An Argus Specialist Publication

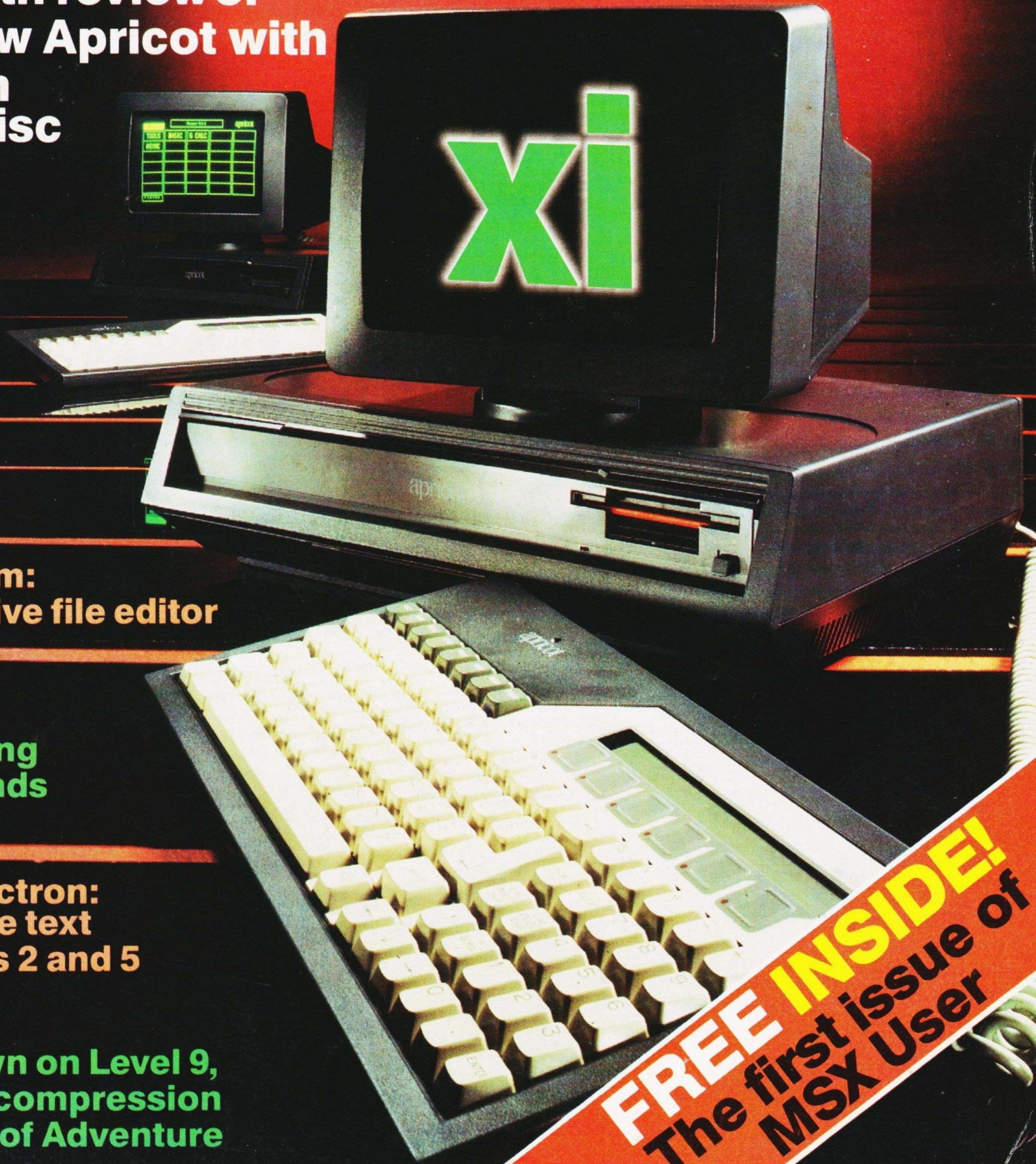
Computing today

90p
AUGUST
1984

INCORPORATING MICRO COMPUTER PRINTOUT incorporating BUSINESS MICRO

THE EXCELLENT xi

In-depth review of the new Apricot with built-in hard disc



Spectrum: Microdrive file editor

CBM 64: New string commands

BBC/Electron: Readable text in Modes 2 and 5

Low-down on Level 9, the text-compression masters of Adventure

FREE INSIDE! The first issue of MSX User

A maxim which eloquently describes the Brother HR-5.

Less than a foot across, it's nonetheless loaded with features.

The little printer that's low on decibels.

There's one thing the HR-5 won't give you.

Earache.

For the annoying 'clickety clack' many printers produce is mercifully absent from the HR-5.

