        JP   NZ,PASESC
0E2E 10F8      1220        DJNZ ELP1
0E30 DF5F      1230        SCAL 05FH
0E32 DF5B      1240        SCAL 05BH
0E34 FEAE      1250 PASESC CP   0AEH
0E36 C2200E    1260        JP   NZ,PASAE
0E39 0603      1270        LD   B,3
0E3B CF        1280 AELPT  RST  8
0E3C FEAE      1290        CP   0AEH
0E3E C2200E    1300        JP   NZ,PASAE
0E41 10F8      1310        DJNZ AELPT
0E43 2A290C    1320        LD   HL,(CURSOR)
0E46 E5        1330        PUSH HL
0E47 0610      1340        LD   B,16
0E49 CF        1350 FILELP RST  8
0E4A F7        1360        RST  030H
0E4B 10FC      1370        DJNZ FILELP
0E4D E1        1380        POP  HL
0E4E 11FC0C    1390        LD   DE,TITLE
0E51 0610      1400        LD   B,16
0E53 1A        1410        LD   A,(DE)
0E54 FE2A      1420        CP   "*
0E56 CA680E    1430        JP   Z,UNIV
0E59 7E        1440        LD   A,(HL)
0E5A FE2A      1450        CP   "*
0E5C CA680E    1460        JP   Z,UNIV
0E5F 1A        1470 TESTLE LD   A,(DE)
0E60 BE        1480        CP   (HL)
0E61 C21E0E    1490        JP   NZ,CRLF
0E64 23        1500        INC  HL
0E65 13        1510        INC  DE
0E66 10F7      1520        DJNZ TESTLE
0E68 DF6A      1530 UNIV   SCAL 06AH
0E6A C36006    1540        JP   0660H

0E6D EF        1560 VERIFY RST  028H
0E6E 56657269  1570        DEFM /Verify file [                  ]/
     66792066
     696C6520
     5B202020
     20202020
     20202020
     20202020
     205D
0E8C 00        1580        DEFB 0
0E8D C3EF0D    1590        JP   GETIN

0E90 DF6A      1610 GENRAT SCAL 06AH
0E92 EF        1620        RST  028H
0E93 47454E45  1630        DEFM /GENERATE not currently available./
     52415445
     206E6F74
     20637572
     72656E74
     6C792061
     7661696C
     61626C65
     2E
0EB4 00        1640        DEFB 0
0EB5 DF6A      1650        SCAL 06AH
0EB7 DF5B      1660        SCAL 05BH
0EB9 DF5B      1670        SCAL 05BH
               1680 ;Get file name subroutine.
0EBB C5        1690 GETIT  PUSH BC
0EBC D5        1700        PUSH DE
0EBD 0610      1710        LD   B,16
0EBF DF7B      1720 GETLP1 SCAL 07BH
0EC1 57        1730        LD   D,A
0EC2 FE0D      1740        CP   CR
0EC4 CAE20E    1750        JP   Z,RETBIT
0EC7 FE08      1760        CP   BACSPC
0EC9 C2D70E    1770        JP   NZ,PASBAC
0ECC 3E10      1780        LD   A,16
0ECE B8        1790        CP   B
0ECF CABF0E    1800        JP   Z,GETLP1
0ED2 04        1810        INC  B
0ED3 04        1820        INC  B
0ED4 C3DE0E    1830        JP   DISPLY
0ED7 7A        1840 PASBAC LD   A,D
0ED8 0E20      1850        LD   C,32
0EDA 91        1860        SUB  C
0EDB FABF0E    1870        JP   M,GETLP1
0EDE 7A        1880 DISPLY LD   A,D
0EDF F7        1890        RST  030H
0EE0 10DD      1900        DJNZ GETLP1
0EE2 D1        1910 RETBIT POP  DE
0EE3 C1        1920        POP  BC
0EE4 C9        1930        RET

ZEAP Z80 Assembler - Symbol Table

0E3BH 1280 AELPT        0C0CH 0040 ARG1
0C0EH 0050 ARG2         0C2BH 0100 ARGX
0008H 0070 BACSPC       000DH 0080 CR
0E1EH 1140 CRLF         0C29H 0030 CURSOR
0EDEH 1880 DISPLY       0E28H 1190 ELP1
001BH 0090 ESCAPE       0E49H 1350 FILELP
0E90H 1610 GENRAT       0DEFH 0990 GETIN
0EBBH 1690 GETIT        0EBFH 1720 GETLP1
0CF8H 0140 HEADER       0D11H 0240 HLP1
0C21H 0060 NUMV         0E20H 1150 PASAE
0ED7H 1840 PASBAC       0DBBH 0860 PASER1
0D9BH 0740 PASERR       0E34H 1250 PASESC
0DCFH 0960 READ         0D7FH 0640 REFROM
0EE2H 1910 RETBIT       0DA1H 0760 RETO
0C80H 0010 START        0C80H 0130 TABLE
0E5FH 1470 TESTLE       0CFCH 0180 TITLE
0BFEH 0020 TSTART       0E68H 1530 UNIV
0E6DH 1560 VERIFY       0D47H 0510 WRITE
```

Calc Result

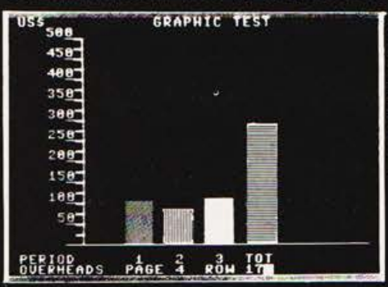
Three Dimensional Spreadsheet that turns your Commodore 64 into a powerful business computer

Only CalcResult gives you all these exciting features . . .

● THREE DIMENSIONAL CALCULATION

CalcResult's powerful three dimensional spread sheet format rapidly becomes an irreplaceable tool for those involved in figure manipulation. Budgets reports, statements, forecasts and planning are all areas in which a saving in time and resources are as important as the ability to handle all forms of calculations. The many features of CalcResult simplify the rapid production and calculation of all aspects of business forecasting and planning. Each model can have up to 32 pages of associated information with a consolidated report and every page has 63 × 254 data positions which can contain text, formulas or values. CalcResult's powerful editing functions mean rapid creation and amendment of models and, once created, split screens and multiple page viewing allow easy review of information.

● GRAPHIC REPRESENTATION



CalcResult can translate rows of columns of numbers into histograms on the screen or printed histograms can be used to enhance presentation of tabular information in reports. (With CalcResult 64 full colour options are available on the screen.)

● HELP FUNCTION AND AN EXPLICIT MANUAL

CalcResult is simple to use and easy to learn thanks to a detailed instruction manual and help screens that are available at the touch of a button, to explain the functions available for each command.

● FLEXIBLE PRINTING

CalcResult allows you to define your own report layout. You can choose the order of columns to be printed and set the width of each independently. Layout definitions can be stored on a disk for future use.

● TIME SAVING COMMANDS AND CONDITIONAL FUNCTIONS

Commands like replicate, copy, move, go to, insert and delete make CalcResult simple and easy to use. CalcResult calculates according to correct mathematical priority and has standard functions for statistics, trigonometry and logarithms. Conditional functions like IF THEN ELSE and OR AND NOT allow greater flexibility in the setting up of models. Once created, all formulas are protected from accidental deletion.

CalcResult will only operate in conjunction with Commodore 64, a Commodore 1541 disk drive, and a Commodore serial printer.

PRICE: CalcResult 64 £109.00 (plus VAT)

CalcResult is distributed solely in the UK by Kobra Micro Marketing. For details of your nearest stockist of CalcResult phone (0491) 572512 or fill in the coupon below.

KOBRA
MICRO MARKETING
P.O. Box 28,
Henley-on-Thames, Oxon.
Telephone
(0491) 572512

To: Kobra Micro Marketing, P.O. Box 28, Henley-on-Thames, Oxon.
Please send details of the exciting CalcResult program

Name: ______________________

Position: ______________________

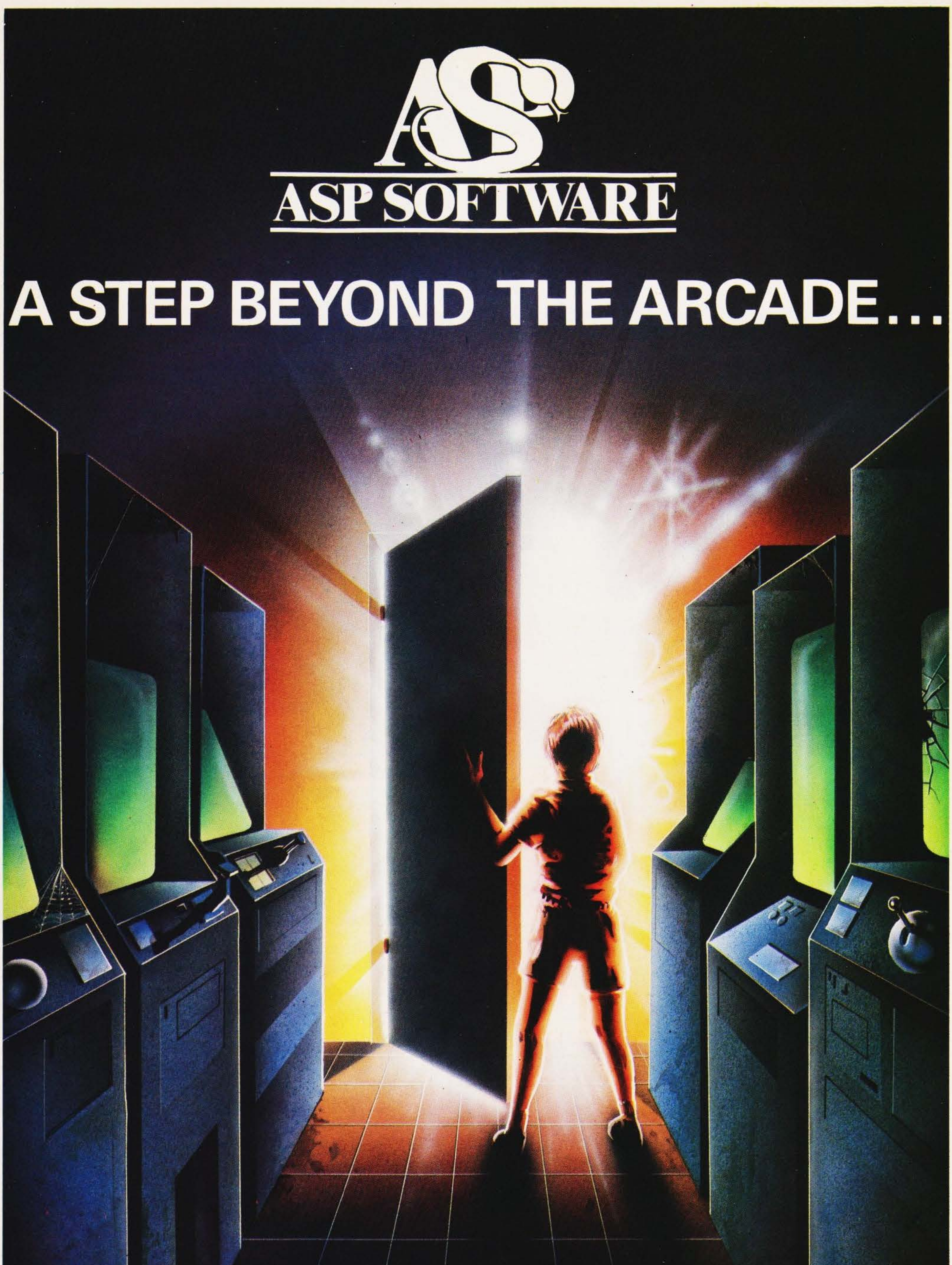
Address: ______________________

____

______________________ Tel. No. ______________________

handic
software ab

-a company in the Datatronic group-


ASP SOFTWARE
A STEP BEYOND THE ARCADE...
Demon Knight · The Valley · Strategy 1–Invasion · Strategy 2–Bismark
White Barrows · Detective · Cells & Serpents · Stockmarket
Conquering Everest · Cloneit/Renumber · Gallery of Death · Planetfall
HORNSBY

Don Thomasson

PROBLEM PAGE

Our sixth and final article in this series answers our poser from last month about the magic square and shows how squares of larger size may be tackled.

The program offered in this final part of the series will allow 4-by-4-sized magic squares to be worked out, and can easily be modified for the 5-by-5 variety. Before looking at it in detail, however, it will be useful to consider the relatively trivial 3-by-3 case, where the numbers 1 to 9 have to be arranged so that each line, column and major diagonal adds up to 15.

This total can be made up in just eight ways, all of which will be used in the diagram:

1+5+9
1+6+8
2+4+9
2+5+8
2+6+7
3+4+8
3+5+7
4+5+6

The number in the centre is part of a column, a row, and two diagonals. It must therefore appear in four combinations, and must therefore be 5. Numbers in the corners are part of a row, a column and one diagonal. They must appear three times, and must therefore be 2, 4, 6 and 8. On this basis, a possible square is;

2 9 4
7 5 3
6 1 8

All other possible arrangements are rotations or reflections of this.

Determination of viable arrangements for larger squares is more complicated, and there are many more possibilities, but the general principle is that the viable combinations for each line are worked out, and the possible values for a given square can then be found by collating the values for the lines which pass through the square. The program uses five arrays:

A contains the given data, fed in by the user.

B starts as a copy of A, but adds deduced entries.

C identifies unused numbers by zeroes.

D holds the viable numbers for lines.

E holds viable numbers for squares.

Subroutine 1000 clears A and B, and subroutine 4000 displays an empty grid, from the contents of array B. Subroutine 1200 then allows the input of given data, a zero number erasing an existing entry. A zero row number gives an exit to the next stage, which copies the data into array B and sets up the used numbers in array C (subroutine 1300). The display of array B contents is refreshed.

An opportunity is then given to return to subroutine 1200 to make any changes. Otherwise, subroutine 3500 clears arrays D and E, and subroutine 2000 analyses the situation. This takes some time in the early stages, and those who are impatient may like to insert a line 2725 to print out the viable combinations of A, B, C and D. TOTAL may be printed out as well, and the addition of L will show which line is being handled: the rows are numbered 1 to 4, the columns 5 to 8, and the diagonals are 9 and 10. The amount of work being done may come as a surprise.

When subroutine 2000 has set up array D, subroutine 1400 collates the lines through each vacant square and puts the results in array E. The variable MIN registers the case offering the minimum number of alternatives, and if MIN=1 the only possible number is set in array B and displayed. If MIN=0, there is no solution, and that is reported. If MIN is more than one, there are options in all squares, and the case with minimum alternatives is reported.

The fact that a particular option is reported is no guarantee it will yield a viable solution. The numbers offered should be noted and tried one at a time, the routine returning to subroutine 1200 for that purpose if the answer to 'Any changes?' is Y or y. If no solution is then found, the next option should be tried.

Note that any numbers deduced earlier in the process are lost, since they may not be viable with the option in use.

If MIN=16, the square is complete, and the routine stops.

The program was originally written for a 5-by-5 matrix, and can be reconverted by altering the constants as follows: 4 to 5; 5 to 6; 9 to 11; 10 to 12 (not in lines 1250 and 4020); 8 to 15; 11 to 21; 15 to 24; 16 to 25; 34 to 65. For this adaptation, you may like to try the two squares in Fig. 1.

0	6	5	0	0
0	0	0	16	10
11	20	0	0	0
9	0	0	15	18
0	13	19	7	0

9	0	0	0	11
0	21	0	12	0
0	0	13	0	4
16	0	7	0	0
0	8	1	24	0

Fig. 1. Two 5-by-5 magic squares to solve using the modified program.

Listing 1. Program to solve 4-by-4 magic squares such as the one given last month.

```
100 DIM A(4,4)
110 DIM B(4,4)
120 DIM C(16)
130 DIM D(16,10)
140 DIM E(16,4,4)
150 GO SUB 1000
160 GO SUB 4000
170 GO SUB 1200
180 CLS
190 GO SUB 1300
200 GO SUB 4000
210 INPUT "Any changes ? ";N$
220 IF N$="Y" OR N$="y" THEN GO TO 170
230 GO SUB 3500
240 GO SUB 2000
250 GO SUB 1400
260 IF MIN=0 THEN PRINT "No solution": GO TO 210
270 IF MIN=16 THEN STOP
280 IF MIN=1 THEN GO TO 230
290 PRINT "There are options at ";HX;",";HY
300 PRINT "They are: ";
310 FOR Z=1 TO 16
320 IF E(Z,HX,HY)<>0 THEN PRINT Z;"[SPC]";
330 NEXT Z
340 GO TO 210

1000 FOR X=1 TO 4
1010 FOR Y=1 TO 4
1020 LET A(X,Y)=0
1030 LET B(X,Y)=0
1040 NEXT Y
1050 NEXT X
1060 RETURN

1200 INPUT "Row ? ";ROW
1210 IF ROW=0 THEN RETURN
1220 INPUT "Column ? ";COL
1230 INPUT "Number ? ";NUM
1240 LET A(ROW,COL)=NUM
1250 PRINT AT 2*ROW,3*COL+(NUM<10)+10;NUM
1260 GO TO 1200

1300 FOR X=1 TO 16
1310 LET C(X)=0
1320 NEXT X
1330 FOR X=1 TO 4
1340 FOR Y=1 TO 4
1350 LET B(X,Y)=A(X,Y)
1360 IF A(X,Y)<>0 THEN LET C(A(X,Y))=1
1370 NEXT Y: NEXT X
1380 RETURN

1400 LET MIN=16
1410 FOR X=1 TO 4
1420 FOR Y=1 TO 4
1430 IF MIN=0 OR B(X,Y)<>0 THEN GO TO 1550
1440 LET COUNT=0
1450 FOR Z=1 TO 16
1460 LET F=1
1470 IF X=Y THEN LET F=D(Z,9)
1480 IF X+Y=5 THEN LET F=D(Z,10)
1490 LET G=F AND D(Z,X) AND D(Z,Y+4)
1500 LET E(Z,X,Y)=G
1510 IF G<>0 THEN LET COUNT=COUNT+1: LET HZ=Z
1520 NEXT Z
1530 IF COUNT<MIN THEN LET MIN=COUNT: LET HX=X: LET HY=Y
1540 IF COUNT=1 THEN LET B(X,Y)=HZ: LET C(HZ)=1: GO SUB 4000
1550 NEXT Y
1560 NEXT X
1570 RETURN

2000 FOR K=1 TO 4
2010 LET L=K: LET COUNT=0: LET TOTAL=0
2020 FOR N=1 TO 4
2030 IF B(K,N)=0 THEN LET COUNT=COUNT+1
2040 LET TOTAL=TOTAL+B(K,N)
2050 NEXT N
2060 GO SUB 2300
2070 LET L=K+4: LET COUNT=0: LET TOTAL=0
2080 FOR N=1 TO 4
2090 IF B(N,K)=0 THEN LET COUNT=COUNT+1
2100 LET TOTAL=TOTAL+B(N,K)
2110 NEXT N
2120 GO SUB 2300
2130 NEXT K
2140 LET L=9: LET COUNT=0: LET TOTAL=0
2150 FOR N=1 TO 4
2160 IF B(N,N)=0 THEN LET COUNT=COUNT+1
2170 LET TOTAL=TOTAL+B(N,N)
2180 NEXT N
2190 GO SUB 2300
2200 LET L=10: LET COUNT=0: LET TOTAL=0
2210 FOR N=1 TO 4
2220 IF B(N,5-N)=0 THEN LET COUNT=COUNT+1
2230 LET TOTAL=TOTAL+B(N,5-N)
2240 NEXT N
2250 GO SUB 2300
2260 RETURN

2300 IF COUNT=0 THEN RETURN
2310 IF COUNT=1 THEN LET A=0: LET B=0: LET C=0: GO TO 2700
2320 IF COUNT=2 THEN LET A=0: LET B=0: GO TO 2600
2330 IF COUNT=3 THEN LET A=0: GO TO 2500
2340 IF COUNT=4 THEN GO TO 2400
2350 FOR N=1 TO 16
2360 IF C(N)=0 THEN LET D(N,L)=1
2370 NEXT N
2380 RETURN

2400 FOR A=1 TO 8
2410 IF C(A)<>0 THEN GO TO 2430
2420 GO SUB 2500
2430 NEXT A
2440 RETURN

2500 FOR B=A+1 TO 11
2510 IF C(B)<>0 THEN GO TO 2530
2520 GO SUB 2600
2530 NEXT B
2540 RETURN

2600 FOR C=B+1 TO 15
2610 IF C(C)<>0 THEN GO TO 2630
2620 GO SUB 2700
2630 NEXT C
2640 RETURN

2700 LET D=34-TOTAL-A-B-C
2710 IF D>16 OR D<=C THEN RETURN
2720 IF C(D)<>0 THEN RETURN
2730 IF A<>0 THEN LET D(A,L)=1
2740 IF B<>0 THEN LET D(B,L)=1
2750 IF C<>0 THEN LET D(C,L)=1
2760 IF D<>0 THEN LET D(D,L)=1
2770 RETURN

3500 FOR X=1 TO 16
3510 FOR Y=1 TO 10
3520 LET D(X,Y)=0
3530 NEXT Y: NEXT X
3540 FOR X=1 TO 4
3550 FOR Y=1 TO 4
3560 FOR Z=1 TO 16
3570 LET E(Z,X,Y)=0
3580 NEXT Z: NEXT Y: NEXT X
3590 RETURN

4000 FOR P=1 TO 4
4010 FOR Q=1 TO 4
4020 PRINT AT 2*Q,3*P+10+(B(Q,P)<10);B(Q,P)
4030 NEXT Q: NEXT P
4040 RETURN
```

H.M. Hoffman

ATARI DISC LIBRARY

If you've got an Atari, a disc drive, and lots of files on lots of discs, here's a utility program that will help you keep track of what's what and where.

VARIABLE	FUNCTION
NAME$	Disc number
DES$	Comments
A$	Filename
C$	Record read from disc
I$	User response
RECORD$	The whole data base
FILE$	Data for one disc
CLS	Clear Screen character (125)
LD	Length of comment
NAME	Number of files read
PT	Pointer into FILE$
ER	Error number
GC	ASCII code of key pressed

Table 1. Use of variables in the program.

This program has been written for Atari owners who possess one (or more) disc drives. The program produces a cumulative catalogue of your disc files, complete with comments for each name to help you keep track of many files spread over several discs.

The menu options in the program are as follows:

1. Create library file (should it not exist already). A suitable warning is displayed should you attempt to make use of this option with an existing data file.

2. Read files. This option automatically reads the files off the disc currently in the drive. There is an opportunity to enter a two-character disc number, then a 20-character comment for each file. There is a command to enable any file to be omitted from the database. Use is made of the multi-size and colour text, mixed modes.

When all the files are read from any disc, the data to be stored is displayed and the user is prompted to insert the data (library) disc, whereupon the new data is added to the database. This can be repeated for the other discs in your collection.

3 and 4. These options are self-explanatory, but 3 must be used before 4, although 4 can be repeated without having to re-load from disc. I have formatted the output to be suitable for 40 or 80 column printers, or just the screen. Use Control-1 to pause the display.

5 and 6. The last two options require no explanation.

The program uses different colour text for different options and on an unexpected error the error number is displayed without breaking out of the program. Some points of interest are that location 710 holds the screen colour, while location 195 holds the error number. Note also that by entering PROGRAMS,E: the library can be viewed under DOS copy 'C'.