Quietly efficient, it delivers high definition dot matrix text over 80 columns at 30 characters per second (maximum).

Text or graphics with ease.

The HR-5 also has something of an artistic bent.

Being capable of producing uni-directional graphics and chart images together with bi-directional text. What's more it will hone down characters into a condensed face, or extend them for added emphasis.

At home with home computers.

Incorporating either a Centronics parallel or RS-232C interface, the HR-5 is compatible with BBC, Spectrum, Oric, Dragon, Atari and most other home computers and popular software.

Perfectly portable, the battery or mains operated HR-5 weighs less than 4 lbs, and has a starting price of only £159.95 (inc. VAT).

Which is really something to shout about.

PLEASE SEND ME MORE DETAILS OF THE REMARKABLE BROTHER HR-5 PRINTER.

NAME____________________________

ADDRESS____________________________

____

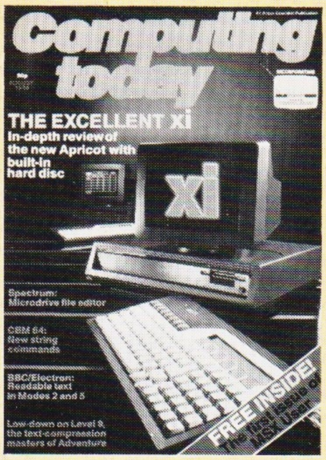
____

______________ TEL NO.______________

AVAILABLE FROM: BOOTS, RYMANS, WILDINGS, SELFRIDGES AND ALL GOOD COMPUTER EQUIPMENT STOCKISTS.

brother®

DEPT P, BROTHER OFFICE EQUIPMENT DIVISION, JONES + BROTHER, SHEPLEY STREET, GUIDE BRIDGE, AUDENSHAW, MANCHESTER M34 5JD
TEL:061 330 6531 (10 LINES) 061 330 0111 (6 LINES) 061 330 3036 (4 LINES). TELEX: 669092. BROTHER INDUSTRIES LTD., NAGOYA, JAPAN.



CONTENTS

VOL 6 NO 6 AUGUST 1984

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

Editor:
Peter Green

Editorial Assistant:
Fiona Eldridge

Advertisement Manager:
Jeremy Ornellas

Advertisement Copy Control:
Sue Couchman,
Ann McDermott

Managing Editor:
Ron Harris BSc

Chief Executive:
T.J. Connell

Origination by Design International.

ABC Member of the Audit Bureau of Circulation

ISSN 0142-7210

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

Subscription rates: UK £13.30 including postage. Airmail and other rates upon application to Computing Today Subscriptions Department, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP1 1BB, England.

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 Golden Square address.

NEWS

EXPANDING THE ELECTRON

Printers, joysticks and ROM software cartridges can now be plugged into the Acorn Electron through the PLUS-1, a £59.90 expansion unit. In a matching compact unit securely fixed to the host Electron, the PLUS-1 adds a Centronics-compatible printer interface, a joystick (analogue) port, and two slots for Acornsoft's new ROM cartridge software.

Cartridge software means instant plug-in loading without any need to connect a cassette recorder. And the cartridge slots also open the door to future hardware expansions, including an RS423 serial interface for serial printers, modems and other computers.

Acornsoft has announced the first six ROM cartridges for the Electron/PLUS-1 system: four games — Snapper, Starship Command, Hopper and Countdown to Doom, the educational Tree of Knowledge and the artificial intelligence programming language LISP. Cartridges cost £14.95 each, including VAT except LISP which cost £39.95, and includes a user guide and demonstration tape.

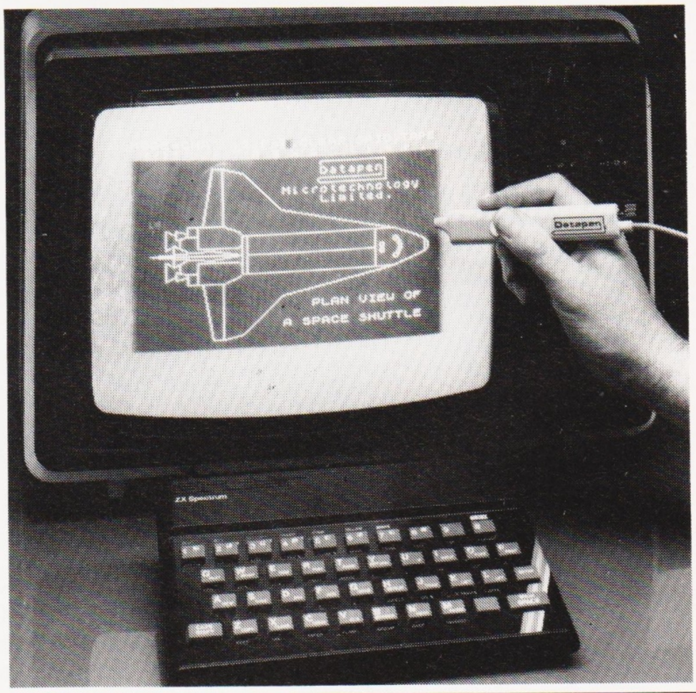
The PLUS-1 come complete with user guide, which includes connection and operating details and tips on how to write programs to use its facilities. It costs £59.90 including VAT and will be available from Acorn Electron stockists and mail order from Vector Marketing, London Road, Denington Estate, Wellingborough, North Hants NN8 2RL.

A PEN FOR YOUR THOUGHTS

Datapen Microtechnology Ltd has just announced details of its ZX Spectrum compatible lightpen and programs. Datapen have followed their usual pattern of providing informative software and literature with the lightpen, but for the ZX Spectrum they have included two programs in addition to the introduction program. One is a user-defined character geenerator and the other is a high-resolution, full colour drawing program capable of producing pictures to a pixel accuracy and incorporating the automatic drawing of geometric shapes, such as circles, triangles and rectangles.

Datapen say that their lightpen out-performs other lightpen because of the micro miniature circuitry built into the pen body. The lightpen is insensitive to local lighting conditions as it contains an electronic filter so that the pen only responds to high frequency light from the TV raster. The pen has a red LED indicator, which lights whenever valid video data is available and the signal from this is available to the computer. Another feature of this lightpen is a switch which allows the computer to ignore any signals that come from the lightpen before you are ready and on the correct place on the screen.

The ZX Spectrum version is available now both from shops and direct from Datapen. It costs £29 inclusive, complete with all three programs and handbook. For further details of the above, or their range of lightpens for different computers, contact Datapen Microtechnology Ltd, Kingsclere Road, Overton, Hants RG25 3JB (telephone 0256 770488).



SOLARIS DAWNS

European users of corporate data processing terminals can now enjoy the flexibility of local processing functions with a new low-cost system that emulates the IBM Personal Computer. TDI Limited, the Bristol-based company who pioneered the first 6800 into the UK in the shape of the Sage microcomputer, and who are the main distributor of the P-SYSTEM Universal Operating System, have launched a new TDI company, TDI Workstations Ltd, based in London, to market the Solaris Personal Computer Emulator (PCE).

With no new terminal hardware, software, or communications purchases needed, the TDI Solaris PCE upgrades most asynchronous video display terminals into IBM PC hardware and software equivalents without disrupting any element of the existing terminals. The PCE micro-to-mainframe link addresses the immediate needs of both corporate data processing managers and the individual users by enhancing a mainframe's existing terminals. Initially the TDI Solaris upgrade is being offered for the most widely used terminal, the DEC VT100, as well as VT100

NEW DIRECTOR FOR OSBORNE

Osborne have added the award-winning 'Financial Director' software to the list of business software supplied with the Osborne Executive personal computer, making it a full specification machine for just £1595 (excluding VAT).

The Osborne Executive is a portable personal computer, with a valuable range of leading-brand business system software, which includes — for word processing — Wordstar with Mailmerge, Supercalc spreadsheet, the Personal Pearl database system and Financial Director cash book and management accounts system. Also included are industry standard operating systems and programming languages.

Mike Healy, Managing Director of Future Management (Portable Computers) Ltd, sole UK distributors of the Osborne range says: "With the addition of Financial Director software on the Osborne Executive, I believe we offer unquestionably the best value for money professional business system available. With an inclusive hardware and software price for the Osborne Executives of £1595, we challenge anyone to beat this specification.

"As is evidenced in many of the complaints upheld by the Advertising Standards Authority, we are disturbed that consumers are being misled over the presentation of the true cost involved to achieve an effective business system. In some instances, the true cost of a full business system is three or even four times the basic price advertised as the cost of entry."

The business software packages offered with the Osborne Executive are valued at £1500 (recommended retail price) by Mr. Healy (but I bet he wouldn't sell you an Executive for £95 without the software! — Ed).

Future Management are at 38 Tanners Drive, Blakelands North, Milton Keynes, MK14 5LL (phone 0908 615274: telex 825220).