```
1A PROGRAMS        DATAFILE
1A DOS      SYS    COULD HAVE SKIPPED
1A DUP      SYS    AND THIS
1A AUTORUN  SYS    RUNS"D:LIBRARY.BAS"
1A LIBRARY  BAS    THE LIBRARY PROGRAM
2A UTL1            CALENDER
2A UTL2            HEX DECIMAL CONVERT
2A UTL3            CREATE AUTORUN RUN
2A UTL4            CLOCK (60Hz)!!!
2A GAM2            MAZE CHASE
2A GAM3            DETECTIVE
2A GAM4            SUPER BARRICADE
2A DEM1            HELICOPTER DEMO
```

Fig. 1. Sample database printout.

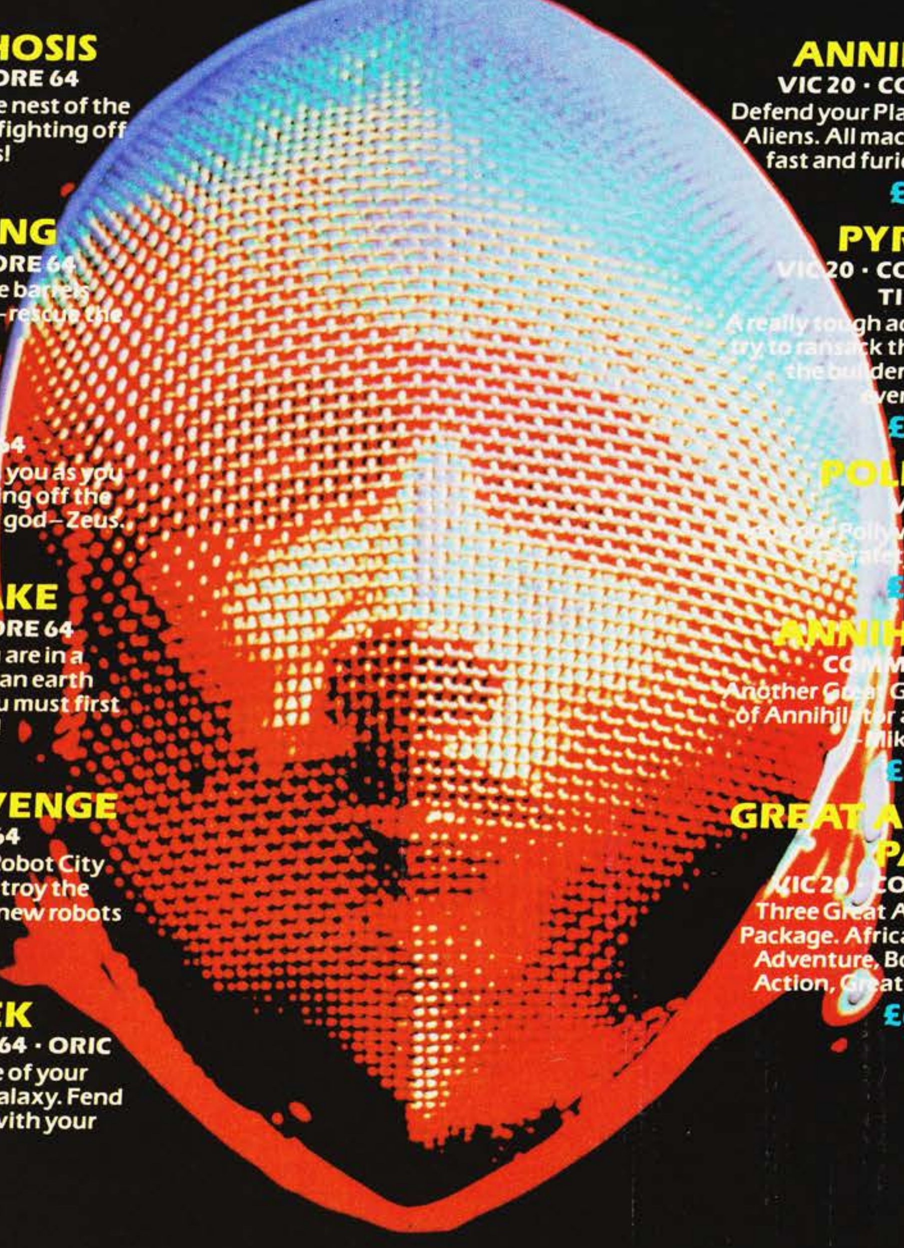
Listing 1. Complete listing for the program Disc Library.

```
2 REM ATARI 400/800 DISK LIBRARY PROGRAM
3 REM 32K MINIMUM MEMORY
4 REM BY H.M.HOFFMAN
9 DIM NAME$(2),FILE$(36*50),DES$(21),A$(20),I$(1),C$(37)
10 DIM RECORD$(500*35):CLS=125:GRAPHICS 0
11 POKE 710,180:POKE 195,0
13 GOTO 380
14 ? CHR$(125):? "INSERT DISK TO BE CATALOGUED":?
15 ? "ENTER DISK NUMBER (2 CHARS MAX)":INPUT NAME$
26 FILE$=" ":FILE$(36*50)=" ":FILE$(2)=FILE$
28 NAME=0
30 CLOSE #1:OPEN #1,6,0,"D:*.*":REM DIRECTORY OPEN
32 PT=0
34 INPUT #1,A$
35 IF A$(5,13)="FREE SECT" THEN 150:REM TEST FOR END OF DIRECTORY
36 GRAPHICS 2:POKE 710,0:POSITION 1,1:? #6;"file";NAME+1;"disk#6;"file ";
NAME+1;"   disk   ";NAME$;
37 POSITION 1,6:? #6;A$;
38 ? "ENTER COMMENTS or 'NO' to skip"
39 ? " --------------------"
40 INPUT DES$:LD=LEN(DES$)
44 IF DES$="NO" THEN 34
```

```
46 IF LD>20 THEN 36
50 FOR B=LD TO 20:DES$(LEN(DES$)+1)=" ":NEXT B
60 FILE$(1+PT)=NAME$
75 FILE$(4+PT)=A$(3,13)
80 FILE$(17+PT)=DES$
100 PT=PT+36
120 NAME=NAME+1
140 GOTO 34
150 GRAPHICS 0:POKE 710,50
200 FOR Z=1 TO NAME*36 STEP 36
220 ? FILE$(Z,Z+35)
230 NEXT Z
240 ? :? :? "INSERT LIBRARY FILE DISK"
242 ?
245 ? :? "    HIT RETURN "
250 INPUT I$:IF I$<>"" THEN 250
260 OPEN #2,9,0,"D:PROGRAMS":REM APPEND FILE OPEN
270 FOR Z=1 TO NAME*36 STEP 36
280 ? #2;FILE$(Z,Z+36)
290 NEXT Z
300 CLOSE #2
380 ? CHR$(CLS)
390 POKE 710,180
395 ? :? "1-CREATE LIBRARY FILE"
400 ? :? "2-CATALOGUE A DISK"
410 ? :? "3-READ LIBRARY FILE INTO MEMORY"
415 ? :? "4-VIEW AND PRINT LIBRARY"
416 ? :? "5-LOAD DOS UTILITIES"
417 ? :? "6-END":? :?
418 ? :? :ER=PEEK(195):IF ER<>136 AND ER<>0 THEN POSITION 2,20:?"ERROR";ER
419 POSITION 2,14:? "SELECTION?":POKE 195,0
420 CLOSE #5:OPEN #5,4,0,"K:":GET #5,GC:NUM=GC-48:REM GET KEY VALUE
440 IF NUM<1 OR NUM>6 THEN 380
450 ON NUM GOTO 900,14,500,600,960,700
500 TRAP 300:REM READ DATA FILE ROUTINE
550 CLOSE #2:OPEN #2,4,0,"D:PROGRAMS":REM READ FILE OPEN
560 NAME=0
570 INPUT #2;C$
580 RECORD$(NAME*36+1)=C$:NAME=NAME+1
585 GOTO 570
600 POKE 710,240:REM PRINT LIBRARY ROUTINE
610 TRAP 380:? CHR$(CLS):FOR Z=1 TO NAME*36 STEP 36
620 ? RECORD$(Z,Z+35)
630 NEXT Z
640 ? "    HARD COPY ? "
650 INPUT I$:IF I$="Y" THEN GOTO 800
670 GOTO 380
700 POKE 710,0:END
800 TRAP 380:? CHR$(125):FOR Z=1 TO NAME*36 STEP 36
810 LPRINT RECORD$(Z,Z+35)
820 NEXT Z
830 GOTO 380
900 ? CHR$(CLS):? "THIS OPTION DELETES ANY EXISTING FILE"
910 ? :? "ARE YOU SURE Y/N"
915 ? :INPUT A$:IF A$<>"Y" THEN 380
920 IF A$="Y" THEN OPEN #2,8,0,"D:PROGRAMS"
930 GOTO 300
960 DOS
```

MOGUL

THE HOTWARE PEOPLE



METAMORPHOSIS
VIC 20 · COMMODORE 64
You have stumbled into the nest of the Cyglorx and find yourself fighting off the Robot Tanks!
£7.95

KONGO KONG
VIC 20 · COMMODORE 64
Climb ladders, avoid the barrels thrown by the Crazy Ape – rescue the Damsel.
£6.95

ZEUS
COMMODORE 64
Your Cone of Cold protects you as you become the Wizard fighting off the Thunderbolts of the angry god – Zeus.
£6.95

EARTHQUAKE
VIC 20 · COMMODORE 64
A great adventure, you are in a shopping centre during an earth quake. To save yourself you must first help the others!
£6.95

CREATORS REVENGE
COMMODORE 64
Penetrate the depths of Robot City and kill the Creator. Destroy the robots – but watch out for new robots as they hatch!
£7.95

SUPER TREK
VIC 20 · COMMODORE 64 · ORIC
Commanding the bridge of your Starship, you explore the Galaxy. Fend off the Klingon attacks with your Phasers.
£6.95

ANNIHILATOR
VIC 20 · COMMODORE 64
Defend your Planet against Invading Aliens. All machine code makes for fast and furious arcade action!
£5.99

PYRAMID
VIC 20 · COMMODORE 64
TI 99/4A
A really tough adventure game as you try to ransack the Pyramid. However the builders have left clues everywhere.
£6.95

POLLYWOG
VIC 20
Pollywog across danger to the safety of his home.
£6.95

ANNIHILATOR 2
COMMODORE 64
Another Great Game from the author of Annihilator and Metamorphosis – Mike Wacker.
£7.95

GREAT ADVENTURE PACK
VIC 20 · COMMODORE 64
Three Great Adventures in One Package. African Escape, Hospital Adventure, Bomb Threat. Great Action, Great Value, Great Fun.
£6.95

PLUS
Four Great Games for the Spectrum 48K
POTTY PLANTER · ADDER ATTACK · LAS VEGAS LADY · CRASH
All at **£5.50**

AVAILABLE FROM ALL GOOD DEALERS
TRADE ENQUIRIES WELCOME

MOGUL COMMUNICATIONS LIMITED · 90 REGENT STREET · LONDON W1R 5PT
Telephone 01-437 3156/7

ADVENTURE COMPETITION

Last month we ran a competition in which we asked you to write an Adventure. This month we want you to take part in an Adventure — one in which you work your way through a series of puzzles and riddles to get to one of the golden artifacts, the key to an amazing prize. The winner of this competition gets something that is definitely rather special — computer hardware to the value of £1000. Not hardware that we've chosen, mind: the type of computer and peripherals you want depend vary much on personal tastes as well as intended applications, so we're letting the winner decide for his or her self what to spend that £1000 on. If you're not too sure of your choice, we'll be happy to discuss your interests with you and recommend a system — and buy it for you.

To enter, you simply have to work your way through the questions below. Each question has a set of possible answers, only one of which is right, and each answer also has a number indicating which is the next question you should attempt. Thus you'll be jumping back and forth through the questions, not necessarily answering them in numerical order and not even answering every one of them.

Eventually you should arrive at one of the golden artifacts strewn throughout the questions. Once you have done so, write on the form below the letters corresponding to the answers you've chosen (for example, you might have CAFBBADCE), add your name and address and post the coupon to Adventure Competition Number 2, Computing Today, 1 Golden Square, London W1 to arrive no later than January 31st. Please write the number of sides on the artifact you found on the back of the envelope to help us do a preliminary sort for the winner.

A helpful hint to those who lose their way — it isn't necessary to visit any question more than once.

RULES OF ENTRY

This competition is open to all UK and Northern Ireland readers of Computing Today except for employees of Argus Specialist Publications Ltd, their printers, distributors or anyone associated with the competition.

All entries must be postmarked before the closing date of January 31st 1984.

No correspondence will be entered into regarding the result of this competition and it is a condition of entry that the judges decision is final.

The winner will be notified by post and the result of the competition will be published in a future issue of Computing Today.

Entries should be clearly marked on the outside of the envelope 'ADVENTURE COMPETITION NO. 2' and be addressed to our new address of 1 Golden Square, London W1.

Start here.

1. You are in a chamber with exits whose doors are marked with the names of beasts with software connections. Your map tells you that the correct door is depicted by a creature that's at home in the very warmest of climates!

A) Elephant	(7)
B) Salamander	(12)
C) Rabbit	(6)
D) Llama	(14)
E) Phoenix	(9)

2. This chamber contains five exits with strange runes (!) carved into the doors. The map says that the correct door is indicated by the answer to a riddle. "I have 40 pins, eight bits and six registers. What am I?"

A) 6809	(8)
B) 1802	(7)
C) 68000	(12)
D) 6502	(13)
E) Z80	(5)

3. This chamber contains a golden pentangle!

4. You find a room with a stack of numbered tablets, each with a key attached. From top to bottom the tablets are numbered thus: 7 11 2 5 4 9. These numbers are repeated, one on each of the six doors. Your map says "DUP 5 ROLL PICK + . " Which door do you unlock and go through?

A) 7	(11)
B) 11	(6)
C) 2	(12)
D) 4	(13)
E) 5	(10)
F) 9	(7)

5. The room you are in has a central column with the number 7248551 carved into it. The base of the column is decorated with the figure 9, while the map merely says "Change to 10 to get the key". What is the key?

A) 3870712	(3)
B) 8144326	(7)
C) 7860158	(14)
D) 2764125	(10)

6. In this room a strange voice says:
"In Greece I've not been heard of late,
But now I've been brought up to date".

A) Prestel	(2)
B) Ceefax	(5)
C) Oracle	(14)
D) Micronet 800	(11)

7. Three doors, each with a picture on it. The map says to follow the 16-bit fruit.

A) Tangerine	(14)
B) Apricot	(3)
C) Apple	(5)

8. This chamber contains a golden hexagon!

9. An octagonal room with a door in each wall and a small elf. On asking directions, he replies "The exit? Try 3 and 5 or 6. It's really quite logical!"

A) Door 1	(11)
B) Door 2	(6)
C) Door 3	(3)
D) Door 4	(7)
E) Door 5	(12)
F) Door 6	(2)
G) Door 7	(5)
H) Door 8	(10)

10. This chamber contains a golden triangle!

11. You enter a room through a trapdoor and find doors in front of you, behind and to the left and right. A message scrawled in the dusty floor reads " Make a copy on disc".

A) Go forward	(8)
B) Go back	(12)
C) Go left	(2)
D) Go right	(13)

12. The walls of this room are decorated with strange symbols:

0000	0010	0110	0111
0101	0111	0010	0101
0101	1001	1000	1100

Which reptile do you use as a direction indicator?

A) Asp	(7)
B) Mamba	(6)
C) Adder	(4)
D) Cobra	(2)

13. This chamber contains a golden square!

14. She was Charles Babbage's assistant, and the US Department of Defence has immortalised her. Who is she?

A) Ada	(9)
B) Agatha	(2)
C) Agnes	(11)
D) Alice	(3)

COMPUTING TODAY ADVENTURE COMPETITION NO. 2

Return to Computing Today, 1 Golden Square, London W1 before Jan 31, 1984

NAME. . . . .

ADDRESS . . . .

. . . .

. . . .

. . . .

My answers are . . . .

P. Jimmieson B.Sc

TRS-80 SCREEN EDITOR

Line editing is a bit of a dinosaur in computing circles, and this program will make it extinct for owners of the TRS-80. You have the choice of using the assembly listing or a BASIC version.

I have owned a TRS-80 for some time now, and have purchased a lot of software for it including various utilities of one type or another. Probably the best utility I ever bought was a full-screen editor, but one problem with it was that you were limited to line lengths of 54 characters. I therefore decided to write my own full-screen editor and the following program is just that. It works in a similar way to the Screen Editor for the BBC computer: you move a second cursor to the characters you wish to copy and press the Copy key.

You enter edit mode by pressing Control E (E for Edit). The control key is Shift Down-arrow. Upon pressing this key a flashing cursor will appear at the position of the normal cursor which you can then pilot around the screen using the arrow keys. Note that the arrow keys and the Clear key will lose their normal functions when edit mode is entered, though all other keys still retain their normal values, and the 'missing' keyboard characters can be accessed via Control keys:

Control H = backspace
Control I = tab
Control J = line feed
Control Y = up arrow
Control X = backspace and erase whole line.

You may copy characters from anywhere on the screen to the normal cursor (just as if you'd typed them) by pressing the Clear key. Any characters can be copied, including graphics characters, and you are limited only to the length of the Level II keyboard buffer — 240 characters.

To exit from the Editor, you may use Control E, Enter or Break. Control E will switch off the editor keeping any charcters that you copied intact, but restoring the function of all keys. Enter will enter the line that you copied as though you had just typed it. Break will throw away the line that you typed, just as though you had typed a line and pressed the Break key.

To run the program you may either type in the assembler listing and assemble it, or type the BASIC program listing which will poke the program into memory (not forgetting to save the BASIC program!). The BASIC program version is written for 16K machines only, does NOT relocate, is patched into the keyboard chain by executing the USR call, and must have had MEM size set at 32352 before running. The assembler version of the program will relocate itself to high memory, set its own MEM size, and patch itself into the keyboard chain. The editor may be entered at any time using Control E.

A few final points:

- The arrow keys and the copy key repeat if held down in edit mode. No other key will repeat unless you have loaded a key repeat program.
- The copy cursor will wrap around the screen.
- The program will not work properly in 32-character screen mode and will reset the screen to 64-character mode.
- All keyboard characters can still be accessed when in Editor, with the exception of the Clear key and Shift-Right arrow (32 characters per line mode) — this will prevent the screen display being inadvertenly messed up by the user.
- The program is compatible with Level II or DOS and does not alter the current keyboard driver, ie this program does not give you key repeat; but if you have such a program in memory this program will not remove it.

Listing 1. The corresponding BASIC listing to POKE the machine code into place.