CLOSE SESAME

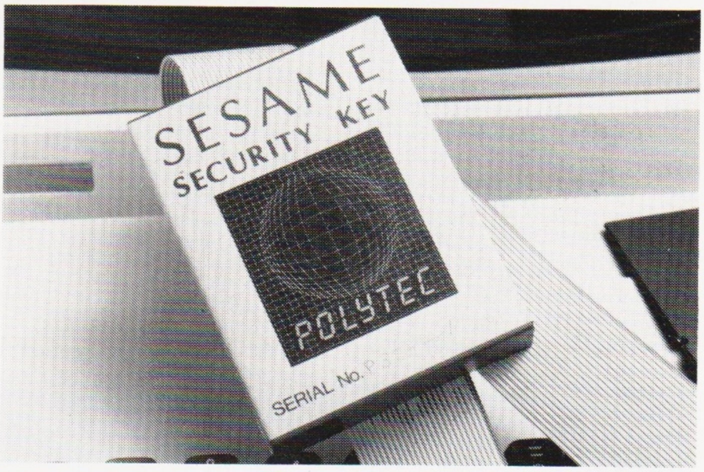
The Sesame Security key is a simple-to-use software protection system which provides a high level of software security. It is interrogated by the software and responds only when a unique code is passed through it, thus ensuring that direct copies of the program will not run on any computer which does not have the correct device connected.

Designed by a Cambridgeshire firm, FT Microsystems and marketed by Polytec Engineering, the device is useable with any computer having an RS232 port, without inhibiting the normal functions of the port in any way. All 25 lines of its D type connectors pass through the device unaffected. It needs no external power and requires that only Transmit data, Receive data and signal ground be supported.

The software must interrogate the device by passing its own unique code through it. The code will activate the device which will respond only if the correct code is sent. The codes, of which there are approximately 100 million variations, are ASCII control characters which would not normally affect any other device using the port. Due to the nature of the device even an infinitely fast computer would take around 20 years to test all the possible combinations, thus making attempts to crack the device totally unecomonic.

Each device is supplied with a randomly selected code, together with notes on use and a flow chart of the necessary interrogation procedures. Identical codes can be supplied for multiple orders. The programmer can design his interrogation routine in any language he chooses that allows him to access to the port in the normal way. He should use his ingenuity to disguise the interrogation procedures within the program he wishes to protect and thus make it very difficult for anyone to crack his program. To this end no standard software is supplied with the unit, only the necessary information to write it.

For more details contact Polytec Engineering Services Ltd, Unit 8, Nuffield Close, Trinity Hall Farm Industrial Estate, Cambridge CB4 1SS (phone 0223 312562).



emulating terminals.

The link between the Solaris PCE and its host terminal is simple to understand and to complete. Only a screwdriver is needed to make the connection between the terminal and the mainframe.

The Solaris PCE is expecially important for data processing and department managers who need local processing capabilities from their mainframes. These managers recognise that most of the installed data processing systems are by design efficient but inflexible.

The installation of a Solaris PCE preserves a company's existing investment, terminals, software and training. It provides complete personal computing capabilities for those people who need it.

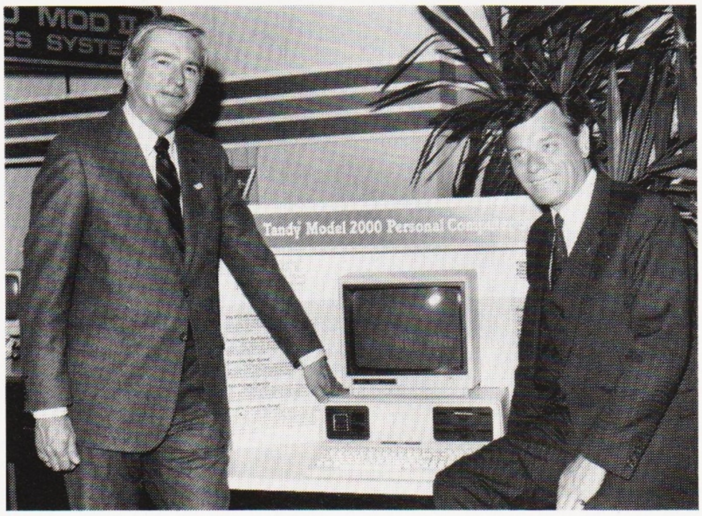
An important productivity-enhancing feature of the system is that the screen can be split to show, simultaneously, information from both the mainframe and the PCE. A 40-position auxiliary keypad allows direct generation of IBM PC specific control codes. Part of Solaris' plans for upgrade of the IBM PC system include protocol conversion, gateways and file transfer. The central resource manager will perform in a true distributed processing mode with the corporate network.

The PCE contains an INTEL 8088 processing unit (CPU) and runs 16-bit MS/DOS and other operating systems used on the IBM PC. It utilises 128K of random-access memory (RAM) that is expandable to 640K and comes with 5¼" (360K) flexible discs. The circuit boards are IBM PC-compatible which ensures low-cost reliability, and guarantees compatibility with any expansion boards used in the IBM PC.

Provided with the PCE is an auxiliary keypad allowing direct generation of control codes that would otherwise take a number of keystrokes to accomplish. The PCE has a battery back-up, so that no data is lost in the event of a power failure, and the 10M of Winchester disc storage which is available as an option is fully compatible with the IBM XT, and fits within the unit housing. The system is priced at £2795 in the UK, which includes a full 12 months on-site warranty. Upgrades to provide local-area networking and distributed data processing will be available in 1985. TDI Workstations are at 29 Buckingham Gate, London SW1 6NF (phone 01-826 6047).

TRASHMAN OF THE YEAR

Who is, believe it or not, your own beloved editor. At New Generation's promotional party a couple of weeks ago, the computer Press were competing for this coveted title (and the accompanying free weekend for two in Paris). The editor of Personal Computer Games, Chris Anderson, got away to a flying start with a massive 9142 points, but your editor's deftness with a joystick pipped him by 45 points in a nail-biting finish which left everyone else standing (third place was Tony Hetherington of PCW with 3100-odd). Thanks for the prize, guys.



TANDY'S 2000

The latest microcomputer to be launched by Tandy in the UK is the high performance MS-DOS system, the Model 2000. Aimed at the professional user, it uses an Intel 80186 16-bit microprocessor, almost three times as fast as other MS-DOS based systems currently in the marketplace. The Model 2000 disc drives (5¼" floppy disc) have over four times the storage of drives in competitive computers such as the IBM-PC. The system also has twice the colour resolution (640 by 400) and twice as many colours (eight) and optionally features a built in 10-million character hard disc drive.

A wide range of programs will run on the Model 2000 including PFS, Microsoft-Word with its state-of-the-art interactive MS windows, word processing, graphics and filing, Microsoft-Multiplan spreadsheet analysis to the Thinking Software series, and a communications program allowing access with major information networks.

The modular 2000 has a detachable low-profile keyboard and optional Digi-Mouse for easy cursor movement. Its 128K RAM is expandable to 768K and it is available with a high resolution monochrome monitor with 12" non-glare green phosphor screen or a colour monitor with a 14" screen. The case is white, and has an 8½" by 12¼" footprint.

If you're shopping around for other equipment, though, a new comprehensive full colour catalogue of the full range of Tandy computers and ancillaries is now available free of charge from all 228 Tandy stores and participating dealers.

Containing 47 clearly illustrated pages it includes concise information on the Tandy ranges of large and small desk-top business micros, portable and transportable models, home, educational and colour computers as well as details of the various operating systems and applications software. It also features printers, accessories, educational systems and a wide variety of computer centre training sessions. Each section clearly defines the user area, specifications, price, and summary of special points.

Tandy's high street computer centres will normally carry the full range of equipment specified in the catalogue but it will prove especially useful as a reference in other outlets where only a limited computer stock is carried.

Copies of Tandy's 1984 Microcomputer Catalogue may also be obtained from Tandy's marketing department at Tameway Tower, Bridge Street, Walsall, West Midlands, WS1 1LA (phone 0922 648181).

Finally, British Telecom's electronic mail service, Telecom Gold, is now available at privileged registration price at £19.95 to individual customers who purchase computer products at Tandy computer centres or already own a Tandy computer.

Tandy Corporation and Telecom Gold have produced the Infocomm package which enables a customer to join the Tandy electronic mail user group and benefit from a wide range of Telecom Gold International Dialcom Network Services. These include instant correspondence, telex, radio paging, external database systems, information storage and Telecom Gold's Helpline facility, and there is the added advantage to the Tandy customer of being able to communicate directly with Tandy computer centres and the Tandy customer service group.

The package contains registration documents and a guide to the service. The application is forwarded from the computer centre to Telecom Gold and the user is shortly allocated a mailbox number, a straightforward guide to electronic mail, a teach yourself tutor and a telephone Helpline number.

Training sessions are available at Tandy computer centres (£39.95 for a ½ day course) to help customers maximise on the service and their particular equipment.