```
20 REM Program Name :EDIT/bas
40 REM Create Date & Time :09/14/83 00:26:37
60 REM Full Screen Editor for 16K (Level II or Dos)
80 REM SET MEM SIZE TO 32352
100 CLS:PRINT"** Full Screen Editor **"
120 PRINT"Poking program into memory."
140 FORN=32354 TO 32767 :READ D
160 CS=CS+D
180 IF N)32767 THENI=65536-N ELSE I=N
200 POKEI,D:NEXTN
220 IFCS()44623THENPRINT"Error in Data":END
240 MS=INT(32747)/256:LS=32747-(MS*256)
260 IFPEEK(16396)=201THENPOKE16526,LS:POKE16527,MS ELSE DEFUSR0=32747
280 S=USR(0)
300 END
320 DATA0,60,0,0,0,0,0,0,42,32,64,34,98,126,126,50,100,126,175,50,61,64,211,255,
      58,127,56,183,32,250

340 DATA175,201,205,0,0,50,102,126,254,26,40,244,254,5,32,11,205,199,127,183,32,
      212,205,221,127,24,229,87,58,101,126
360 DATA183,32,4,58,102,126,201,122,254,13,40,4,254,1,32,10,205,221,127,205,199,
      127,58,102,126,201,42,98,126,58,104
380 DATA126,183,32,5,205,208,127,24,13,58,103,126,60,254,100,40,5,50,103,126,24,
      22,175,50,103,126,42,98,126,62,127
400 DATA190,32,3,58,100,126,119,1,184,11,205,96,0,58,60,64,42,98,126,71,230,32,4
      0,6,205,221,127,43,24,80,120
420 DATA230,64,40,6,205,221,127,35,24,69,120,17,64,0,230,8,40,8,205,221,127,183,
      237,82,24,53,120,230,16,40,12
440 DATA58,128,56,183,32,51,205,221,127,25,24,36,120,230,2,40,40,205,221,127,126
      ,71,254,32,242,57,127,203,247,71,230
460 DATA192,238,192,120,32,2,203,183,35,205,169,127,34,98,126,24,21,205,169,127,
      34,98,126,195,128,126,175,50,105,126,58
480 DATA102,126,183,200,205,221,127,245,42,32,64,17,64,0,25,124,254,64,32,11,125
      ,254,63,40,14,241,245,254,10,40,8
500 DATA241,254,25,32,2,62,91,201,42,98,126,175,237,82,205,142,127,34,98,126,24,
      234,245,213,124,17,0,4,254,60,250
520 DATA160,127,254,64,242,163,127,24,6,25,24,3,183,237,82,209,241,201,205,142,1
      27,245,197,58,105,126,183,32,5,1,32
540 DATA78,24,3,1,172,13,205,96,0,62,1,50,105,126,193,241,201,58,101,126,238,1,5
      0,101,126,201,126,50,100,126,62
560 DATA1,50,104,126,34,98,126,201,245,58,100,126,42,98,126,119,175,50,104,126,2
      41,201,42,22,64,34,131,126,33,130,126
580 DATA34,22,64,201,151,87,35,52,195,120,127,229
```

Listing 2. The assembler listing for the full screen editor.

```
                00100 ;the following line is only legal if you are using
                00110 ;the Misosys Edas assembler and should not be entered
                00120 ;if you are using any other assembler.
0000            00130         TITLE   '(Full-Screen Editor 1.4)'
                00140 ;EDITOR - version for Dos or Level II
6002            00150         ORG     06002H  ;for now
6002 7070       00160 START   DEFW    7070H   ;flag!
4020            00170 CURSOR  EQU     4020H   ;with a bit of luck
0060            00180 PAUSE   EQU     0060H   ;rom delay routine
403C            00190 ARROWS  EQU     403CH   ;arrow key row store
387F            00200 KEYON   EQU     387FH   ;scans all keys at once!
4E20            00210 DELAY1  EQU     20000   ;initial delay for repeat keys
0DAC            00220 DELAY2  EQU     3500    ;debounce delay
0064            00230 DELAY3  EQU     100     ;flash speed
3880            00240 SHIFT   EQU     3880H   ;shift key row address
0033            00250 DSP     EQU     0033H   ;rom video display routine
4049            00260 MEMSIZ  EQU     4049H   ;mem size in Doss
40B1            00270 MEMLII  EQU     40B1H   ;Mem size in Level II
400C            00280 SYSTAT  EQU     16396   ;In Level II this location
                00290                         ;contains a RET. In Dos it
                00300                         ;does not - hence we can tell
                00310                         ;by looking at this location
                00320                         ;whether we are in L II or Dos.
6004 003C       00330 CURS2   DEFW    3C00H   ;second (copy) cursor
6006 00         00340 CHAR    DEFB    00H     ;char @ second cursor
6007 00         00350 EDITON  DEFB    0       ;editor switch
6008 00         00360 STROKE  DEFB    0       ;keystroke store
6009 00         00370 CURS2F  DEFB    0       ;timer for flash
600A 00         00380 SAVED   DEFB    0       ;current-char-saved flag
600B 00         00390 RFLAG   DEFB    0       ;key repeat flag
600C 2A2040     00400 ENTER1  LD      HL,(CURSOR)     ;cold start
600F 220460     00410 RLD1    LD      (CURS2),HL
6012 7E         00420         LD      A,(HL)  ;get char
6013 320660     00430 RLD2    LD      (CHAR),A        ;store it
                00440 ;the next 3 instructions stop the program from running
                00450 ;in 32 chars per line mode.
6016 AF         00460         XOR     A       ;ld a with 0
6017 323D40     00470         LD      (16445),A       ;put us in 64 cpl mode
601A D3FF       00480         OUT     (0FFH),A        ;reset port
601C 3A7F38     00490 WASTE   LD      A,(KEYON)       ;test all keys in one
601F B7         00500         OR      A       ;any keys pressed?
6020 20FA       00510         JR      NZ,WASTE        ;wait 'til they let go!
                00520 ;this means that the cursor doesn't move until our user
                00530 ;has let go of the shift down-arrow and E  keys!
6022 AF         00540 THROW   XOR     A       ;ld A,0
6023 C9         00550         RET     ;return with 0 in A
6024 CD0000     00560 KEYRTN  CALL    0000H   ;the address of the current
                00570 ;keyscan routine will be put here when the program
                00580 ;is initialised.
6027 320860     00590 RLD3    LD      (STROKE),A      ;store it for now
602A FE1A       00600         CP      26      ;shft d-arrow (old-rom TRS-80's)?
602C 28F4       00610         JR      Z,THROW ;return with no key press
602E FE05       00620         CP      5       ;control E?
6030 2008       00630         JR      NZ,TRYEDT
6032 CD6961     00640 RLD4    CALL    TOGGLE  ;toggle value in (editon)
6035 B7         00650         OR      A       ;editor off? - z if zero
6036 2004       00660         JR      NZ,ENTER1       ;enter editor
6038 CD7F61     00670 EXIT1   CALL    RESTOR  ;turn off curs2 & restore char
                00680                         ;at cursor 2 from (char)
603B 18E5       00690         JR      THROW   ;return with a 0 in A reg
603D 57         00700 TRYEDT  LD      D,A     ;store A val
603E 3A0760     00710 RLD5    LD      A,(EDITON)
6041 B7         00720         OR      A       ;is edit on?
6042 2004       00730         JR      NZ,TRYSK3       ;yes
                00740 ;now process for (enter) and (break)
6044 3A0860     00750 RLD6    LD      A,(STROKE)      ;no - restore keystroke
6047 C9         00760         RET     ;go back with key
6048 7A         00770 TRYSK3  LD      A,D     ;check for Enter or Break
6049 FE0D       00780         CP      13      ;is it (Enter)?
604B 2804       00790         JR      Z,EXIT2 ;yes - (Enter) exit
604D FE01       00800         CP      1       ;is it (Break)?
604F 200A       00810         JR      NZ,EDIT1        ;no - a normal key
                00820 ;only come through here if (enter) or (break) pressed
                00830 ;and edit is turned on
6051 CD7F61     00840 EXIT2   CALL    RESTOR
6054 CD6961     00850 RLD7    CALL    TOGGLE  ;turn it off!
6057 3A0860     00860 RLD8    LD      A,(STROKE)      ;restore the char
605A C9         00870         RET     ;go back with it
605B 2A0460     00880 EDIT1   LD      HL,(CURS2)
605E 3A0A60     00890 RLD9    LD      A,(SAVED)       ;is this char stored?
6061 B7         00900         OR      A       ;zero if not
6062 2005       00910         JR      NZ,EDIT2        ;yes it is
6064 CD7261     00920 RLD10   CALL    STOR    ;no - go and do it
6067 180D       00930         JR      FLASH   ;display the flashing cursor
6069 3A0960     00940 EDIT2   LD      A,(CURS2F)
606C 3C         00950         INC     A
606D FE64       00960         CP      DELAY3  ;have we waited long enough?
606F 2805       00970         JR      Z,FLASH ;yes
6071 320960     00980 RLD11   LD      (CURS2F),A      ;no - put back counter
6074 1816       00990         JR      CONTN2  ;carry on processing
6076 AF         01000 FLASH   XOR     A       ;ld a,0
6077 320960     01010 RLD12   LD      (CURS2F),A      ;reset the timer
607A 2A0460     01020 RLD13   LD      HL,(CURS2)
607D 3E7F       01030         LD      A,127
607F BE         01040         CP      (HL)    ;was the cursor on display?
6080 2003       01050         JR      NZ,DISPLY       ;no - display cursor
6082 3A0660     01060 RLD14   LD      A,(CHAR)        ;yes - display char
6085 77         01070 DISPLY  LD      (HL),A  ;display the char or cursor
6086 01B80B     01080         LD      BC,3000 ;pause
6089 CD6000     01090         CALL    PAUSE
608C 3A3C40     01100 CONTN2  LD      A,(ARROWS)      ;status of arrow keys
608F 2A0460     01110 RLD15   LD      HL,(CURS2)      ;address of copy cursor
6092 47         01120         LD      B,A     ;store A
6093 E620       01130 TRYBA   AND     32      ;back arrow?
6095 2806       01140         JR      Z,TRYFA ;no - try forward arrow
6097 CD7F61     01150 RLD16   CALL    RESTOR
609A 2B         01160         DEC     HL      ;yes it was b.a.
609B 1850       01170         JR      TESTIT
609D 78         01180 TRYFA   LD      A,B
609E E640       01190         AND     64      ;forward arrow?
60A0 2806       01200         JR      Z,TRYUA ;no - try up arrow
60A2 CD7F61     01210 RLD17   CALL    RESTOR
60A5 23         01220         INC     HL      ;yes it was f.a.
60A6 1845       01230         JR      TESTIT
60A8 78         01240 TRYUA   LD      A,B
60A9 114000     01250         LD      DE,64   ;needed by next 2 routines
60AC E608       01260         AND     8       ;up arrow?
60AE 2808       01270         JR      Z,TRYDA ;no - try down arrow
60B0 CD7F61     01280 RLD18   CALL    RESTOR
60B3 B7         01290         OR      A
60B4 ED52       01300         SBC     HL,DE   ;yes it was u.a
60B6 1835       01310         JR      TESTIT
60B8 78         01320 TRYDA   LD      A,B
60B9 E610       01330         AND     16      ;down arrow?
60BB 280C       01340         JR      Z,TRYCLR        ;test the clear key
                01350 ;the following section is needed for New Rom trs-80's
                01360 ;it stops shift down-arrow (which returns a NOP on
                01370 ;the newer machines and is therefore not picked up by
                01380 ;the scan routine at the begining of the program) from
                01390 ;being picked up as just down-arrow which would cause
                01400 ;the copy cursor to move about!
60BD 3A8038     01410         LD      A,(SHIFT)       ;is it shift down-arrow
60C0 B7         01420         OR      A       ;test it
60C1 2033       01430         JR      NZ,CONTN3       ;yes - go deal with it
60C3 CD7F61     01440 RLD19   CALL    RESTOR  ;no its not
60C6 19         01450         ADD     HL,DE   ;yes it was d.a.
60C7 1824       01460         JR      TESTIT
60C9 78         01470 TRYCLR  LD      A,B
60CA E602       01480         AND     2       ;(clear) key pressed?
60CC 2828       01490         JR      Z,CONTN3        ;no - a different key
60CE CD7F61     01500 RLD20   CALL    RESTOR
60D1 7E         01510         LD      A,(HL)  ;get char at hl
60D2 47         01520         LD      B,A     ;store the char
60D3 FE20       01530         CP      32      ;is it a control char?
60D5 F2DB60     01540 RLD42   JP      P,OK003 ;no - its ok
60D8 CBF7       01550         SET     6,A     ;yes - make it an alpha char
60DA 47         01560         LD      B,A     ;store THIS char
60DB E6C0       01570 OK003   AND     192     ;is it a space compression code?
60DD EEC0       01580         XOR     192
60DF 78         01590         LD      A,B     ;restore char value
60E0 2002       01600         JR      NZ,OK004        ;character is ok
60E2 CBB7       01610         RES     6,A     ;yes - make it a graphics char
60E4 23         01620 OK004   INC     HL      ;move cursor 2 forward
60E5 CD4B61     01630 RLD21   CALL    CHKHLD  ;A reg is not changed!
60E8 220460     01640 RLD22   LD      (CURS2),HL      ;store new cursor value
60EB 1815       01650         JR      CHKSCR  ;is the screen going to scroll?
60ED CD4B61     01660 TESTIT  CALL    CHKHLD  ;check for under & over flow
60F0 220460     01670 RLD23   LD      (CURS2),HL      ;put back cursor value
60F3 C32260     01680 RLD24   JP      THROW   ;go get another character
60F6 AF         01690 CONTN3  XOR     A       ;ld a,0
60F7 320B60     01700 RLD38   LD      (RFLAG),A       ;reset repeat-flag
60FA 3A0860     01710 RLD39   LD      A,(STROKE)      ;restore key pressed
60FD B7         01720         OR      A       ;is a key being pressed?
60FE C8         01730         RET     Z       ;no - don't turn off cursor
60FF CD7F61     01740 CONT5   CALL    RESTOR  ;restore char at hl if key is
                01750                 ;passed back to caller (prevents screen being
                01760                 ;mucked up by screen scrolling)
                01770 ;check if screen will scroll
6102 F5         01780 CHKSCR  PUSH    AF      ;store the keystroke
6103 2A2040     01790         LD      HL,(CURSOR)
6106 114000     01800         LD      DE,64
6109 19         01810         ADD     HL,DE   ;see if we're on the last line
610A 7C         01820         LD      A,H
```