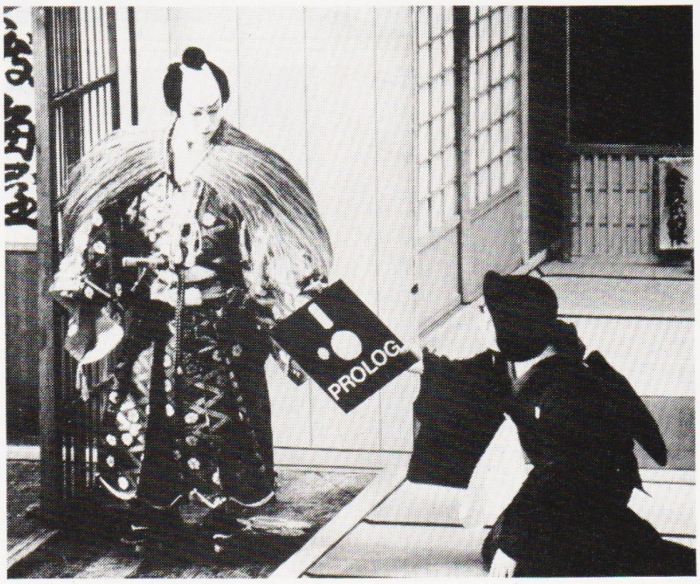
MR AND MRS

We'd like to offer our congratulations to ex-editor of Computing Today, Henry Budgett, and his wife Jennie, on their marriage last month. Can we expect the pitter-patter of tiny peripherals soon?

THE DISC THAT IS NOT ROUND...

. . . the arrow that is not aimed, and other Samurai phrases. In contrast to the normal direction of imports and exports, the British company Expert Systems Ltd has signed a contract with a Japanese company to distribute their range of Prolog-1 interpreters in Japan. Expert Systems' version is in advance of Japanese developments in Prolog, and is leading to a soon-to-be-announced expert system development tool.

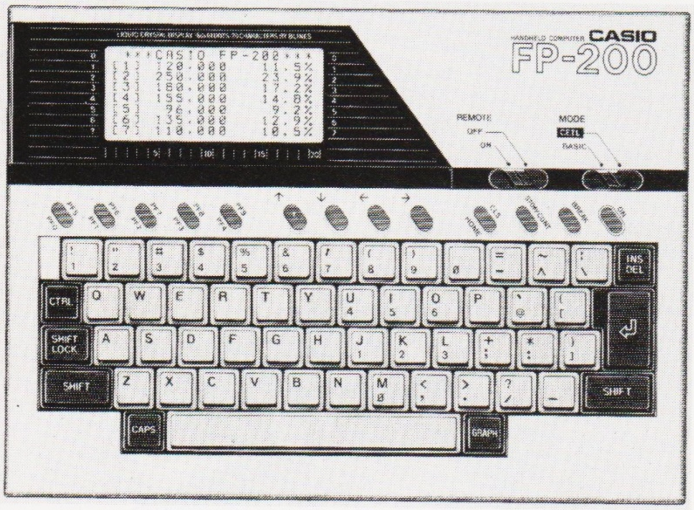
The photo, we think, shows a representative of Expert Systems offering a demonstration disc to a leading Japanese computer expert. Doesn't it. . .?



DRG MEAN BUSINESS

DRG Business Systems have launched a major attack on the fast growing market for computer systems for the small business. Through their Microsystems Division, DRG have announced a low cost microcomputer package called The Business Manager at an entry price of £2995.

The Business Manager is based on the Apricot Microcomputer, voted the Best Business Microcomputer of 1984, and has been designed specifically for the first-time business computer user in that it includes all the hardware and



CARRY A CASIO

Extending the Casio computer range is their latest FP200 design, an A4 size unit ideal for carrying in a briefcase. This handy personal computer has a built-in 'spread sheet' function using CETL (Casio Easy Table Language) for easy structuring and manipulation of tabular data. The FP200 also supports extended BASIC language.

The liquid crystal display, which can be adjusted for optimum visibility at any viewing angle, has eight lines each of 20 characters for easy reading of table formats, with data positioning indexed under CETL through simple file name/row/column address. Alternatively, for graphics, the LCD offers 160 by 64 dot placings.

CETL has only 16 fundamental commands to handle all data editing, processing and input/output. It is therefore very easy to learn.

In standard form, the FP200 is supplied with 8K RAM and 32K ROM. It has an RS232C modem port, Centronics/parallel printer port, plus a cassette socket with remote on/off. Memory expansion is possible in 8K steps up to 32K RAM and up to 40K ROM.

The Casio FP200 mainframe (! — Ed) is a compact 310 by 220 by 55½ mm unit, weighing barely 1½ kilograms and fitting neatly in a briefcase for carriage and on a lap for active use. Based on the 80C85 processor, it is powered by four AA size batteries, with an extra pair of AAs for memory protection.

Optional accessories include AC mains adaptor and the 8K RAM and ROM packs, while attachments available include four-colour mini plotter-printer, RS232C modem lead, and the cassette lead.

The Casio FP200 handy personal computer starts at a recommended retail price of £345 plus VAT: sales enquiries to Casio Electronics Co Ltd, Unit Six, 1000 North Circular Road, London NW2 7JE (phone 01-450 9131).

---

software requirements of a small business, together with a comprehensive package of computer supplies sufficient for at least three month's normal usage.

Software provided with Business Manager includes a general ledger accounting system modelled closely on a manual book-keeping system which allows the user to implement the system quickly and with minimum disruption to the normal operations of the business. As the user's experience of the system grows, the system grows, the full facilities of a highly sophisticated computerised accounting system can be implemented on a gradual basis to accommodate the requirements of the particular business. Upgrades to the system can be added with minimal disturbance to normal operations.

A spreadsheet business and financial planning package is included with the system to allow the small business to develop powerful models of business operations, and to produce forecasts of business activity and cash flow, previously inconceivable for the small business. If desired, this package can be upgraded to allow results of a business model to be presented in graphic form, to allow trends to be more easily interpreted.

Wordprocessing software allows the creation of letters, memoranda, reports, and mailings. The wordprocessor also includes a flexible mailing list facility for the production of standard letters and a spelling check utility to eliminate typing errors. Full facilities are available to create dictionaries for specific trade nomenclature and jargon, thus reducing dramatically the requirement for typing staff to be fully trained in the 'language' of any specific business.

Communications facilities allow the user to establish a link with other microcomputers. And if desired, an on-board autodial modem is available giving access to Telecom Gold (an electronic post system) which allows instant distribution of messages to remote offices, as well as the capability to transmit telex messages direct from the computer keyboard. Further upgrades are planned to allow full access to the Prestel Information system.

A calendar and addressbook database package is provided allowing the user to store details of customers, suppliers, and important dates in an easy-to-use format.

The software provided is all accessed via a specially developed menu system with on-line help facilities which allow the user full access to the facilities available without specific computer knowledge or experience. More sophisticated users can upgrade this system by developing their own menus to further assist their staff in getting to know the system.

A high quality dot matrix graphics printer is included in the package for computer output, and a letter quality daisywheel printer can also be connected if typewriter quality output is a priority. All necessary cabling and connections are supplied as standard.

The DRG Business Manager package is only available from the nationwide chain of DRG Microdealers who are equipped to supply the local back-up and support necessary to ensure that users obtain maximum benefit from the system.

For further information contact Chris Lindesay/Linda Good DRG Business Systems, Microsystems Division, 13/14 Lynx Crescent, Winterstoke Road, Weston Super Mare, Avon (phone 0934 32525).

BET ON QUICKSILVA

The Argus Press Group, publishing arm of British Electric Traction (BET) has acquired one of Britain's leading computer games software companies, Quicksilva Limited and its US associate, Quicksilva Inc, for an undisclosed amount.

Quicksilva, founded four years ago by software entrepreneurs Nick Lambert and John Hollis, was one of the first companies to produce highly visual, fast-action games for popular makes of microcomputers.

Quicksilva Inc operates as a sales and marketing arm throughout the USA for Quicksilva products and those of other British software houses, including Virgin, A & F, and Salamander.

Argus Press plans no major change to Quicksilva's existing operations. It will continue to be run by managing director Rod Cousens as an autonomous company within the Argus Press Group, based at its Southampton offices.

Chief executive of the Argus Press Specialist Magazines Division, Jim Connell commented, "Quicksilva have rapidly established themselves as one of the leading software companies within the United Kingdom and this investment furthers our expansion within the software market place. I am delighted that Rod Cousens will be continuing in his role of managing director of the company to mastermind its expansion."

Rod Cousens stated, "I look forward to working with a new board of directors who are committed to the continued growth of Quicksilva. This marks a new era for the company. We are confident, enthusiastic and excited at the prospect of future developments which will enable us to maintain our position of prominence in the market."

A WORD IN YOUR EPSON

Epson is including the Intext word processing and communications software package by Talbot Computers of Bournemouth with all HX-20 portable computers shipped over the coming months. The software is provided as a ROM (read only memory) and comes with a comprehensive user manual. The package transforms the HX-20 into a powerful communications system for easy access across telephone lines to other databases or electronic mail systems.

The Intext package allows the HX-20 to perform as a portable memory typewriter with its own integral printer for proofing; a microcassette drive for storing text and an LCD (liquid crystal display) for viewing and editing.

A text file of up to 12,500 characters can be created on the standard HX-20, or increased to 29,00 characters using an optional memory expansion unit. Standard editing facilities include insertion, delete and search. Special function keys make it possible to

W. H. SMITH'S BRANCH OUT

On 1 June, the first W. H. Smith Business Computer Centre — quite separate from the High Street branches — opened in Crawley, Sussex. This is the first of three centres opening this year in the south of England. W. H. Smith Business Computer Centres intend to become a national chain.