```
610B FE40               CP      40H     ;last line?
610D 200B               JR      NZ,OKRTN        ;no - don't worry
        ;to get here we're on the last screen line, and so we
        ;have to check our character
610F 7D                 LD      A,L
6110 FE3F               CP      63      ;last screen location?
6112 280E               JR      Z,ALTER2        ;char will cause a scroll
6114 F1                 POP     AF
6115 F5                 PUSH    AF      ;restore value in A reg
6116 FE0A               CP      10      ;line feed?
6118 2808               JR      Z,ALTER2        ;yes so we'll scroll
611A F1         OKRTN   POP     AF
611B FE19               CP      25      ;control Y ? (up arrow in edit)
611D 2002               JR      NZ,OK2  ;no
611F 3E5B               LD      A,91    ;ld a with up arrow char
6121 C9         OK2     RET     ;go back with key
6122 2A0460     ALTER2  LD      HL,(CURS2)
6125 AF                 XOR     A
6126 ED52               SBC     HL,DE   ;de was 64 already
6128 CD3061     RLD25   CALL    CHEKHL  ;keep HL on screen
612B 220460     RLD26   LD      (CURS2),HL      ;store it
612E 18EA               JR      OKRTN
        ;chekhl only alters hl if it lies outside of the range
        ;of the video display 3C00H to 3FFFH
6130 F5         CHEKHL  PUSH    AF      ;save the regs
6131 D5                 PUSH    DE
6132 7C                 LD      A,H     ;get msb!
6133 110004             LD      DE,1024 ;video size
6136 FE3C               CP      3CH     ;underflow?
6138 FA4261     RLD27   JP      M,TOOLOW        ;yes!
613B FE40               CP      40H     ;overflow?
613D F24561     RLD28   JP      P,TOOHI ;yes!
6140 1806               JR      OK      ;no problem
6142 19         TOOLOW  ADD     HL,DE   ;make it the right size
6143 1803               JR      OK
6145 B7         TOOHI   OR      A       ;clear carry
6146 ED52               SBC     HL,DE   ;make it the right size
6148 D1         OK      POP     DE
6149 F1                 POP     AF
614A C9                 RET
        ;check hl value and delay
614B CD3061     CHKHLD  CALL    CHEKHL
614E F5                 PUSH    AF
614F C5                 PUSH    BC
6150 3A0B60     RLD40   LD      A,(RFLAG)       ;get repeat-status
6153 B7                 OR      A       ;is the flag set?
6154 2005               JR      NZ,OK001        ;yes - do a short delay
6156 01204E             LD      BC,DELAY1       ;no - do a long delay
6159 1803               JR      OK002
615B 01AC0D     OK001   LD      BC,DELAY2       ;debounce delay
615E CD6000     OK002   CALL    PAUSE
6161 3E01               LD      A,1     ;set auto-repeat flag
6163 320B60     RLD41   LD      (RFLAG),A       ;store it
6166 C1                 POP     BC
6167 F1                 POP     AF
6168 C9                 RET
6169 3A0760     TOGGLE  LD      A,(EDITON)      ;toggle value in (editon)
616C EE01               XOR     1       ;toggle it
616E 320760     RLD29   LD      (EDITON),A      ;put it back
6171 C9                 RET     ;go back to caller
6172 7E         STOR    LD      A,(HL)  ;save char @ hl
6173 320660     RLD30   LD      (CHAR),A
6176 3E01               LD      A,1
6178 320A60     RLD31   LD      (SAVED),A       ;current char stored flag
617B 220460     RLD32   LD      (CURS2),HL      ;and store HL
617E C9                 RET
617F F5         RESTOR  PUSH    AF
6180 3A0660     RLD33   LD      A,(CHAR)        ;get old char
6183 2A0460     RLD34   LD      HL,(CURS2)      ;get posn of char
6186 77                 LD      (HL),A
6187 AF                 XOR     A       ;ld a,0
6188 320A60     RLD35   LD      (SAVED),A
        ;This routine is called prior to changing the value
        ;in (curs2) we will not have stored the char at this new
        ;location - Hence we flag this fact in (saved)
618B F1                 POP     AF
618C C9                 RET
618D E5         SETUP   PUSH    HL
618E 219E62             LD      HL,WELCOM       ;loading message
6191 CD2D62     SETUP2  CALL    VWRITE  ;my Video write routine!
6194 2A1640             LD      HL,(4016H)      ;get addr of current
        ;keyboard scan routine.
6197 222560     RLD36   LD      (KEYRTN+1),HL   ;use THIS routine for
        ;keyboard scan.
619A 212460     RLD37   LD      HL,KEYRTN       ;get addr of my rtn
619D 221640             LD      (4016H),HL      ;keyscan will go there
        ;we have inserted the routine into the keyboard scan
        ;vector, and our routine is now called instead. The
        ;current keyboard driver is preserved.
61A0 E1                 POP     HL
61A1 3A0C40             LD      A,(SYSTAT)      ;are we in dos or L 2 ?
61A4 FEC9               CP      201
61A6 C22D40             JP      NZ,402DH        ;Dos return
61A9 CD7A1E             CALL    1E7AH   ;basic clear fixes pointers
61AC C3191A             JP      1A19H   ;level II return
        ;relocating bit!
61AF 0000       NUBASE  DEFW    0000H
61B1 0000       DISPL   DEFW    0000H
61B3 310060     RELOC   LD      SP,START-2      ;Level II or Dos will
        ;reset their value for the SP on return from this
        ;routine. We only move the stack pointer to prevent our
        ;routine being overwritten by the stack during our
        ;set-up procedure.
61B6 E5                 PUSH    HL      ;save those registers
61B7 F5                 PUSH    AF
61B8 C5                 PUSH    BC
61B9 D5                 PUSH    DE
        ;the next line restores SYSTEM command
        ;the tape autostart places a JP instruction there
        ;so we put back the RET that used to be there
61BA 3EC9               LD      A,201
61BC 32E241             LD      (41E2H),A       ;set up a RET at 41E2H
61BF 3A0C40             LD      A,(SYSTAT)
61C2 FEC9               CP      201     ;test for level II
61C4 2805               JR      Z,NODOS1
61C6 2A4940             LD      HL,(MEMSIZ)     ;mem size in DOS
61C9 1803               JR      CONT2
61CB 2AB140     NODOS1  LD      HL,(MEMLII)     ;get mem size in level II
61CE 11B801     CONT2   LD      DE,SETUP-START  ;get program length
61D1 ED52               SBC     HL,DE   ;set up new mem size!
61D3 22AF61             LD      (NUBASE),HL     ;new base address
61D6 E5                 PUSH    HL
61D7 2B                 DEC     HL      ;value in mem size must be
61D8 2B                 DEC     HL      ;2 bytes less than required.
61D9 FEC9               CP      201     ;test for level II
61DB 2805               JR      Z,NODOS2
61DD 224940             LD      (MEMSIZ),HL     ;new mem size for dos
61E0 180A               JR      CONT3
61E2 22B140     NODOS2  LD      (MEMLII),HL     ;new mem size for level II
61E5 11CEFF             LD      DE,0FFCEH       ;clear 50bytes string spc
61E8 19                 ADD     HL,DE
61E9 22A040             LD      (40A0H),HL      ;string area pointer
61EC 110260     CONT3   LD      DE,START        ;get program start
61EF E1                 POP     HL
61F0 AF                 XOR     A       ;clear carry
61F1 ED52               SBC     HL,DE
61F3 22B161             LD      (DISPL),HL      ;HL gets how much each
        ;old address has to be altered by to relocate
        ;it to the New address.
        ;displ contains HL-DE  (newbase-oldbase)
61F6 213662             LD      HL,TABLE        ;table of addresses to
                                ;be relocated
61F9 7E         RELOC2  LD      A,(HL)  ;load DE,(HL)
61FA 5F                 LD      E,A
61FB 23                 INC     HL
61FC 7E                 LD      A,(HL)
61FD 57                 LD      D,A
61FE 23                 INC     HL
61FF B3                 OR      E       ;a zero entry in the table?
        ;The table is terminated by a zero entry so we know when
        ;to stop relocating!
6200 2816               JR      Z,NMORE ;return
6202 E5                 PUSH    HL      ;our table posn is now stored
6203 EB                 EX      DE,HL
        ;HL now has addr containing addr which needs altering
6204 E5                 PUSH    HL      ;store it - we're going to
        ;have to put our calculated address back later
6205 7E                 LD      A,(HL)
6206 5F                 LD      E,A
6207 23                 INC     HL
6208 7E                 LD      A,(HL)
6209 57                 LD      D,A
620A 2AB161             LD      HL,(DISPL)
620D 19                 ADD     HL,DE   ;add displacement to old addr
        ;new address now calculated - now put it back!
620E EB                 EX      DE,HL
620F E1                 POP     HL      ;addr of old address
6210 7B                 LD      A,E     ;ld (HL),DE
6211 77                 LD      (HL),A
6212 23                 INC     HL
6213 7A                 LD      A,D
6214 77                 LD      (HL),A
        ;address now rewritten
6215 E1                 POP     HL      ;restore table position
6216 18E1               JR      RELOC2
        ;only get out of it if no more addresses to relocate!
        ;Move the program up to high memory using an LDIR instr.
        ;We need to-
        ;Load HL with the Source address
        ;Load DE with the Destination address
        ;Load BC with the number of bytes to move
        ;and then LDIR'
6218 2AAF61     NMORE   LD      HL,(NUBASE)
621B EB                 EX      DE,HL
621C 21B801             LD      HL,SETUP-START
621F E5                 PUSH    HL
6220 C1                 POP     BC
6221 210260             LD      HL,START
6224 EDB0               LDIR
6226 D1                 POP     DE      ;restore the registers
6227 C1                 POP     BC
6228 F1                 POP     AF
6229 E1                 POP     HL
622A C38D61             JP      SETUP
        ;SETUP will NOT be relocated and so we can jump into it.
        ;
        ;VWRITE displays a message at the current cursor location
        ;using the ROM dsp routine. The message is pointed at by
        ;the HL register pair and terminated by a zero byte.
622D 7E         VWRITE  LD      A,(HL)  ;message byte
622E B7                 OR      A       ;zero byte?
622F C8                 RET     Z       ;end of message
6230 CD3300             CALL    DSP
6233 23                 INC     HL
6234 18F7               JR      VWRITE
        ;The following table contains all the addresses in the
        ;program that contain references whose values will
        ;change when the program is relocated. The labels
        ;themselves point to the actual instruction, and most
        ;of these are 3 byte instructions. An instruction of this
        ;type has the operation code as the first byte, and the
        ;address as the next 2 bytes. We want to change the
        ;address and so we add 1 to the value of the label
        ;which will point us at the reference we wish to alter.
        ; This is the reason for the "+1" after most of the
        ;references in the list. In the case of an instruction
        ;using 4 bytes a "+2" would need to be used instead.
6236 1060       TABLE   DEFW    RLD1+1
6238 1460               DEFW    RLD2+1
623A 2860               DEFW    RLD3+1
623C 3360               DEFW    RLD4+1
623E 3960               DEFW    EXIT1+1
6240 3F60               DEFW    RLD5+1
6242 4560               DEFW    RLD6+1
6244 5260               DEFW    EXIT2+1
6246 5560               DEFW    RLD7+1
6248 5860               DEFW    RLD8+1
624A 5C60               DEFW    EDIT1+1
624C 5F60               DEFW    RLD9+1
624E 6560               DEFW    RLD10+1
6250 6A60               DEFW    EDIT2+1
6252 7260               DEFW    RLD11+1
6254 7860               DEFW    RLD12+1
6256 7B60               DEFW    RLD13+1
6258 8360               DEFW    RLD14+1
625A 9060               DEFW    RLD15+1
625C 9860               DEFW    RLD16+1
625E A360               DEFW    RLD17+1
6260 B160               DEFW    RLD18+1
6262 C460               DEFW    RLD19+1
6264 CF60               DEFW    RLD20+1
6266 E660               DEFW    RLD21+1
6268 E960               DEFW    RLD22+1
626A EE60               DEFW    TESTIT+1
626C F160               DEFW    RLD23+1
626E F460               DEFW    RLD24+1
6270 0061               DEFW    CONT5+1
6272 2361               DEFW    ALTER2+1
6274 2961               DEFW    RLD25+1
6276 2C61               DEFW    RLD26+1
6278 3961               DEFW    RLD27+1
627A 3E61               DEFW    RLD28+1
627C 6A61               DEFW    TOGGLE+1
627E 6F61               DEFW    RLD29+1
6280 7461               DEFW    RLD30+1
6282 7961               DEFW    RLD31+1
6284 7C61               DEFW    RLD32+1
6286 8161               DEFW    RLD33+1
6288 8461               DEFW    RLD34+1
628A 8961               DEFW    RLD35+1
628C 9861               DEFW    RLD36+1
628E 9B61               DEFW    RLD37+1
6290 F860               DEFW    RLD38+1
6292 FB60               DEFW    RLD39+1
6294 4C61               DEFW    CHKHLD+1
6296 5161               DEFW    RLD40+1
6298 6461               DEFW    RLD41+1
629A D660               DEFW    RLD42+1
629C 0000               DEFW    0000H   ;Flag to relocate routine to
        ;indicate that there are no further references to be
        ;relocated.
629E 46         WELCOM  DEFM    'Full Screen Editor Version 1.4'
     75 6C 6C 20 53 63 72 65
     65 6E 20 45 64 69 74 6F
     72 20 56 65 72 73 69 6F
     6E 20 31 2E 34
62BC 0D                 DEFB    13
62BD 00                 DEFB    0
41E2                    ORG     41E2H
41E2 C3B361             JP      RELOC
61B3                    END     RELOC
00000 Total errors
```

Paul Gardner

LEARNING FORTH PART 3

Now you're getting more familiar with the FORTH dictionary, you may find you need more types of control structure and extra words. Here's how to branch and loop, and 'roll your own'.

No serious programming language would be complete without the necessary structures which allow you to repeat sections of a program or make decisions within a program. This month we will see what FORTH has to offer the 'structured programmer'. Also, by explaining the way definitions are stored in the dictionary we shall see how it is possible to create your own data types.

LOOPS AND BRANCHES

Several structures are available in FORTH for decision making or repeating, but the format of these is a little unusual when compared with languages such as BASIC. The first of these is the IF-ELSE-THEN structure. The way it works is best shown by an example. Assume there is a number on the top of the stack which represents the temperature in centigrade of a cooling unit.

```
: HEATINGMONITOR ( temperature --)
DUP ABS .
0<
IF ( if temperature below zero)
." degrees below freezing "
ELSE ( temperature is above zero)
." degrees above freezing "
THEN
;
```

If you try the program with a few different values you see the output is of the form "10 degrees below freezing" if − 10 was on the stack, or "10 degrees above freezing" if + 10 was on the stack. Figure 1 is a flowchart representation of this program.

The FORTH word ABS takes a value off the stack and returns its absolute value, ie the same number but with its sign ignored, so that it is left zero or positive. This is printed out using . . DUP makes sure that we still have a copy of the temperature, either positive or negative, on the stack for the next phrase to use.

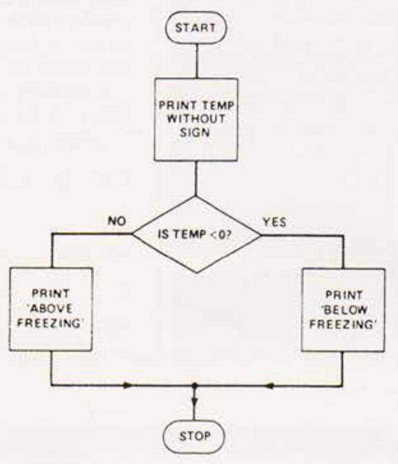


Fig.1. Flowchart of the IF-ELSE-THEN structure.

The next expression is the 'conditional phrase' 0<(less than zero). I will explain the use of conditional phrases shortly but for the moment we will assume that this phrase will leave a 'flag' meaning true or false on top of the stack for IF to use.

IF makes a decision between two paths: one from IF to ELSE, and the other from ELSE to THEN. The paths 'join up' again after THEN. IF bases its decision on the number on top of the stack (and it discards the number afterwards), so this number is the 'condition'. If the condition is 0 (false), it goes to the path between ELSE and THEN. If the condition is not 0 (true) it goes to the path between IF and ELSE. You can think of IF, ELSE and THEN as meaning:

IF the number on the stack is true follow this path
ELSE if it was false follow this one
THEN afterwards in either case carry on here.

You should notice here the difference between FORTH and BASIC with regard to the order of things:
BASIC — IF condition THEN action
FORTH — condition IF action if true ELSE action if false THEN.
With FORTH the conditional expression which leaves a true or false flag on the stack comes before the IF in the same way that the numbers used by the arithmetic operators come before the operator. Also, FORTH provides you with an ELSE section to follow if the condition yielded false. This ELSE section is optional and can be left out. For example a word that would use a number on the stack and give a warning if it was negative:

```
: BANK-MONITOR        ( balance --)
0<
IF
." Your account is overdrawn "
THEN
;
```

Since for numbers that are positive you don't need to do anything, you don't need ELSE.

In the previous examples, IF used a flag left on top of the stack by a FORTH testing word 0<(less than zero). Several words are available which operate on the top of the stack to leave either a true, 1, or false, 0, flag, as shown in Table 1. As many comparisons are based on the value 0, two words have been defined, 0= and 0<, which work in the same way as 0 = and 0 < (ie number space operator) but rather faster.

FOR MATHS BUFFS

Numbers or flags which are on top of the stack can be combined using the Boolean operators AND, OR and XOR. Each of these words takes the top two values from the stack and leaves the result. If the numbers on the stack were technically valid as flags (ie either 0 or 1) then the words act as true Boolean operators. If the numbers, however, were greater than one then the words AND, OR, etc act

=	(n1,n2 -- flag)	takes the top two numbers off the stack and tests to see if they are equal
<	(n1,n2 -- flag)	flag true if n1<n2, otherwise false
>	(n1,n2 -- flag)	flag true if n1>n2, otherwise false

Table 1. The comparison functions that are available.

as bitwise Boolean operators. For example:

```
1 Ø AND .
Ø ok
```

whereas:

```
4 6 AND .
4 ok
```

as the bitwise operator gives

```
ØØØØØ1ØØ  =  4
ØØØØØ11Ø  =  6
ØØØØØ1ØØ  =  4
```

LOOPING THE LOOP

We've all come across expressions in Basic such as

```
FOR A=1 TO 1Ø: PRINT A: NEXT A
```

FORTH offers a similar structure for repeating sections of a program, for example:

```
: COUNT-UP
11  1
DO
I  .  ( I returns the value of the
        counter for the loop)
LOOP
;
```

The DO-LOOP is a little different from the BASIC FOR-NEXT in several ways. The limit and the starting point are removed from the stack by the word DO and are kept in a 'safe place' (actually another stack called the return stack — more of that later). The test to compare the count and the limit is performed by the word LOOP so that the commands between DO and LOOP are executed at least once. Also, execution of the loop finishes when the count equals or exceeds the limit, so in my version of COUNT-UP the output would be:

```
1 2 3 4 5 6 7 8 9 1Ø ok
```

The word LOOP actually adds 1 to the count and then compares it to the limit. If they are not equal then the program carries on from just after DO; if they are equal then the looping stops. This is why the number 11 is not printed by COUNT-UP.

You should also note that the counter (which is held in that 'safe place') does not have a name like the BASIC FOR loop which used the variable A. There are, however, three useful words which allow you to keep track of the counter, shown above.

I	(-- n)	returns to the stack the value of the counter of the loop
I'	(-- n)	returns to the stack the value of the limit of the loop
J	(-- n)	will return the counter of an outer loop if one loop is nested within another

Table 2. Loop counter manipulation words.

For example:

```
: TIMES-TABLES
13  Ø
DO
13  Ø  DO CR
I  .  ." multiplied by "
J  ." equals "  I  J  *  .
LOOP
LOOP
;
```

If you do not want the count to increase by one on each loop then a word +LOOP is available which is shown below:

```
: COUNT-UP-IN-TWOS
1Ø  Ø  DO
I  .
2  +LOOP
;
```

The word +LOOP takes a value off the stack and adds it to the loop counter before comparing the counter to the limit. In this way the loop can be made to count down, by adding a negative number to the counter:

```
-2  +LOOP
```

It is possible to leave a DO-LOOP prematurely by using the command LEAVE, which sets the counter equal to the limit so that the loop will terminate at the next test.

BEGIN-UNTIL

A useful word in Abersoft's FORTH is INKEY; this returns to the top of the stack the ASCII value of the key being pressed (if any), or 255 if no key is being pressed. This is often used to make the computer wait for a key press before continuing:

```
: WAIT-FOR-ME
1ØØØ  Ø  DO  LOOP  ( gives you
       time to let go of enter key)
BEGIN
INKEY  255  <
UNTIL
;
```

UNTIL (flag —) expects a true (non-zero) or false (zero) flag on top of the stack. If the flag is false then control jumps back to BEGIN, if the flag is true then the program carries on after UNTIL: see Fig. 2. In this way an infinite loop can be set up using 0 UNTIL. (Don't try using this unless you want to turn off your machine soon!) For example:

```
: INFINITE
." I go "
BEGIN
." on and "
Ø  UNTIL
;
```

In this case the loop will never be terminated.

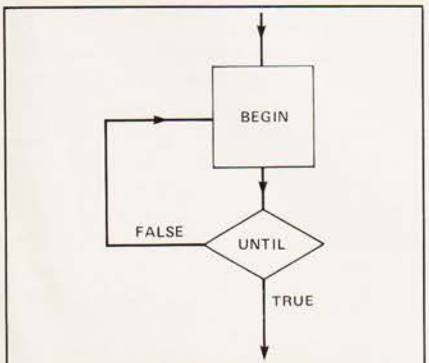


Fig.2. Flowchart for the BEGIN-UNTIL structure.

BEGIN-WHILE-REPEAT

You should have noticed that in the BEGIN-UNTIL structure, the commands between BEGIN and UNTIL are executed at least once. The test whether to loop around again is made by UNTIL. Verbally it goes:

BEGIN here and go round and round until the condition used by UNTIL is true.

There is another structure in FORTH which allows you to test a condition before the contents of a loop are executed. For example, using INKEY again we could have a small loop that waits until you let go of a key before continuing.

```
: WAIT-TILL-HE-LETS-GO  ( long
    titles are sometimes
    descriptive!)
BEGIN
INKEY  255  =  NOT  ( NOT changes
    the flag around)
WHILE
." Get your fingers off me " CR
REPEAT
;
```

The WHILE word expects a flag on the stack. If the flag is true the path between WHILE and REPEAT is followed; if the flag is false control passes to the point after REPEAT.

If the condition is true, then when execution reaches REPEAT control jumps back to the word BEGIN. It is usually the commands between BEGIN and WHILE that evaluate the flag for WHILE to use: see Fig.3. An example that uses both BEGIN-UNTIL and BEGIN-WHILE-REPEAT is given below. ▶

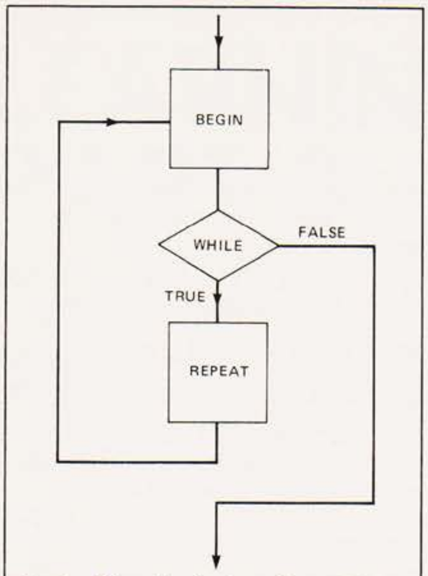


Fig.3. Flowchart for the BEGIN-WHILE-REPEAT structure.

Imagine your micro controls the home's heating system, and that you have defined a few words already as follows:

```
TEMP?  (-- n)
measures the temperature of the
room and puts the value on the
stack

THERMOSTAT?  (-- n)
reads the current setting of the
thermostat and puts the value on
the stack

HEAT-ON  (--)
turns central heating on

HEAT-OFF  (--)
turns central heating off
```

```
: TILL-HOT   ( waits until
               temp >= thermostat)
BEGIN
TEMP?  THERMOSTAT?  <
WHILE          ( do nothing)
REPEAT
;
```

```
: TILL-COLD  ( waits until
               temp <= thermostat)
BEGIN
TEMP?  THERMOSTAT?  >
WHILE          ( do nothing)
REPEAT
;
```

```
: HEATING-CONTROL
HEAT-ON         ( at start of system)
BEGIN
TILL-HOT   HEAT-OFF
TILL-COLD  HEAT-ON
Ø  UNTIL
;
```

JUST IN CASE

One of the most useful commands in a language such as Pascal is the CASE statement. This allows for the testing of a number for many different values and executing different procedures on each value (similar to the ON-GOSUB structure in some BASICS). This is available in standard FORTH only by using many nested IF-IF-THEN-THENs and can be very difficult to follow. A structure provided by Abersoft's FORTH is called the CASE structure. Its use is:

```
( instructions leaving a single
value on the stack)

CASE
 8  OF  ." This is 8"   CR  ENDOF
12  OF  ." This is 12"  CR  ENDOF
99  OF  ." This is 99"  CR  ENDOF
ENDCASE
```

This structure provides a much more readable and less errorprone way of making multiple decisions. In the above example, if 99 was the number on the stack, "This is 99" would be printed. Any value other than 8, 12 or 99 would produce no output at all.

For those without the Abersoft version, I'll demonstrate a program to implement the CASE structure in a future article in this series.

CONSTANTS, ARRAYS AND MUCH MORE

Last month I showed you how you can define variables for use in FORTH and explained that the shape in which any number is stored as a variable is part of the dictionary itself. You can see the name of any variable you define by doing a VLIST.

Another useful defining word in FORTH is the word CONSTANT; it is used in the form:

```
number  CONSTANT  name
```

For example:

```
12  CONSTANT  DOZEN
```

Like VARIABLE, CONSTANT sets up space in the dictionary where the number 12 is stored. But in this case, whenever you use the newly defined word DOZEN it is not the memory address of the storage location that is returned, but the contents of the location associated with the word DOZEN. Try it:

```
DOZEN  .
```

will produce:

```
12 ok
```

You can use a CONSTANT to store a number that is going to be used often without being changed, and where the program might be clearer if you referred to this number by a name. An example of this might be defining:

```
89  CONSTANT  KEY-Y
78  CONSTANT  KEY-N
```

so that if you use INKEY to put the ASCII value of a key being pressed onto the stack you could test for key Y being pressed using:

```
BEGIN
INKEY
KEY-Y  =
UNTIL
```

instead of the less obvious:

```
BEGIN
INKEY
89  =
UNTIL
```

STORING DEFINITIONS

So far we've seen three different kinds of words defined using : (colon), VARIABLE and CONSTANT. All words created in the dictionary have a similar structure and the principles that govern the creation of a new word allow you to invent your own types of definitions that are very different from the three types we have seen.

Before we can see how to create new definition types it is useful to see how definitions are stored in the dictionary. Each word in the dictionary consists of four basic elements.

Name Every word has a name; you can see all the names when you type VLIST.

Link field Every word has a two-byte address called a link field which holds the address of the previously-defined word in the dictionary. This link is used when the dictionary is being searched through.

Code field Every word contains another two-byte address called the code field. This address 'points' to a routine executed whenever the word is used.

Parameter field Almost every word has a 'parameter field'. In the case of variables and constants this field holds the number being stored. For a colon definition this field holds the list of FORTH words that make up the definition. The name, link and code fields have the same format for all FORTH words and can be regarded as a group as the header of the word.

A few diagrams here should clarify the way dictionary entries are stored. Let us assume we have just defined (in the following order) three new definitions for our dictionary:

```
39  VARIABLE  BOX
12  CONSTANT  DOZEN
: ?DOZENS  BOX  @  DOZEN  /  .  ;
```

The variable BOX holds a number representing the number of items in a given box.

DOZEN is a constant which holds the number 12.
?DOZENS when executed will print a number representing the number of complete dozens of items that can be taken from the box. In this case ?DOZENS would print '3 ok'.

The way these three definitions are stored is shown in Fig.4. In this diagram each horizontal line represents two bytes (often referred to as a 'cell') of the dictionary. The first byte of the definition holds the number of letters in the word's name. The next few bytes contain the ASCII codes for the characters in the name. The next two bytes are the link address which points to the previous definition in the dictionary. This is followed by the code field. It is this field that distinguishes a variable from a constant or a colon definition. The address contained in the code field points to a routine which is executed immediately the new word is called. For example, in our definition of BOX the code field points to a routine that will leave the memory address of the 'variable space' on the stack.

In the case of DOZEN the code field points to a routine that will copy onto the top of the stack whatever number is contained in the parameter field.

For ?DOZENS the code pointer points to a routine that will execute the list of FORTH words contained in the definition of ?DOZENS.

The final field in all definitions is the parameter field. For words defined using VARIABLE or CONSTANT the parameter field contains the number that is stored in the variable or constant. The parameter field of a colon definition (like ?DOZENS) contains the addresses of the previously defined words which comprise the definition. Here again is our example of ?DOZENS

```
: ?DOZENS  BOX  @  DOZEN  /  .  ;
```

When ?DOZENS is executed the definitions that are located at the successive addresses are executed in turn. The mechanism which reads the list of addresses and executes the definitions at each address is called the 'address interpreter'.

It is an important point that when a word like ?DOZENS is being executed the computer does not need to search through the dictionary for the words BOX, @ and so on. The dictionary search is done during compilation and the actual addresses of these words are enclosed in the dictionary definition of ?DOZENS. This is one of the things that makes FORTH so fast.

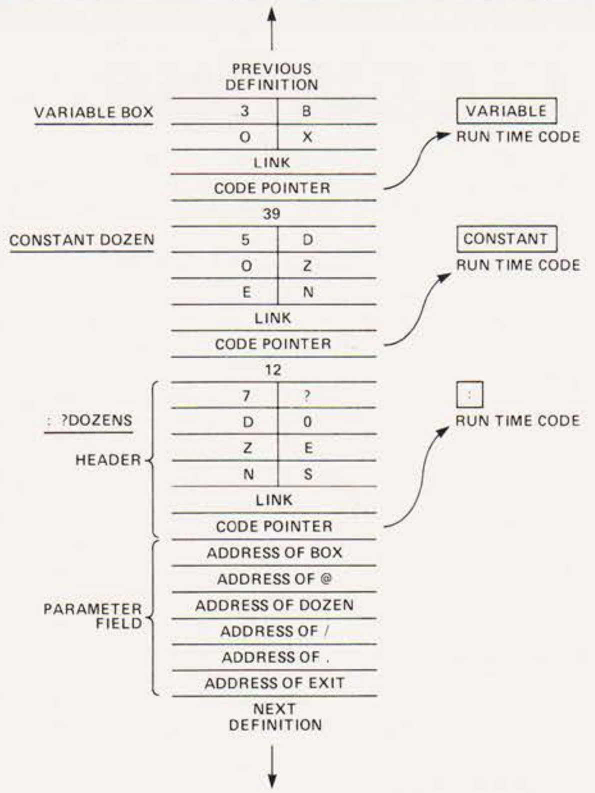


Fig.4. How FORTH stores the definitions given in the text.

The address of a word EXIT is compiled into the parameter field of a colon defintion to mark the end of the definition. This word lets the address interpreter 'know' when it has compiled that particular word.

We can see now, in outline at least, what happens during the execution of a colon definition. When you type a word like ?DOZENS, the keyboard interpreter looks for the definition in the dictionary. The code field of the word indicates what needs to be done to execute the definition. In this case the code pointer points to a routine which executes the list of addresses contained in the definition. The first address in ?DOZENS is that of BOX. Before BOX is executed, however, the computer puts the position of the next address in the list onto another stack, called the 'return stack'. This lets the computer 'know' where to come back to once it has finished with BOX. BOX is a variable, so it simply leaves the address of its parameter field on the stack. Execution then returns, using the pointer stored on the return stack, to the next address in the parameter field of ?DOZENS. The next address is that of @ and so @ will be executed next, and so on. This process continues until the EXIT in ?DOZENS, when the computer 'returns' to the keyboard.

An analogy that might help here is to imagine that you are following a list of instructions for a recipe, but that you don't know what some of the instructions like 'sauté', 'blanch' and so on mean. As you work down your list of instructions you have to leave the recipe to turn to another page where it tells you what to do to sauté something. Before you leave the recipe you make a note (a mental note perhaps) of what page the recipe is on and what instruction you have reached. Thus when you've found out how to sauté, you can return to the main recipe. The place you note your page numbers is your 'return stack' and in some cases the description of, say, sauté might include some other terms of expressions that you will need to find elsewhere in the book.

While it might be tricky for us to keep of what were doing if we can't cook very well and only understand instructions like 'turn gas on', 'stir contents of pan' etc, the computer will keep on 'turning the page' until it finds a definition it can execute directly, ie one in machine code.

We can now see that some of the definitions in the dictionary are defined as machine code instructions. All the other words in the dictionary are then (if somewhat indirectly) defined in terms of these definitions. ▶

BUILDING BLOCKS

Hopefully the explanation of the way definitions are stored will help you understand the way you can create your own 'defining words'. Let's use an example:

```
: CON  <BUILDS  ,  DOES>  @  ;
```

CON is a defining word which can be regarded in two parts. The first part (<BUILDS ,) creates a new dictionary entry and sets up the name, link and code pointer fields. The second part (DOES>@) decides what the new word does when it is executed, for example:

```
24  CON  2DOZEN
```

The word CON uses its first part, the 'defining part', to set up a new dictionary entry with the name 2DOZEN. The word , (comma) in CON takes the number 24 off the stack and encloses it in the parameter field of the new word. Figure 5 gives a diagramatic explanation of the dictionary entry.

When 2DOZEN is executed the code pointer points to the second part of the definition of CON. The word DOES> leaves the parameter field address of 2DOZEN on the stack and @ fetches the contents of this address and puts the number on the stack. So now, 2DOZEN (return) will leave 24 on the stack.

Our new word CON behaves just like the defining word CONSTANT. CON can now be used to define any other constant-type definition, such as:

```
13  CON  UNLUCKY
21  CON  KEY-OF-THE-DOOR
```

Let's try another example.

```
: VAR  <BUILDS  ,  DOES>  ;
```

Any word that uses <BUILDS-DOES> is a 'defining word'. When it is used like

```
Ø  VAR  AGE
```

it defines a new dictionary entry called AGE. The second part of VAR differs from CON only in that the word @ is missing from the part after DOES>. The word VAR now behaves exactly as VARIABLE. The first part (<BUILDS ,) sets up the dictionary header and the second part (DOES>) simply leaves the parameter field address of, for example, AGE on the stack whenever AGE is used. So:

```
AGE  @  .
```

will print:

```
Ø ok
```

The first part of a defining word we can refer to as the 'define time action', and the second part as the 'run time action'. Both of these parts can include other FORTH words, and need not necessarily refer to the parameter field of the new word at all. Here's another example.

```
: PRINTLETTER  <BUILDS KEY  ,  DOES>
@  EMIT  SPACE  ;
```

This can be used to define a word such as LETTERF:

```
PRINTLETTER  LETTERF    (Return)
```

Now press the key F and the computer will respond ok.

The define time action of PRINTLETTER sets up the header for LETTERF, then the word KEY waits for you to press a key. The ASCII code of the key you press is then enclosed in the parameter field of LETTERF, and that is the end of the define time action. When you now type:

```
LETTERF    (Return)
```

the code field of LETTERF points to the run time action of its defining word, ie PRINTLETTER. DOES> puts the parameter field address of LETTERF on the stack, @ fetches the contents of that address and EMIT is a FORTH word that prints the character corresponding to the ASCII code on the stack. SPACE simply prints one space.
So now:

```
LETTERF    (Return)
```

will produce

```
F ok
```

A defining word like PRINTLETTER can be used to define as many other words as you like, such as:

```
PRINTLETTER  LETTERG
```

All words defined using PRINTLETTER behave similarly, they differ only in whatever value was enclosed in the parameter field when they were defined.

As a final example I will show you a defining word which sets up an ARRAY in the dictionary:

```
: ARRAY
<BUILDS  Ø  DO  Ø  ,  LOOP  DOES>
;
```

This word can be used to define an array of any number of elements, such as:

```
12  ARRAY  MONTHS
```

which will set up an array of 12

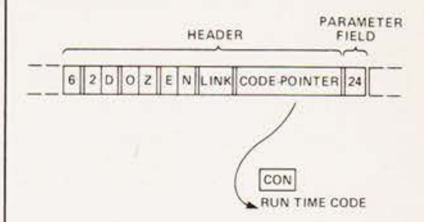


Fig.5. A diagram showing the structure of a dictionary entry.

elements: each element will be initialised to zero when the word is defined. This is done by the expression in ARRAY:

```
Ø  DO  Ø  ,  LOOP
```

Remember there is a 12 on the stack, so a loop is used that counts from zero to twelve and each time round encloses a zero in the dictionary.

When the word MONTHS is used the run time action of ARRAY is only DOES> so only the address of the parameter field is left on the stack and nothing else. This address is the address for the first byte of the array MONTHS. We can use any part of this array as we would use a variable.

```
MONTHS  @  .
```

would print the first element of the array.

```
MONTHS  2  +  @  .
```

would print the second element of the array as MONTHS 2 + would produce the address of the second element.

```
MONTHS  4  +  @  .
```

would print the third element of the array — remember each element occupies two bytes of memory.
This definition of an array is cumbersome, as you have to calculate the position of the particular element yourself. It would be handier to say

```
4  MONTHS  @
```

if we wanted the contents of the fourth element. Also, the definition does not check if you try to fetch the contents of an element that doesn't exist, for example

```
MONTHS  3Ø  +  @
```

for the fifteenth element.

Next month I'll continue with defining words and show you how to define an array-building command that works like DIM in BASIC. We'll also see how to create multi-dimensional arrays and even a 'monsterarray' for holding a monster's attributes in a valley-type game.

SNOWBALL

at £9.90 is a great new adventure for:

BBC 32K ***COMMODORE 64*** ***SPECTRUM*** 48K

LYNX 48K ***NASCOM*** 32K ***ORIC*** 48K ***ATARI 400/800*** 32K

Snowball is a massive adventure with over 7000 locations. it took nine months to perfect and marks a new leap forward in adventure games – it has a detailed, planned background and is set aboard a huge starship that would really work. Snowball could be a glimpse of the future!

You play Kim Kimberley, security agent. Your mission is to guard the colony ship Snowball 9 from sabotage.

Thus when your freezer-coffin wakes you with the Snowball still in flight, you know that something must be very wrong. You're weakened and disorientated by lengthy hibernation, but the fate of the 5 mile long space-ship is in your hands!

Snowball is our new fourth adventure. Here's what the reviewers have just said about the first three:

"The Level 9 Adventures are superbly designed and programmed, the contents first rate. The implementation of Colossal Cave is nothing short of brilliant; rush out and buy it. While you're at it, buy their others too. Simply smashing!"
- SOFT, September 83

"Of the programs reviewed here, the only one that is wholly admirable is Level 9's Colossal Adventure."
- Your Computer, September 83

"I found Dungeon exceedingly well planned and writen, with a fast response. There are well over 200 locations and the descriptions are both lengthy and interesting."
- Computer & Video Games, September 83

"This has to be the bargain of the year . . . If adventures are your game, then this is your adventure".
- Home Computing Weekly, 30th August 83

"There are three excellent adventures on offer from Level 9 . . . the descriptions are so good that few players could fail to be ensnared by the realism of the mythical worlds where they are the hero or heroine . . . great fun to play."
- Which Micro?, August 83

MIDDLE EARTH ADVENTURES

for the same micros as Snowball

Each of these games has over 200 locations and a host of puzzles. They can be played singly or together as an impressive trilogy. Each game could well take months to solve!

1) Colossal Adventure
The classical mainframe game "Adventure" with all the original puzzles plus 70 extra rooms.

2) Adventure Quest
An epic puzzle journey in Middle Earth.

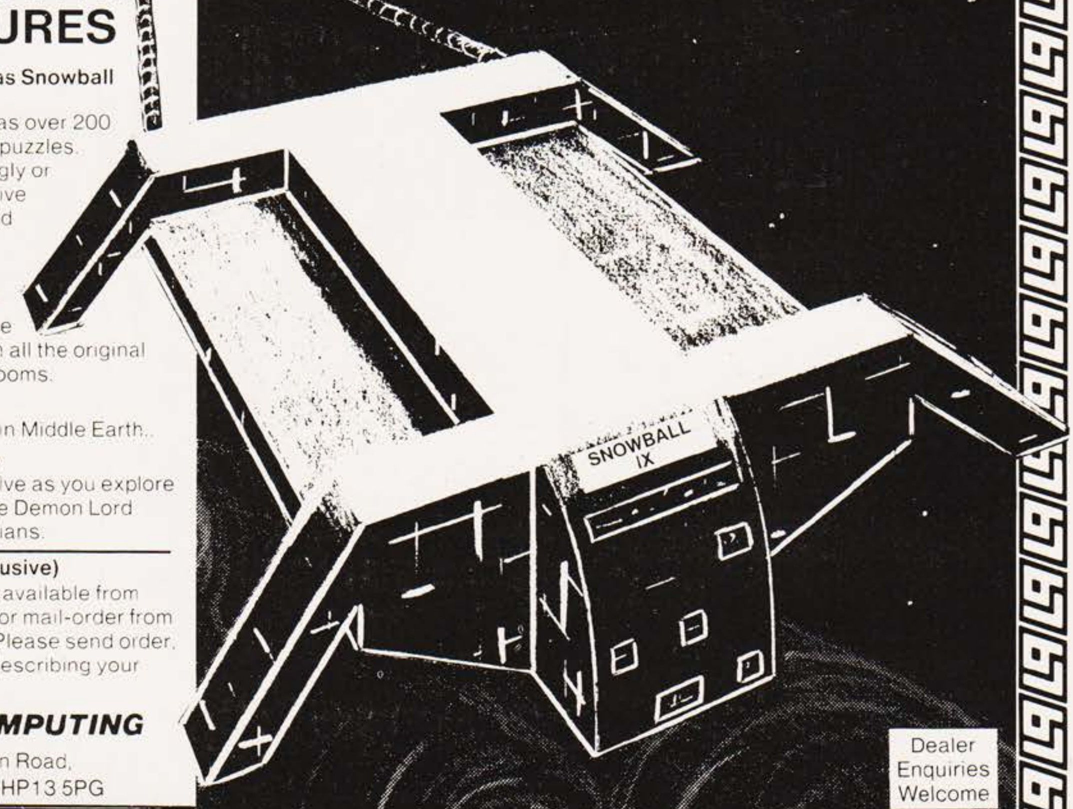
3) Dungeon Adventure
Over 100 puzzles to solve as you explore the dungeons left by the Demon Lord – outwitting their guardians.

Price: £9.90 each (inclusive)

Level 9 adventures are available from good computer shops, or mail-order from us at no extra charge. Please send order, or SAE for catalogue, describing your micro to:

LEVEL 9 COMPUTING

Dept C, 229 Hughenden Road,
High Wycombe, Bucks HP13 5PG



Dealer
Enquiries
Welcome

Roger Glanville

VIEWPOINT

If you've been having problems learning to use your brand new micro it may be a comfort to know that you're not alone!

For many months I had been increasingly intrigued by the idea of having a word processor; all those words appearing as if by magic on the screen, that little flashing square shooting all over the place to bring paragraphs from other pages, checking the spelling and inserting names and addresses of people I didn't know into personalised letters. What a fascination! I had to have one.

But could I really justify such a glorified piece of paper and pencil? What could I use it for? I assumed this curiosity would be just a passing fancy and with any luck would finally go away.

However, it didn't go away and as 1982 advanced and we were increasingly bombarded by articles and adverts reminding us that this was IT82 and as I didn't have any other expensive hobbies, no Rolleiflex, no Quad electrostatic speakers, no carbon fibre fishing rod, no Yamaha piano or anything — I finally succumbed!

YOU PAYS YOUR MONEY

I bought an Apple II. I have been using it for a fortnight and it's driving me crazy. Is that par for the course? Insanity in two weeks? 'This bloke is nuts,' you must be saying. 'Everyone knows that the Apple is brilliant, so he must be an idiot.'

Well, I can't agree with that. I am just an ordinary sort of fellow, quite bright I reckon, but no computer genius — I still spell program programme. So when the engineer came to install the machine (and take my cheque), out of every box he removed a beautiful piece of hardware and a manual. All the bits of hard and firmware (you see, I'm learning!) he hooked together to give me a system. All the manuals he piled on the desk — to give me 1821 closely printed A5 pages to read!

Undaunted, I started work. My logic had been simple. I couldn't afford a business word processor at anything upwards from two and a half thousand pounds, I didn't know enough to risk a two hundred quid engineering solder job and I wanted to keep some options open, like using Pilot. So in front of me were manuals for the Apple, Applesoft, DOS, Videx, Zardax, Digitek, Epson, parallel interface, Philips video monitor, VisiCalc and Pilot.

As I had used Pilot before and was familiar with the first few pages of Zardax, I started with the word processing. The disc had been 'set-up', so I booted it. Drive 1 lit up, whirred and stopped. OK. Press 'S' for setup, any other key to continue. Tap the Space bar. Drive 1, red light, whirr, whirr, 10 seconds, 20, 30, 45, 60. Open the manual and read "it takes about 30 seconds to complete the boot process". Look in the DOS manual! 'if booting doesn't work, . . . re-read the manual carefully — that cures 90% of all problems." It's now two minutes and the disc is still whirring round. Read on: "you'll have to press the RESET key to stop it (normally this is a BAD idea)". I panic. My disc is still going round. Does the bit I have just read mean that it's OK for an uninitialised disc to be stopped with Reset but not for a disc with something written on it? No idea, no option. Press Reset.

I tried the other disc and that worked all right, so I tried the first one again. No joy, press Reset. I carried on working with the back-up, all went well and I was very impressed with the facilities offered by the package and the 80-column screen. After writing some short pieces I turned to the printer. I entered a range of print instructions according to Zardax rules and hardly anything worked properly. Back to the manual. Tucked away in the back flap was a piece of A4 entitled "Errata" and tucked away in that was another instruction. "Simply insert the following line into SETUP." So I entered the Main Menu and looked for the way in. There ain't no way in.

GETTING INTO PRINT

Mike, the engineer who had installed the machine, explained to me on the phone how to use DOS to list the SETUP program, unlock it, add the required line and BINGO I'd got my print — well, some of my print — instructions to work. Now my interest was aroused. Could I print the Pilot programs I had written at Christmas? I've got a dot matrix printer and it is in Slot 1, so it should work as the Pilot printing arrangements are for a Silentype in Slot 1.

I put the relevant discs in, ran the programs, pressed 'P' for print and waited. One second, click, buzz, silence. Why won't it work? That's not a rhetorical question, by the way. I still don't know the answer, but, and this is the most important point, I don't know why the manuals don't tell me the answer.

Other answers not given in any of the manuals are why the printer, if asked to print something from page 2 of a document, can apparently remember from the top line of page 1 that the right margin is 125 but forgets that it's supposed to be using condensed print and why, if it has been printing double-width characters and is given the single-width instruction, it suddenly slips into the condensed typeface.

The reason underlying this kind of problem seems to be that the hardware and software come from different manufacturers whose primary concern is to match the electronics and make the equipment work. Having done so, they can market their products as being designed for use with a particular system, but each manufacturer then produces a manual for his product and the manuals don't square with each other. A few examples might support my point, because I don't want readers or suppliers to think this is just another of the easily written 'knock the manuals' pieces.

Let's look at the way the manuals deal with the ASCII code. You must remember that since I was looking mainly for word processing facilities in the first instance, I had no idea what the code meant, but I now realise that every damn thing that moves in my system is using it.

The Zardax manual doesn't mention it. All the formatting and printing instructions are given as simple keystrokes, which I think is the best way of doing it since the average Pom likes to know what to press. However, there could be additional reference pages matching the Zardax commands to the ASCII code.

The Epson manual has the direct manner of a Japanese Technical College lecture. It is fairly easy to grasp at the unpacking stage but gets quite complex later. For instance, my printer kept chewing up the paper until I found "Do not lock the release lever" hidden away in a note on page 26. Control codes are given in a form which makes no sense to me, because you can't, sorry, I can't (as far as I can tell) use the ASCII decimal codes for all the instructions to the printer. Even from within a BASIC program a DOS instruction has to have the ESCape bit in CHR$(n) form followed by a letter or number in quotes. Do I sound like a drowning man? I am certainly out of my depth! I have just been picking bits out of various sample programs (is that better?) and trying them out. Sometimes they work, sometimes they don't.

Apple manuals are generally well thought of, but even in these the ASCII information is inadequate because on page 138 of the **Applesoft reference manual** a neat chart gives just about everything required except the particular bit I need which is the full meaning of each character so that I can match it up to the bits left out of the **Epson Interface Kit manual.**

GRAPHICS OR NOT?

What I have been trying to do is to use Applesoft and DOS commands to gain total control over my printer for the purposes of Zardax. But I have been running into a lot of problems which I am absolutely certain are fairly simple but which I am not trained to solve. Surely the routines could be included in the manuals in keystroke form.

The Videx manual gives some ASCII information, but this time it takes the form of a grid relating the characters to decimal and binary equivalents. How will I ever find out what SOH, ETX, BS, etc mean? BEL I can manage!

Of course, having a Videx card is great for the 80 columns on the screen but adds to the complication slightly by offering alternative methods of obtaining upper/lower case displays. For fun I copied into the Apple the listing of the program Point Symmetry by G W Gallagher (*Computing Today* – May 1982) but as I had the Videoterm card installed I decided to put all the screen instructions in lower case. I can hear from the chuckles that you are all ahead of me again! That left me either with text and no graphics on the monitor or with reflections in graphics and gobbledegook in text. Yet, I have found no reference to the fact that the lower case instructions would not work.

This idea of using programs published in magazines is very helpful to the slow learner like me. Having established that Mrs Gallagher's printer routine worked with the Epson, although it had been written for the Silentype, I lifted that bit out and tacked it on the end of another short program, but it wouldn't work. So back to the manuals to find out what all this POKE – 12528,7 was about. They don't say.

For sound though, you POKE around in the – 16336 area. It says in the Applesoft manual that you can obtain higher notes by increasing the speed of the loops but so far I have only managed to get lower notes. I was impressed to hear the Apple version of Boris Christoff coming from the 2" speaker. Kiri Te Kanawa must be in there somewhere, but she's eluding me at present.

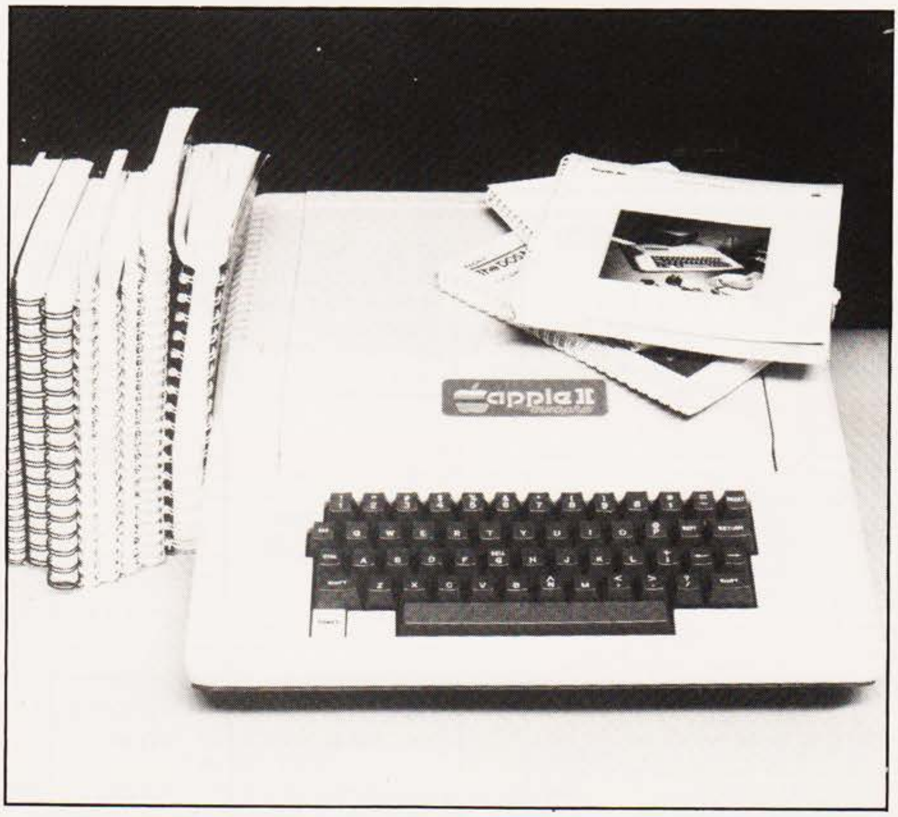
To return to the word processing, however, reminds me that I want a routine for counting the words in an article. I assumed that I could arrive at a good approximation by counting the spaces. Apparently, Zardax stores its texts as textfiles and I have tried retrieving them using the sample programs in the DOS manual. In this venture only partial success can be reported because among the words on the screen was a load of junk I didn't write; lots of ????s and REENTERs. I don't think I can win with that one.

BUT IN THE END...

The upshot of all this is that I am very happy with the system but very frustrated with some aspects of the manuals. On page one they all talk to me as if I am an absolute beginner; for example, "This manual was in the accessory box. This box should also contain the Apple's power cord (the cord that plugs into the outlet on the wall)." That's a pretty simple start, matched by the Videx approach; "Is the TV monitor turned on? Is it plugged into the power outlet?" and the Zardax introduction, "Zardax is an easy to use Word Processor and Text Editor for the Apple II Plus Computer."

I found all these words helpful, comforting and reassuring when I started. My frustration is that by the time the writer of the manual arrives at the back of his book he thinks he is dealing only with the highly skilled programmer who speaks fluent Hex, assembler and machine language. There must be many others like me who are threshing about in the dark trying in a very hit and miss way to make procedures work.

The guides and reference books cope then with the beginners and the brilliant. What about us average manual workers?



COMPUTAMART

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

BERKSHIRE

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

CHESHIRE

northern computers
Churchfield Rd., Frodsham.
Tel: (0928) 35110
Open: 6 days 9-6. Retail and Wholesale. Apple II & III, Atom, BBC, VIC20/64, Newbrain, Dragon 32, Electron, Spectrum. All accessories. Easy parking off M56.

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

CLEVELAND

BRIERS COMPUTER SERVICES
Polytechnic Bookshop (at the Poly)
1 King Edward Square, Middlesborough.
Tel: 0642 242017.
"Everything for the Genie computer".

CORNWALL/DEVON

A. B. & C. COMPUTERS (CT)
Duchy House, 6 Lower Aylmer Sq., St. Austell.
Tel: 0726 64463/67337
Wide range of popular Micros, Printers, books and accessories. We stock all U need — try us first for service and competitive prices

PLYMOUTH'S nascom DEALER
S & R BREWSTER LIMITED
86-88 Union St., Plymouth PL1 3HG.
Tel: 0752 665011 Open: 6 days.

DORSET

West Dorsets Nascom Dealer
Business and Personal Computer Systems, Printers, Software. Tel: Bridgeport (0308) 56539
Mail Order Service operated.
Access/Barclaycard accepted.

HERTFORDSHIRE

NEWBRAIN SPECIALISTS
HARDWARE & SOFTWARE
Printers, Epson, Shinwa, Juki etc. Monitors. Tape Recorders. Books. Expansions. CP/M. **Mail Order, Access.** 2 new books — see classified ad for details.
ANGELA ENTERPRISES
Tel: Stevenage (0438) 812439 anytime

NORTH KENT

MEDWAY COMPUTERS
141 NEW ROAD, CHATHAM.
Tel: 0634 826080
ACORN COMPUTER
Open Mon-Sat 10 - 5. Closed Weds.
Most computers and software stocked.

LANCASHIRE

LANCASHIRE MICROS
51 QUEEN STREET, MORECAMBE.
Tel: 411435. Also open Sundays.
Sinclair, Commodore, Acorn/BBC + Dragon. Authorised dealer for Bug-Byte, Imagine, Quicksilva, Artic, Melbourne House, Silversoft etc.

BUG-BYTE AUTHORISED DEALER
LEIGH COLOUR LABORATORY LTD
87 Chapel St., LEIGH,
Tel: 0942 607661
Open: 9 — 5.30.

GREATER MANCHESTER

A & F SOFTWARE
830 HYDE ROAD, MANCHESTER.
Tel: 061 223 6206
Open: Mon-Fri 9.30-5.30, Sat 9.30-5.00
Full range of software/hardware for most micro computers. Credit cards accepted.

NSC COMPUTER SHOPS
29 Hanging Ditch, Manchester.
Tel: 061 832 2269
Open: Mon-Fri 9.30am-5.30pm.
Sat 10-5. Retail and Wholesale.

LINCOLNSHIRE

SHARP CENTRE
16 Melville Street, Lincoln.
Tel: Lincoln 32379.
Open: 9am-5.30pm closed Wed.

LONDON

HENRY'S COMPUTER SHOP
404-406 Edgware Road, London W2 1ED.
Tel: 01-402 6822
Open: 6 days a week. Order by phone or call in and see for yourself.

BYTESHOP COMPUTERLAND
324 Euston Road, NW1.
Tel: 01-387 0505.
Open: Mon-Fri 9am-5.30pm.
(C.R.A. member)

SOUTH LONDON

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

MIDDLESEX

L.B. ELECTRONICS
11 Hercies Rd, Hillingdon.
Tel: Uxbridge 55399 (24hr ans. service)
Open 6 days, 9.30am-6pm, (lunch 1-2.15 except Sat) Surplus equipment, memory, EPROMs etc. Also established mail order service.

NORFOLK

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

NORTHAMPTONSHIRE

NORTHAMPTON HOME COMPUTER CENTRE
58A Wellingborough Road, Northampton.
Tel: (0604) 22539
Open: 6 days a week from 10 - 6

NORTHERN IRELAND

NEWBURN Ballycorry Co. Antrim
6 Days until 8pm. Gemini, Galaxy, Nascom, Acorn Computers. Discs, monitors, printers, servicing, hire. Industrial Control, Accounts, Word Processors etc.
WHITEHEAD 78330

SCOTLAND

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

JARAK SOFTWARE CORNER
Copyg.l Print Centre, 98-100 Bath Street, Glasgow G2.
Specialists in Software and Peripherals only. Tel: 041 336 0226

STAFFORDSHIRE

COMPUTER CABIN
24 The Parade, Silverdale, Newcastle.
Tel: 0782 636911
VIC-20, ZX81 hardware and software. BBC, Atom, Spectrum software.

COMPUTAMART

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

COMPUTING TODAY PRESENTS YOUR OWN 'WHERE TO BUY IT' GUIDE.

SUSSEX

CROWN
56-58 South Street, Eastbourne.
Tel: Eastbourne (0323) 639983/20496
Open: 6 days 9am-5.15pm.

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

TYNE AND WEAR

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

WALES

SIR Computers Ltd.
91 Whitchurch Road, Cardiff.
Tel: 0222 - 21341
BBC — ACORN COMPUTER

STEVE'S COMPUTER COMPANY LTD
CASTLE ARCADE, CARDIFF
Tel: 0222 — 371578
OPEN: Mon — Sat. 9 — 5.30
FOR COMPONENTS TO COMPUTERS

WARWICKSHIRE

OEM Rugby MicroCentre
OEM COMPUTER SYSTEMS
9-11 Regent Street, Rugby CV21 2PE,
Tel: (0788) 70522/3/4
THE FINEST COMPUTER SHOWROOMS IN THE MIDLANDS

CARVELLS OF RUGBY LTD
3/7 Bank Street, Rugby CV21 2QE
The Acorn/BBC Specialists
We also stock Texas T199, VIC 20, Dragon 32, Commodore 64, Sinclair ZX81 / Spectrum, Oric, Atari Disk drives, Epson, Juki, Shinwa & Olivetti Printers and Books.
Tel: Rugby 65275/6

LOOKING FOR MICROCOMPUTER HARDWARE OR SOFTWARE? LOOK NO FURTHER THAN COMPUTAMART!

WARWICKSHIRE

BUSINESS & LEISURE MICROCOMPUTERS
16 The Square, Kenilworth.
Tel: Kenilworth 512127.
Open: Mon-Fri 9am-5pm, ½ day Thur (lunch 1-2). Retail and Wholesale.

YORKSHIRE

BRADFORD'S COMPUTER SHOP
at Thomas Wright (Bradford) Ltd., Thorite House, Laisterdyke.
Tel: Bradford 668890.
Open: Mon-Fri 8.45-5.30. (Sat 12am)

SYSTEM-SOFT
THE BEST FROM THE REST
Freepost, Ackworth, Pontefract WF7 7BR
We stock an excellent range of software from leading manufacturers for Spectrum, Dragon, Vic 20, BBC.
SAE for FREE catalogue — Mail Order Only

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

Business Name: ....

Address: ....

....

....

Tel. No.: ....

Open Hrs: ....

Contact (Office Use Only): ....

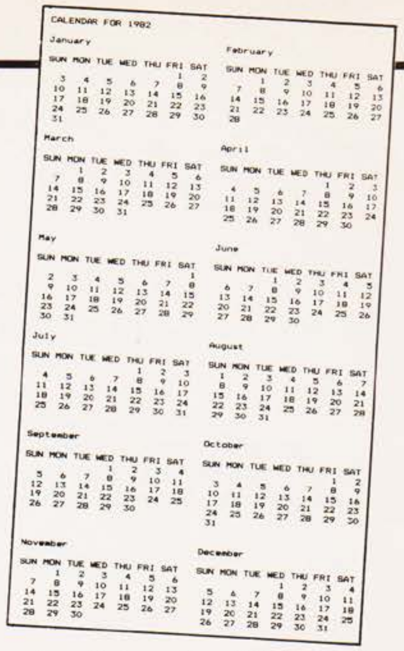
Post To: Computamart, Computing Today, 145 Charing Cross Rd., London WC2H 0EE.

ONLY £17.50!

Don Thomasson

MAKE A DATE

This is the January issue, so you'll be reading it in December (confusing, isn't it!) and at this time of year our thoughts turn to calendars. Here's a comprehensive piece of programming that should cover most requirements.



CALENDAR FOR 1982

January

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

February

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

March

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June

SUN	MON	TUE	WED	THU	FRI	SAT
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

August

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

September

SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

October

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

November

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

December

SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Calculation involving dates can be very complicated, because of the varying number of days in a month and the incidence of leap years. The program given here provides a possible solution to the problem. Written in Standard BASIC, it should not be difficult to convert to other BASIC forms.

The program begins by setting up the arrays:

MA gives the number of days in each month.

MB gives the number of days in the year before each month.

MC$ gives the names of the months.

D$ gives the names of the days of the week.

A menu of three options is then offered, the choice being made by keying in C, M or Y. These function as follows:

C: Date calculations can be performed, the input formats being:

18.7.1919 The date given is set up, and displayed in full, with the day of the week named.

– 4.6.1919 The date given is subtracted from the date which has been set up, and the difference in days is displayed.

+ 500 The number given is added to the set date as a number of days, and the resulting date is displayed. This date becomes the set date.

– 423 The number given is subtracted from the set date as a number of days, and the resulting date is displayed. This date becomes the set date.

E The routine exits to the main menu.

M: Input of a year and a month (the month as a number) will generate a calendar for that month.

Y: Input of a year will generate a calendar for that year with two columns of months side by side.

All this can be done for any year from 1600 AD on. Why 1600 AD? Well, there was a reform of the calendar in the sixteenth century, and it seemed easier to dodge the need to cope with this. (Coward! Ed. Agreed. DWT)

CALCULATION MODE

The calculation mode is handled by lines 400-680. When the input has been made, an 'E' results in return to the menu at 220. Otherwise, the P or L flag is set if the input starts with a + or − sign. The input is then scanned character by character, and the numerics are assembled in E$. When a full stop is found, the currently assembled number is set in a variable, the value of F determining whether the variable should be Y, M, D or N, the latter applying when no full stops are found.

If the input is a date, subroutine 1000 is called at line 560 to calculate a 'base number' for the given year, taking 1st January 1600 as 1. The number of subsequent years is multiplied by 365, and corrections are then applied for the intervening leap years: one is added for each year divisible by four; one is subtracted for each year divisible by 100 (not a leap year); and one is added for each year divisible by 400, starting with 1600.

Some points regarding this routine need comment. The 'ABS' functions are included because for the year 1600 the overall terms would otherwise give − 1. The two terms would cancel each other out, as matters stand, but it seemed worth drawing attention to the point. The figure subtracted is 1601, not 1600, because the effect of a leap year does not show until the start of the subsequent year. In line 1020 a value of 1201 is subtracted, because 1600 itself is a leap year, and 1 must be added to the base for 1601.

The base number for the given day is then set in N by adding MB(M) and D to the year base number. Subroutine 1100 is called to set YL = 1 if the current year is a leap year, and YL is added to N if M>2, ie if the month is March or later, when the effect of the extra day in February must be taken into account.

Any arithmetic indicated by the flags P and L is then carried out, using MA to contain the set value. Otherwise NA is set from N. If a date has been subtracted, L = 1 and F = 2, in which case N is displayed. Otherwise, the date corresponding to N is displayed. Purists could extend this to report errors for illegal operations, such as adding two dates together.

DATE CALCULATION

Deriving N from a given date is fairly painless, but the reverse process, carried out by subroutine 1200, is more difficult. One program used a loop to check off each year individually until the total matched N. Here, an approximation is first made by dividing N by 365.25. This will usually give the correct year, but not for all dates. For instance, 1.1.1982 gives 1981 with a remainder of 366 days. As 1981 was not a leap year, this is detected as an error by line 1240, the year is incremented and the surplus days recalculated as 1.

The correct month is then identified by reference to array MB, not forgetting to add 1 for March onward in leap years. An iterative loop seemed permissible here.

Finally, the day of the month is found and the original value of N is restored from NH.

MONTH CALENDAR

The month calendar routine at 3000 is fairly straightforward. Once the year and month have been input, two parameters determine the printout.

BA is set from MA(M) to the number of days in the month, YL being added for March onward. The process continues until BA days have been entered. The value of N for the 1st of the month is calculated, and GA is set to (N+5) modulo 7. This indicates the day of the week on which the month starts, and the printout of each numeral is positioned by tabbing 4*GA columns. The extra term in the tab bracket, −(W<10), adds another space for numbers less than 10, obtaining right justification.

As printing proceeds, GA is incremented modulo 7, and this ensures continued correct positioning of the numbers.

YEAR CALENDAR

The year calendar program is more complex. Because two month calendars are printed side by side, BA and GA have to be duplicated, and arrangements must be made to cater for either of the two months occupying fewer lines. The principle is otherwise the same as that used for the single month output.

No provision has been made for switching the calendar outputs to a printer, because that will vary with each type of machine. Where LPRINT is available, there is no problem.

CODING

There are a number of places where some slightly fancy coding would reduce the program length, but the objective was to avoid obscurity. The conditional addition of YL, for example, could have been achieved by −(M>2)*YL, rather than by an IF-THEN statement. Elsewhere, BA could have been replaced by MA(R), BB by MA(R+1), but the use of the copied variable seems to assist clarity. Those who value space saving more than program clarity can easily make the necessary changes.

Thanks to the rude comments of colleagues, most of the potential snags and errors seem to have been ironed out. There has been some argument regarding the date of introduction of the '400 year' rule, but if it was after 1600 AD only the first two months of 1600 are affected. Some similar programs have had much more blatant errors.

Extension of the program to cover biorhythms should be fairly simple, but at the age of 23055 days mine are fading too much to be of great significance!

```
100 DIM MA(13),MB(13),MC$(12)
110 FOR X = 1 TO 13
120 READ MA(X):READ MB(X)
130 NEXT X
140 FOR X = 0 TO 6:READ D$(X):NEXT X
150 FOR X = 1 TO 12:READ MC$(X):NEXT X
199 REM ................Main Menu
200 PRINTCHR$(12)
210 PRINT "MAKE A DATE":PRINT:PRINT
220 PRINT "To create a year calendar, type 'Y'."
230 PRINT "To create a month calendar, type 'M'."
240 PRINT "To calculate dates, type 'C'."
250 A$=""
260 INPUT A$
270 IF A$ = "C" THEN 400
280 IF A$ = "Y" THEN 2000
290 IF A$ = "M" THEN 3000
300 PRINT "Try again...":GOTO 250
399 REM ................Calculation Routine
400 P = 0:L = 0
410 INPUT ">";A$
420 IF A$="E" THEN 220
430 IF LEFT$(A$,1) = "+" THEN P=1
440 IF LEFT$(A$,1) = "-" THEN L=1
450 E$ = "":F = 0
460 FOR X = 1 TO LEN(A$):B$ = MID$(A$,X,1)
470 IF ASC(B$) >47 AND ASC(B$) <58 THEN E$=E$+B$
480 IF B$ <> "." THEN 520
490 F = F + 1
500 IF F=1 THEN D = VAL(E$):E$=""
510 IF F=2 THEN M = VAL(E$):E$=""
520 NEXT X
530 IF F = 0THEN N = VAL(E$):E$=""
540 IF F<>2 THEN 590
550 Y = VAL(E$):E$=""
560 GOSUB 1000
570 N = N+MB(M) + D
580 GOSUB 1100:IF M>2 THEN N=N+YL
590 IF P=1 THEN NA = NA + N:GOTO 620
600 IF L=1 THEN NA = NA - N:GOTO 620
610 NA = N
620 N = NA
630 IF F=2 AND L=1 THEN 670
640 GOSUB 1200
650 G = N+5 - 7*INT((N+5)/7)
660 PRINT TAB(20) D$(G) D;MC$(M);Y
670 IF (P=1 OR L=1) AND F<>0 THEN PRINT TAB(30) N
680 GOTO 400
999 REM ............. Calculate Year Base
1000 N=365*(Y-1600)
1010 N=N + INT(ABS(Y-1601)/4) - INT(ABS(Y-1601)/100)
1020 N=N + INT((Y-1201)/400)
1030 RETURN
1099 REM..............Identify Leap Years
1100 YL = 0
1110 IF (INT(Y/4) = Y/4) AND (INT(Y/100)<>Y/100) THEN YL=1
1120 IF INT(Y/400) = Y/400 THEN YL=1
1130 RETURN
1199 REM ........... Calculate Date
1200 Y = INT(N/365.25) + 1600:NH = N
1210 GOSUB 1000
1220 DF = NH - N
1230 GOSUB 1100
1240 IF DF > 365 + YL THEN Y = Y + 1:GOTO 1210
1250 IF DF < 1 THEN Y = Y - 1:GOTO 1210
1260 X=1
1270 DK = MB(X+1):IF YL = 1 AND X>1 THEN DK=DK+1
1280 IF DF > DK THEN X = X + 1: GOTO 1270
1290 M = X
1300 D = DF - MB(X):IF X>2 THEN D = D - YL
1310 N = NH
1320 RETURN
1999 REM ................Year Calendar
2000 INPUT "Year";Y:D=1:M=1
2010 PRINT CHR$(12):PRINT "CALENDAR FOR" Y
2020 FOR R = 1 TO 12 STEP 2
2030 PRINT:PRINT MC$(R);:PRINT TAB(30) MC$(R+1):PRINT
2040 PRINT "SUN MON TUE WED THU FRI SAT";
2050 PRINT TAB(30) "SUN MON TUE WED THU FRI SAT"
2060 GOSUB 1100
2070 BA = MA(R):BB = MA(R+1)
2080 IF R = 1 THEN BB = BB + YL
2090 GOSUB 1000: N = N + MB(R) +1
2100 IF R>1 THEN N=N+YL
2110 GA = N+5-7*INT((N+5)/7)
2120 GOSUB 1000: N = N + MB(R+1) +1
2130 IF R>1 THEN N=N+YL
2140 GB = N+5-7*INT((N+5)/7)
2150 W=1:Z=1
2160 IF W>BA AND Z>BB THEN 2290
2170 IF W>BA THEN 2270
2180 PRINT TAB(4*GA-(W<10)) W;
2190 W=W+1
2200 GA = GA+1:IF GA>6 THEN GA = 0:GOTO 2220
2210 GOTO 2160
2220 IF Z>BB THEN PRINT:GOTO 2160
2240 PRINT TAB(30+4*GB-(Z<10)) Z;
2250 Z=Z+1
2260 GB=GB+1:IF GB>6 THEN GB=0:PRINT:GOTO 2160
2270 IF Z>BB THEN PRINT:GOTO 2160
2280 GOTO 2220
2290 PRINT:NEXT R
2300 INPUT Q:GOTO 220
2999 REM ................Month Calendar
3000 INPUT "Year";Y
3010 INPUT "Month";M
3020 PRINT CHR$(12):PRINT:PRINT MC$(M);" "Y
3030 PRINT:PRINT "SUN MON TUE WED THU FRI SAT
3040 GOSUB 1100
3050 BA = MA(M):IF M=2 THEN BA = BA+YL
3060 GOSUB 1000
3070 N = N+MB(M) +1
3080 IF M>2 THEN N=N+YL
3090 GA = N+5 - 7*INT((N+5)/7)
3100 W=1
3110 IF W>BA THEN 3190
3120 PRINT TAB(4*GA-(W<10)) W;
3130 W=W+1
3140 GA = GA+1:IF GA > 6 THEN GA=0:PRINT
3150 GOTO 3110
3190 PRINT:PRINT:PRINT
3200 INPUT Q:GOTO 220
4000 DATA 31,0,28,31,31,59,30,90,31,120,30,151
4010 DATA 31,181,31,212,30,243,31,273,30,304,31,334,0,365
4020 DATA Sunday,Monday,Tuesday,Wednesday,Thursday
4025 DATA Friday,Saturday
4030 DATA January,February,March,April,May,June
4040 DATA July,August,September,October,November,December
READY
```

LUCAS MICRODEALER

LUCAS LX

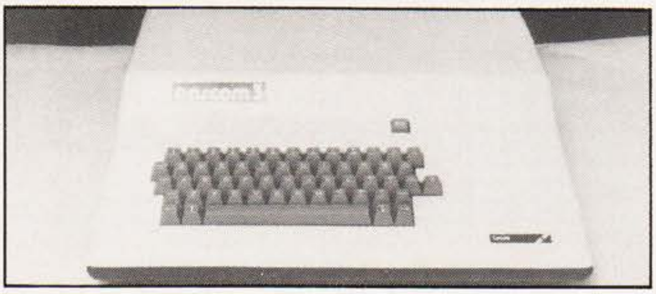
MEMORY	64K RAM expandable to 256K
LANGUAGE	Microsoft BASIC
CASSETTE	300 or 1200 baud
DISC	Single or twin 5¼ floppy disc drives DOS CP/M 2.2 (supplied) or NAS-DOS
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☑ FUNCT ☑
DISPLAY	TV ☑ MONITOR ☑ SUPPLIED ☑
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑ LINE ☐ RES 392 by 256 COLOUR 8 TEXT 80 by 25

Notes. The Lucas LX is a Z80 microcomputer aimed more at the professional and business user. Hence 5M Winchester disc interfacing is provided. Popular printers may be used with the RS232 serial interface, and a Centronics interface is also provided. There is an additional parallel interface connector for providing up to 16 on/off signals. The monitor supplied as standard is a 12" monochrome version: a colour monitor is also available. The high res colour graphics may be 392 by 256 in eight colours, or 784 by 256 in two colours. A wide range of applications software is available via the CP/M operating system, including Wordstar, Supercalc, and Calcstar.

NASCOM 3

MEMORY	48K RAM 10K ROM
LANGUAGE	Microsoft BASIC
CASSETTE	300 or 1200 baud
DISC	extra DOS CP/M or NAS-DOS
KEYBOARD	QWERTY ☑ CURSOR ☐ NUMERIC ☐ FUNCT ☐
DISPLAY	TV ☑ MONITOR ☑ SUPPLIED ☐
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑ LINE ☐ RES 800 by 256 COLOUR 8 TEXT 25 by 80
SOUND	Three channels

Notes. The Nascom 3 is a Z80 based micro. A second version of BASIC and Pascal are also available, as are a cassette recorder and light pen.

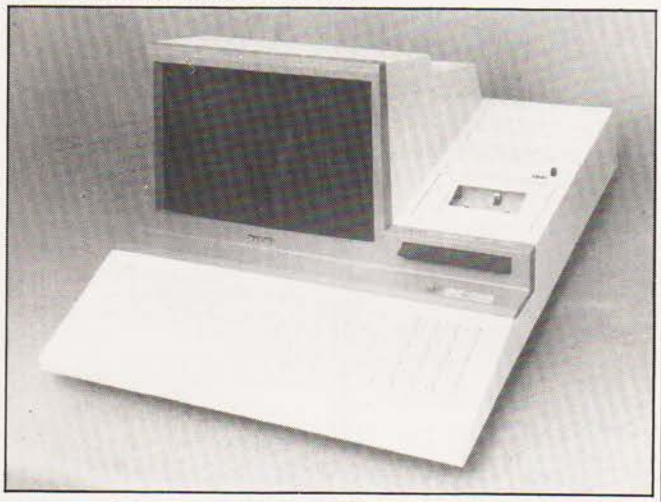


SHARP MICRODEALER

SHARP MZ-80A

MEMORY	48K RAM	4K ROM		
LANGUAGE	Microsoft BASIC			
CASSETTE	1200 baud (built-in)			
DISC	extra	DOS		
KEYBOARD	QWERTY☑	CURSOR☑	NUMERIC☑	FUNCT☐
DISPLAY	TV☐	MONITOR☑	SUPPLIED☑	
INTERFACE	PARA☑	SERIAL☐	BUS☑	
GRAPHICS	BLOCK☑	USER☐		
	LINE☐	RES 80 by 50		
	COLOUR	TEXT 25 by 40		
SOUND	Single channel			

Notes: The Sharp MZ-80A is a Z80 based micro. An expansion unit, printer, floppy disc unit and other peripherals are available. Other languages can also be used such as Pascal merely by replacing the tape. With the floppy disc option the machine can respond to higher level software such as Disc BASIC and FDOS (including BASIC compiler). A small range of business and educational software is available. The supplier is **Sharp Electronics (UK) Ltd.** Thorp Road, Newton Heath, Manchester M10 9BE.



SHARP MZ-80B

MEMORY	64K RAM	2K ROM		
LANGUAGE	BASIC (on tape)			
CASSETTE	1800 baud built-in			
DISC	extra	DOS		
KEYBOARD	QWERTY☑	CURSOR☑	NUMERIC☑	FUNCT☐
DISPLAY	TV☐	MONITOR☑	SUPPLIED☑	
INTERFACE	PARA☐	SERIAL☐	BUS☑	
GRAPHICS	BLOCK☑	USER☐		
	LINE☑	RES 320 by 200		
	COLOUR	TEXT 25 by 80		
SOUND	3 channels			

Notes: The Sharp MZ-80B is a Z80A based micro. Various other languages can be loaded as the machine is "soft", no language being fitted in ROM. Expansion unit, the MZ-80P5 printer and the MZ-80FB floppy disc drive are also available. The supplier is **Sharp Electronics (UK) Ltd.** Thorp Road, Newton Heath, Manchester.

COMPUTING TODAY

Lineage: 35p per word.
Semi display: £8.00 per single column centimetre
Ring for information on series bookings/discounts.

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

01-437 1002
EXT 305

Send your requirements to:
MARK BECAREVIC
ASP LTD, 145 Charing Cross Road, London WC2H 0EE.

ADD-ONS

NASCOM

COLOUR DISTEL Information Display Executive **£9.95 inc.**
Enhanced version controls colour via the PIO and simple interface. DISTEL is an advanced program for control of dynamic teletext type displays with simple keyboard commands for five fonts, pauses, speed changes, blanking, borders, special graphics, overlaying for animation, mimic diagrams and scrolling announcements. Paging system for replaying in any order. Ideal for advertising, lecturing or creative amusement.

ANIMEX Animation Executive **£7.95 inc.**
Fast machine code program for frame animation. Generate partial images with Edifis for rapid composition. Cartoon examples supplied on tape.

EDIFIS Editor for images **£9.95 inc.**
3K machine code image composition program for high res. characters of pixel images for DISTEL and ANIMEX. Comprehensive commands for translations, mirror, logical combinations, fill, multiple scale etc.

ALL PROGRAMS ON HIGH QUALITY TAPE WITH FULL HANDBOOKS

24HR VISA/ACCESS Service **Tel: 031-552-7501/2**

A.D.D. Ltd, 50 West Harbour Rd., Edinburgh EH5 1PU

ANIMATION GRAPHICS BOARD FOR NASBUS/Z80 BUS SYSTEMS
11½" x 8" PCB WITH NASBUS EDGE CONNECTOR CONTAINING:

• Video Display Processor TMS 9928A/29A • 16K screen RAM driven by the VDP chip • 256 x 192 pixal resolution in 16 colours • Backdrop, pattern plane and 32 'Sprite' planes • 32 'Sprites' 8 x 8, 16 x 16 or 32 x 32 pixal objects • Sprites magnified by bit in VDP registers • X,Y co-ordinates set by two bytes/sprite • 3D 'Eclipse' effect of sprite super-imposition • coincidence flag detects sprite collisions • 8 channel 8 bit A-D on board (joysticks etc.) • Twin sound generators for stereo sound • 2K Bytes C-Mos battery backed up RAM for non volatile storage • Choice of 2 real time clocks + calendar • CTC for sound generator programming and timing functions •
All functions mapped as i/o ports PCB, i/o decode prom, construction manual, 60 page software manual, tape with test and demonstration software all for £33.80 + £1.75 post and package. Colour difference Matrix Board for linear and TTL monitors PCB £8.80 + 50p post and package.

TANELORN SYSTEMS
Bank End, Micklethwaite, Bingley, West Yorkshire BD16 3JR 0274 568380

PET CONVERSIONS AND UPGRADES

It's here at new LOW price!!!
Don't throw out your old 9" PET.
The micro port 80 column conversion board for the 9" PET/CBM basic 4.0 or upgrade 2001/3000 series machines will upgrade it to a true 8000 series. Simply plugs into the main logic board, and with a small keyboard modification turns your PET into an 8032. Compatible with any 8032 software and ROM facilities. Supplied complete with full fitting instructions, or fitted free of charge at our workshops. Full money back guarantee within 14 days if not delighted **£125.00**
Externally mounted switching unit for above (40/80 column suitable) **£15.00**
12" VDU (FAT 40) 8 column conversion **£82.00**
40/80 column switchable (two machines in one)................ **£96.00**
Full keyboard functions i.e. TAB, ESC, REPEAT, SCROLL, up/down, DEFINE, WINDOW, LOWER CASE/GRAPHIC MODE AND DELETE ROM/2 'Cursor'. All available in direct or program mode. Any micro port converted machine is also upgradable to 8086 specification with our 64K add on board **£250.00**.
MEMORY UPGRADES WHILE-U-WAIT conversion (Dynamic RAM PETS only) Memory expansion: 8K-32K **£53.00**, 16K-32K from **£43.00**, 8K-16K **£34.40**.
CBM 64 QUALITY SOFTWARE
If you own a PET system with disk or printer and have added a Commodore 64, then why not use your PET as an interface for the 647 SIPOD is a IEEE to parallel IEEE interface allowing you to use all the PET's disk drives, printer, plotters, modems etc. directly from the 64. No memory space is consumed within the 64 as SIPOD loads and executes within the PET. SIDPOD is supplied complete with interface, lead and instructions **£29.00**
Prepaid orders are post free.
Telephone Mick Bignall for details for any of the above at: **Microport, 7 Clydesdale Close, Boreham, Herts WD6 2SD. Tel: 01-953 8385.**

CHILTERN ELECTRONICS
THE MONITOR SPECIALISTS
Amazing Value in Video Monitors
Professional quality equipment by AGC Corp ideal for your micro. Look at these features. ★ Superb resolution for graphics, 80 col. etc ★ Bandwidth 10hz-22Mhz ★ Green Screen P31 phorsphor ★ Antiglare tube ★ Attractively styled case ★ Composite video mains operation.
Available in 9" Model HM911 **£85.**
12" Model HM123 **£89.**
FANTASTIC KEYBOARD BARGAIN!!
Scoop purchase allows us to offer a 49 key QWERTY layout, ASCII encoded keyboard. Fully cased, used but guaranteed working. Full logic diagrams and information supplied. **ONLY £15.** Add carriage £1.85 and VAT. Educational discounts. Access welcome. Same day despatch. Call Gary Kent on 02407 71234. **High Street, Chalfont St Giles, Bucks HP8 4HQ. Telex: 262284.**

Microtan 65 Eprom Switching Board
Perm any 4 from 16. Software controlled. Bareboard + instructions **£19.95** post-free. Also available 64 x 25 colour VDU card, 3 pass assembler, word processor etc. Large SAE for details.
M.P.D., 7 Cedar Close, Grafham, Huntingdon PE18 0DZ.

SOFTWARE LANGUAGES

FORTH
AVAILABLE NOW

Nas-FORTH 1.11 for Nascom 2 etc.
Fig-FORTH with FORTH-79 additions. Complete with ZX80 assembler and screen editor, Ram-Disc simulation, stack display utility, full support of Nas-sys routines. Extensive manual full system 9K on tape £25 + VAT
PET FORTH 1.11 for 4000 and 8000 series.
Fig-FORTH with FORTH-79 additions complete with 6502 assembler and screen editor. Ram-Disc simulation for tape version. Stack display utility. Powerful IEEE control words. Extensive manual. Tape version £25.00 + VAT. Disc version including tutorial disc £75.00 + VAT
xFORTH for CPM systems.
FORTH-79 for all Z80 CPM systems. Extensive manual. All above features and more. £75.00 + VAT including tutorial disc.

Purchase of our products includes free membership of xFORTH Users Group, and FORTH tutorial disc or tape. Purchase price includes multiple copies. Full range of FORTH books available.
Write or telephone for more details: 0202 764724.
David Husband,
2 Gorleston Road, Branksome, Poole, Dorset BH12 1NW

GO FORTH & * ;
DIY FORTH KITS
Installation manual **£7.00**
How to do it + model + definition + editor
Source code listing for one processor **£7.00**
6502, 6800, 6809, 8080/Z80, 8086/8088, 9900, 1802, 68000, Z8000, VAX, Apple] [, LSI-11, Eclipse
JUPITER ACE — a very nice FORTH micro now with free Rampack........ **£78.00 + VAT**
FULL FORTHS ON CASSETTE Spectrum 48L w+ editor/colour/sound **£14.95**
BBC + editor/assembler/graphics toolkit **£25.00**
Dragon 32 + editor/graphics commands **£18.95**

MicroProcessor Engineering Ltd
21 Hanley Road, Shirley
Southampton SO1 5AP
Tel: 0703 775482

LIBRARIES

THE MZ80K/A SOFTWARE LIBRARY
Choose from 150 titles. These include exciting Arcade/Adventure games, and Educational/Business applications. Once only membership fee **£5.00**. Hire charges only **90p** per 14 day period. Plus your first 5 games FREE. Send now for free illustrated catalogue to:
The Yorkshire Software Library, 13 Park Top, Pudsey, Yorkshire LS28 6BY.

CARTRIDGE CITY will rent Atari 400 and 800 owners cartridges at low rates. Discounts on software purchases. Details from Cartridge City, 25 Gaitside Drive, Aberdeen AB1 7BH.

SERVICES

ELIMINATE FAULTY CASSETTES

DataClone is the first company in the UK established specifically for the duplication of data cassettes.

All other duplicating houses are audio orientated — only DataClone has a duplicating system designed from scratch purely to handle computer information.

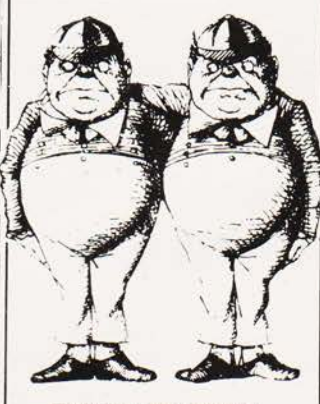
The result?

Greatly improved reliability in data transfer rates from 300 to beyond 2400 baud — previously unattainable.

All formats catered for. Quantities from 100 to infinity.

Contact us now for brochure.

DataClone — the first specialist service for computer cassettes.



DATACLONE
Unit 1, Roslin Square, Roslin Rd., Acton, London W3.
Tel: 01-723 0477 Telex: 21879

OFFERING A SERVICE?

PHONE ASP CLASSIFIED

01 - 437 1002
EXT 305

SOFTWARE APPLICATIONS

WHICH TAPE COPIER?
Only two can copy ALL Spectrum programs and use no memory area. One is £4.95, other is 007 Spy at £3.95, with free program to stop any machine code programs, and works with/without micro drive.
007 SPY £3.95
ZX GUARANTEED (DEPT ZX1), 29 CHADDERTON DRIVE, UNSWORTH, BURY, LANCS
Tel: 061-766-5712

NEWBRAIN SOFTWARE. 3D graphics allows the creation of moving 3D images. Machine code vector drawing integer maths and 3D transformation routines allow rapid simulated motion through scenes, image rotation etc. Mainframe-like images on a micro for £9.95. Word Processor contains extensive machine code for high speed document reformatting etc. Save documents on cassette £8.95. Renumber handles goto, gosub etc. £4.95. All programs supplied as easily loaded listings to eliminate tape problems. Comprehensive user guide. Cheques to S. Gaymer, 16/18 Princes Street, Ipswich.

BBC copy protected Acornsoft tapes with master key. No more "locked" messages. Unprotected copy can be copied to disc if required. Works on ST*RSH*P C*MM*ND, SN**K*R, etc. Full instructions. Tape £5.00. To: Nicholas Benton, 1 Cow Lane, Steeple, Aston, Oxon.

HARDWARE

NEWBRAIN
The professional micro computer for the price of the hobby machine. Delivery free. Call us for very special offer. Mail Order. Access.
Call **Stevenage (0438 812439)**
Anytime for the latest details of Newbrain Hardware, Software, Books. "Newbrain Dissected" Book **£9.00**, tape **£9.95** and both **£18.45**. "Getting More from your Newbrain" **£7.50**. All include P&P. New Epson, CP80 and Juki printers.
ANGELA ENTERPRISES
4 Ninnings Lane, Rabley Heath, Welwn, Herts AL6 9TD.

FUTURE TECHNOLOGIES
(PCT) 1 Hollybrook Rd, Shirley, Southampton, SO1 6RB
(Reg. Office. Callers By Appointment)
CBM 64K Computer £198
With Cassette & 2 Blank Tapes £242
VIC 20 Starter Kit £139
Cassette Alone — £44.90 — Tapes £1 each
Spectrum 16K — £99.95, 48K — £129.95
Also Atari, Texas, Accessories, Software
Cheque/P.O. with Order
Send large SAE for Lists (23p Stamp)
Add £1 p&p for Orders under £50
All prices include VAT

ASP CLASSIFIED
01 - 437 1002
EXT 305

SOFTWARE GAMES

GROVE SOFTWARE
Games for the Jupiter Ace with 16K
TBT 1 Mastermind + Bomber
TBT 2 Pairs + Submarines
£5.50 each inc p&p
SAE for Catalogue
Cheques and PO's to:
GROVE SOFTWARE
2 Grove Court, Oakfield Road, Penge, London SE20 8RG.

NASCOM ACTION? Get it with Revenge of Drosophila! Fast Synchronised Animation, Scrolling Maze, 5 different Maze's to survive and 8 Game Variations. Requires 16K 1, 2 or 3 with NASGRA ROM (Ver. 3). Monitor Independent - Free Loader. Specify CUTS or NI format. £8 post free from: Garry Rowland, 24 Parsloes Avenue, Dagenham RM9 5NX.

UK 101 SOFTWARE — Frogger, Dominoes, Boston, Yahtzi, Connect 4, Lincs, Simon, Fireman, PF21, Pairs, Defuse. Other programs from 4K to 22K, colour or black/white with sounds. Send for details to Dr Gilchrist, 70 Skirbeck Road, Boston, Lincs PE21 6DG.

MICROTAN, Tanex, cased keyboard, 19" rack, power supply, X-bug. Cost £350. Offers over £150. Malvern 68865.

FOR SALE

TRS80 LEVEL 2 48K £160. Acculab floppy tape system £80, 1/2/4K Eprom programmer £20, 81 Centronics Microprinter £50. All together £280. Tel: (027 74) 56145 after 6.15 pm.

TRS80 LEVEL 2 16K. Bounce — addictive arcade favourite £4.50. Drag Race — superb sound and graphics £4.50. Cassetee, P&P included, or SAE for lists. R. Bowden, 6 Briarley Gardens, Woodley, Stockport SK6 1PJ.

ACORN ATOM 12+12K floating point manual leads games inc 747. £130 ono. Telephone after 7pm 01-274 8464 (Stockwell).

TRS80 LEVEL 2 48K in keyboard with line printer 7. Complete with plenty of books and software. £275 or ono. Tel: (0908) 606256.

NASCOM-2 cased, 48K RAM-B, NAS-SYS 1, NASGRA, NASPEN, toolkit, Imp printer, Imprint, spare ribbon, soundbox, Castle interface, tape recorder, associated manuals. £500. Woking 72739 evenings.

TRS-80 V.G. extended cassette-based BDFORTH. 79-standard with over 170 commands. Only £12.50 including manual. T. Dorricott, 6 Gunton St., Peter's Ave., Lowestoft, Suffolk.

ACCESSORIES

SPECTRUM TAPE ACCESSORIES
HIGH-SPEED LOADER: load and save at 3000 baud on a normal recorder with our short Speedyload program. Easy to use with any command. Cassette (48K), £3.95.
AUTOMATIC TAPE CONTROL: plugs in Spectrum cassette sockets for auto ON/OFF and LOAD/SAVE switching of one or two recorders. Operated by BASIC program or keyboard commands. Built-in BEEP amplifier with volume control, plus semi-auto switch for rewind, etc. Black case. £19.95 + £1.50 P&P. (£16.95 as kit).
BEEP AMPLIFIER: plugs in MIC socket, volume control, in black case, £5.95 + 85p P&P. Also available with recorder ON/OFF and LOAD/SAVE switches. £9.95 + £1.
TAPE CARE: cleaning cassette with built-in head de-magnetiser, and cleaning fluid, £1.95 + 50p P&P.
VIDEO EXTENSION LEAD: 6ft. TV plug to socket, £1.85.
AERIAL SWITCH: saves unplugging TV, plug-in, £2.45.
MAINS LEAD: 9ft, plug to 2 sockets, £3.85 + P&P. **£1**
CASSETTES: C5-46p, C12-48p, C60-36p. 10% off for 10.
CASSETTE LABELS: blank, adhesive, 50-£1, 100-£1.90. Min. P&P 30p + 10p per item. Send SAE for details.
NESS MICRO SYSTEMS
100 Drakies Avenue, Inverness IV2 3SD

XMAS GIFTS

Micro-Users SOFTWARE T-SHIRTS and SWEATSHIRTS

1 Apple	6 I'm User Friendly
2 Spectrum	7 Have you seen my Peripherals?
3 Oric 1	8 Z80A
4 Epson	9 6502
5 Invader	

100% Cotton T-shirts in white, red or sky £3.25 each inclusive. Poly-cotton Sweatshirts in white or grey £5.50 each inclusive, small, medium, large and extra large sizes. Mail order only from:
One Per Cent Screens, Unit 12, Star Lane Estate, Great Wakering, Essex.
Xmas orders send first class return post.

ADD-ONS

PRINT LINK 64
Interfaces a 64 directly to any printer with centronics input. The printer will respond to all the normal basic 4 commands. Supplied complete with instructions and interface cable ... **£26**
We only sell one wordprocessor for the 64. We think it's the best!!!
Vizawrite 64. Text formatting word-processors disk or cartridge **£69.00**
Vizaspell 64. Spelling checker with 30,000 word dictionary **£50.00**
Vizawrite + Vizaspell. Combined package (disk only) **£99.00**
Mikro Assembler. Write machine code with the ease of Basic **£50.00**
Arrow. Save and load to tape up to 7 times faster than normal **£39.00**
Victree. All toolkit and basic 4.0 commands + a lot more **£49.00**
Zoom. An excellent machine code monitor for the 64 .... **£10.00**
Graphix 64. Easy to use high resolution graphics from basic **£10.00**
Stix. A most addictive and compelling game. (Joysticks required) **£17.35**

Official orders from schools, universities, major companies etc are welcome. UK customers please add 15% VAT.

MICROPORT
7 Clydesdale Close, Boreham, Herts WD6 2SD.
Tel: 01 - 953 8385

CLASSIFIED ADVERTISEMENT — ORDER FORM

1.	2.	3.
4.	5.	6.
7.	8.	9.
10.	11.	12.
13.	14.	15.

When placing your ad, please state classification required. 35p per word.
Send to: ASP Classified, 145 Charing Cross Road, London WC2.
Tel: 01-437 1002
Name ....
Address ....
....
....
Tel.No.(Day) ....

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

ADVERTISERS INDEX

DRAGON 32 OWNERS

DELTA DISK SYSTEM

Make your Dragon turn into a real computer with the new Double-Density Delta Disk System.

The Delta Disk System Gives You . . .

- An affordable disk system.
- Powerful Delta disk commands.
- Lets you produce and handle random access files as easily as serial files.
- Random sequential and indexed file handling.
- Simple plug into Dragon there are no HARDWARE MODS needed to run DELTA!
- Easily expandable 180K to 1.4 megabyte ON LINE storage.
- Full range of business utility and games software AVAILABLE NOW!
- The price you see is the price you pay. NO HIDDEN RAM upgrade costs.
- Uses under 2K of user – RAM as DELTA is held in EPROM.
- Enables programmer to easily produce applications Software which automatically starts up and operates without any intervention from the user.

FULL RANGE OF BUSINESS SOFTWARE AVAILABLE. SEND SAE FOR DETAILS.

DELTA CARTRIDGE – contains DELTA disk Operating System, User Manual, demonstration diskette	£120.00
DELTA 1 – DELTA Cartridge, User Manual, a single-sided 40-track (180K) drive plus cable	£320.00
DELTA 2 – as DELTA 1, but with a single-sided 80-track (360K) drive	£355.00
Disk Interface cable (supplied with DELTA 1 or 2)	£9.95
ENCODER 09 assembler/disassembler/editor – integral with DELTA	£24.95
HOME ACCOUNTS full home package for DELTA	£14.95
INFORM – Data Base Management System commissioned especially for DELTA Systems	£39.95

NEW

TOOLKIT FOR DRAGON 32

PREMIER'S ASTONISHING NEW TOOLKIT FOR THE DRAGON 32 PERFORMS THE FOLLOWING AMAZING FUNCTIONS:–

- FULL screen editor allowing copying, concatenating of lines etc. Fully linked to the DRAGON'S line editor.
- Eight PROGRAMMABLE KEYS – all easily accessed or checked.
- 25 full colour low-resolution graphics screens available with commands for instantly writing to any of them or swopping/moving. MAKES COLOUR ANIMATION AT MACHINE CODE SPEED a simple task.
- Full range of ERROR HANDLING commands with several RESUME options.
- OVER SIXTY NEW WORDS fully linked to your DRAGON BASIC – your new commands become built-in as soon as you plug in the TOOLKIT cartridge.
- VARIABLE GOTO/GOSUB commands, allowing routines to be called by name. Eg GOSUB TITLE or GOTO FINISH.
- Superb TRACE command which uses top right of screen only, thus leaving graphics/text intact.
- SEARCH and REPLACE commands for easier program modification.
- Comes complete with COMPREHENSIVE 40+ page MANUAL.

Available in CARTRIDGE or DELTA enhancement format .£29.95inc.

NEW

● DRAGON Fig FORTH ON DISK FOR DELTA ● Executes up to 10 times faster than BASIC ● Language extendable by user defined words ● Words can even be defined using the FORTH assembler for maximum speed ● BASIC and DELTA commands still available from FORTH ● Source code stored and can be compiled from disk ● FORTH can be saved to disk, including any new definitions ONLY £29.95 including MANUAL + Fig FORTH glossary + Model. Inc. VAT

SCRIBE FOR THE DRAGON 32

- FULL UPPER and LOWER CASE direct from the keyboard.
- An enlarged 42 x 24 screen display which gives a superb READABLE text.
- Full text display on hi-res screen.
- Automatic underline and reverse image.
- Up to 255 user-defined graphics.
- Precision character position command giving super/subscripts.
- Black on white text display option.
- A new Print @ command extending to 1008 thus almost doubling screen area.

ENCODER 09 FOR THE DRAGON 32

ENCODER 09 – is a full symbolic assembler using standard mnemonics and pseudo op-codes. Source code can be incorporated into BASIC programs. The monitor section contains commands to allow memory display, modification and execution. Memory block move, breakpoint handling, full disassembly and a full editor are only a few of its many features. The most powerful assembler/-disassembler/editor available for the DRAGON 32. Available as either an integral DELTA fitment or in cartridge.
Cartridge £29.95. Disk – see above.

HIPRINT FOR THE DRAGON 32

HIPRINT – screen dumper.

– will dump the entire contents of your DRAGON 32 high-res screen to a high resolution printer. Can be used for design, display etc (see left). Available at present for EPSON printers only. Other modules to follow shortly.
Tape £7.95. DELTA disk £9.95.

PREMIER MICROSYSTEMS

PREMIER MICROSYSTEMS
208 Croydon Road, Anerley, London SE20 7YX
Telephone 01-659 7131 or 778 1706
Dealer and export orders welcomed.

POSTAGE and PACKING
Tapes 1–2 95p, 3+ £1.25.
Cartridges £2.50
Drives or Disk packages £4.50
ALL PRICES INCLUDE VAT.
Send SAE for full catalogue.
ALLOW 28 DAYS FOR DELIVERY.

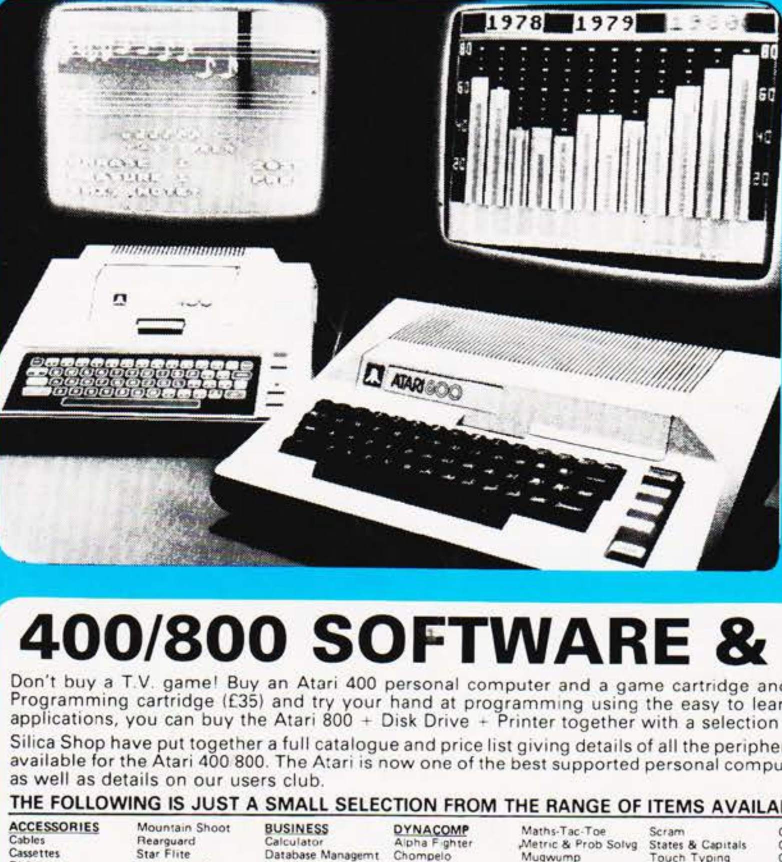
Access BARCLAYCARD VISA

100 FREE PROGRAMS

FROM SILICA SHOP — WITH EVERY PURCHASE OF AN

ATARI 400 800

ATARI



ATARI PRICES REDUCED!

We at Silica Shop are pleased to announce some fantastic reductions in the prices of the Atari 400/800 personal computers. We believe that the Atari at its new price will become the U.K.'s most popular personal computer and have therefore set up the Silica Atari Users Club. This club already has a library of over 500 programs and with your purchase of a 400 or 800 computer we will give you the first 100 free of charge. There are also over 350 professionally written games and utility programs, some are listed below. Complete the reply coupon and we'll send you full details. Alternatively give us a ring on 01-301 1111 or 01-309 1111.

ATARI 400 with 16K	**£99**
ATARI 400 with 48K	**£158**
ATARI 800 with 48K	**£249**

400/800 SOFTWARE & PERIPHERALS

Don't buy a T.V. game! Buy an Atari 400 personal computer and a game cartridge and that's all you'll need. Later on you can buy the Basic Programming cartridge (£35) and try your hand at programming using the easy to learn BASIC language. Or if you are interested in business applications, you can buy the Atari 800 + Disk Drive + Printer together with a selection of business packages.

Silica Shop have put together a full catalogue and price list giving details of all the peripherals as well as the extensive range of software that is now available for the Atari 400/800. The Atari is now one of the best supported personal computers. Send NOW for Silica Shop's catalogue and price list as well as details on our users club.

THE FOLLOWING IS JUST A SMALL SELECTION FROM THE RANGE OF ITEMS AVAILABLE:

ACCESSORIES
Cables
Cassettes
Diskettes
Joysticks
Le Stick - Joystick
Misc Supplies
Paddles

ADVENTURE INT
Scott Adams Adv
No 1 Adventureland
No 2 Pirate Adv
No 3 Mission Imp
No 4 Voodoo Cast
No 5 The Count
No 6 Strange Ody
No 7 Mystery Fun
No 8 Pyramid of D
No 9 Ghost Town
No 10 Sav Island 1
No 11 Sav Island 2
No 12 Golden Voy
Angle Worms
Deflections
Galactic Empire
Galactic Trader
Lunar Lander
Mountain Shoot
Rearguard
Star Flite
Sunday Golf

AUTOMATED SIMULATIONS
Crush Crumble Cmp
Datestones of Ryn
Dragons Eye
Invasion Orion
Rescue at Rigel
Ricochet
Star Warrior
Temple of Apshai
Upper Reaches Aps

BOOKS
Basic Ref Manual
Compute Atari DOS
Compute Bk Atari
Compute Magazine
De Re Atari
DOS Utilities List
DOS2 Manual
Misc Atari Books
Op System Listing
Wiley Manual

BUSINESS
Calculator
Database Managemt
Decision Maker
Graph-It
Invoicing
Librarian
Mort & Loan Anal
Nominal Ledger
Payroll
Personal Finl Mgmt
Purchase Ledger
Sales Ledger
Statistics 1
Stock Control
Telelink 1
Visicalc
Weekly Planner
Word Processor

CRYSTALWARE
Beneath The Pyram
Fantasyland 2041
Galactic Quest
House Of Usher
Sands Of Mars
Waterloo
World War III

DYNACOMP
Alpha Fighter
Chompelo
Crystals
Forest Fire
Intruder Alert
Monarch
Moonprobe
Moving Maze
Nominoes Jigsaw
Rings of The Emp
Space Tilt
Space Trap
Stud Poker
Triple Blockade

EDUCATION from APX
Algicalc
Atlas of Canada
Cubbyholes
Elementary Biology
Frogmaster
Hickory Dickory
Inst Comptg Dem
Lemonade
Letterman
Mapware
Maths-Tac-Toe
Metric & Prob Solvg
Mugwump
Music Terms/Notatn
Musical Computer
My First Alphabet
Number Blast
Polycalc
Presidents Of U.S.
Quiz Master
Starware
Stereo 3D Graphics
Three R Math Sys
Video Math Flash
Wordmaker

EDUCATION from ATARI
Conv French
Conv German
Conv Italian
Conv Spanish
Energy Czar
European C & Caps
Hangman
Invit To Prog 1/2/3
Kingdom
Music Composer
Scram
States & Capitals
Touch Typing

EMI SOFTWARE
British Heritage
Cribbage/Dominoes
Darts
European Scene Jig
Hickory Dickory
Humpty Dumpty
Jumbo Jet Lander
Snooker & Billiards
Submarine Commdr
Super Cubes & Tilt
Tournament Pool

ENTERTAINMENT from APX
Alien Egg
Anthill
Attank
Avalanche
Babel
Blackjack Casino
Block Buster
Block 'Em
Bumper Pool
Castle
Centurion
Checker King
Chinese Puzzle
Codecracker
Comedy Diskette
Dice Poker
Dog Daze
Domination
Downhill
Eastern Front
Galahad & Holy Grl
Graphics/Sound
Jax-O
Jukebox
Lookahead
Memory Match
Midas Touch
Minotaur
Outlaw/Howitzer
Preschool Games
Pro Bowling
Pushover
Rabbotz
Reversi II
Salmon Run
747 Landing Simul
Seven Card Stud
Sleazy Adventure
Solitaire
Space Chase
Space Trek
Sultans Palace
Tact Trek
Terry
Wizards Gold
Wizards Revenge

ENTERTAINMENT from ATARI
Asteroids
Basketball
Blackjack
Centipede
Chess
Entertainment Kit
Missile Command
Pac Man
Space Invaders
Star Raiders
Super Breakout
Video Easel

ON LINE SYSTEMS
Crossfire
Frogger
Jawbreaker
Mission Asteroid
Mouskattack
Threshold
Ulysses/Golden Fl
Wizard & Princess

PERIPHERALS
Centronics Printers
Disk Drive
Epsom Printers
Program Recorder
RS232 Interface
Thermal Printer
16K Memory RAM
32K Memory RAM

PERSONAL INT from APX
Adv Music System
Banner Generator
Blackjack Tutor
Going To The Dogs
Keyboard Organ
Morse Code Tutor
Personal Fitness Prg
Player Piano
Sketchpad

PROGRAMMING AIDS from Atari
Assembler Editor
Dsembler (APX)
Microsoft Basic
Pascal (APX)
Pilot (Consumer)
Pilot (Educator)
Programming Kit

SANTA CRUZ
Basics of Animation
Bobs Business
Display Lists
Graphics Machine
Kids 1 & 2
Horizontal Scrolling
Master Memory Map
Mini Word Processor
Page Flipping
Player Missile Gr
Player Piano
Sounds
Vertical Scrolling

SILICA CLUB
Over 500 programs write for details

FOR FREE BROCHURES — TEL: 01-301 1111

FREE LITERATURE

I am interested in purchasing an Atari 400/800 computer and would like to receive copies of your brochure and test reports as well as your price list covering all of the available Hardware and Software.

Name ....

Address ....

....

....

....

Postcode ....

CT184 —Computing Today

SILICA SHOP

For free brochure and reviews on our range of electronic products please telephone 01-301 1111. To order by telephone just quote your name, address, credit card number and order requirements and leave the rest to us. Post and packing is FREE OF CHARGE in the UK. Express 24 hour delivery available at an additional charge.

- SHOP DEMONSTRATION FACILITIES – we provide full facilities at our shop in Sidcup Monday to Saturday 9am to 5.30pm (closing Thursday 1pm, Friday 8pm).
- MAIL ORDER – we are a specialist mail order company and are able to supply goods direct to your door.
- MONEY BACK UNDERTAKING – if you are totally unsatisfied with your purchase, you may return it to us within 15 days. On receipt of the goods in satisfactory condition we will give you a full refund.
- PART EXCHANGE SECOND HAND MACHINES – we offer a part exchange scheme to trade in many makes of T.V. games for personal computers.
- COMPETITIVE PRICES – our prices, offers and service are very competitive. We are never knowingly undersold, and will normally match any lower price quoted by our competitors.
- HELPFUL ADVICE – available on the suitability of various computers.
- AFTER SALES SERVICE – available on all computers out of guarantee.
- VAT – all prices quoted above include VAT at 15%.
- CREDIT FACILITIES – we offer credit over 12, 24 or 36 months, please ask for details.

SILICA SHOP LIMITED
Dept CT184 1-4 The Mews, Hatherley Road, Sidcup, Kent DA14 4DX Telephone 01.301 1111 or 01-309 1111