Mr Val Lewthwaite, the W. H. Smith Divisional Director responsible for the Business Computer Centres, said: "Computers are now very much part of our everyday lives, yet market research shows that about 70% of all small businesses have no computer of any kind. Microcomputer applications can be a huge help to small businesses and the professions. Through our W. H. Smith Business Computer Centres we will help businessmen and women make the best use of the new technology available to them.

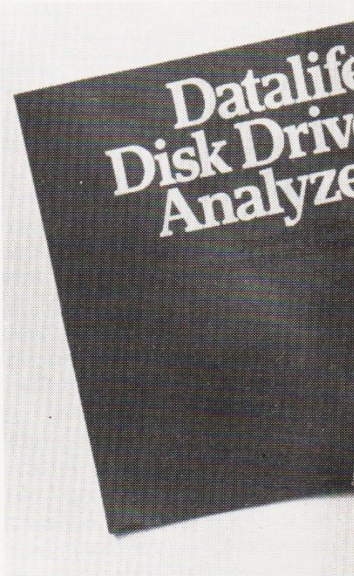
Key features of the W. H. Smith Business Computer Centres will be:

- A large staff of highly-qualified communicators, combining considerable business and computer software knowledge.
- An emphasis on spending as much time with customers as is needed. Said Mr Lewthwaite: "We want to understand the problems and opportunities of customers' businesses. We must gain their trust and confidence and provide support in the way of service, advice and training."
- A training room — called a Computer College — is an integral part of each Centre. This will be run by an experienced software trainer.
- Flexible opening hours to meet the diverse needs of customers. It is appreciated that it is often quite difficult for businessmen to find sufficient time within their regular working schedule to learn about microcomputers.
- Comprehensive after-sales service — two engineers will work from a fully-equipped workshop at each Centre.
- Carefully selected sites in high traffic locations on the edge of town centres with good parking facilities.

The W. H. Smith Business Computer Centres are targeted at small businesses and professional firms. The relaxed yet professional approach will encourage customers to discuss their business requirements in everyday language. They will be encouraged to sit in front of the product and use it extensively before purchase.

Mr Lewthwaite said: "We want to build a relationship with our customers akin to that of a general practitioner in the medical profession. First of all we will undertake diagnosis to establish the nature of the customer's problem; secondly, we will prescribe a solution; and third, we will undertake longer-term aftercare."

The computer hardware available at the W. H. Smith Business Computer Centres comes from WANG, ACT, Hewlett Packard and IBM. A wide variety of proven software suitable for business needs is stocked, as well as computer peripherals.



jump to the top or bottom of the text and to scroll through a document, four lines at a time.

Frequently used words, short phrases or passwords and IDs for electronic mail can also be programmed into special fuction keys which allow insertion in the text file with a single keystroke. Similarly, user-selectable codes can be defined to indicate bold face, underlining, centering, tabbing and so on.

The integral communications facilities allow users to send and receive messages across the telephone line using Telecom Gold, Comet or other user networks. The HX-20 also functions as a portable telex machine sending messages from anywhere in the world. These can be automatically routed by Telecom Gold or through Talbot Computer's own Intext Users Group for a registration fee of £27 plus a monthly rental of £7. Alternatively, messages can be sent directly to any location with an appropriate receiving station using Talbot's special Dialtext remote printing system.

While the HX-20 has its own integral microcomputer, it can also be used with external printers for producing hard copy and communicates with other computer equipment including minis and mainframes.

The Intext word processing with comms package, normally costing £75 plus VAT, is now included in the standard HX-20 price of £411 plus VAT. For further details, contact Epson at Dorland House, 388 High Road, Wembley, Middlesex HA9 6UH (phone 01-902 8892 : telex 8814169).

BEEB'S PRESTEL

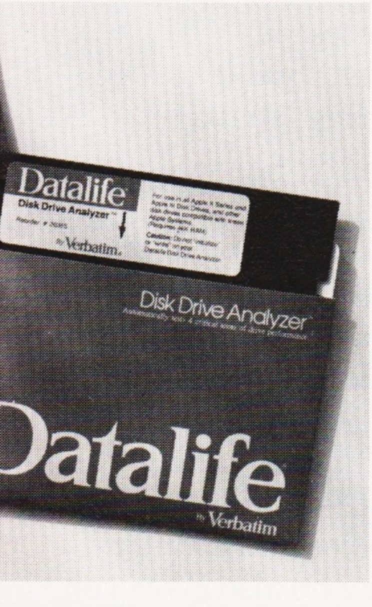
A new viewdata interface that links the BBC Microcomputer to Prestel and electronic mail services has been launched by Acorn Computers Ltd. The Prestel Adaptor connects the Beeb directly to the telephone network, turning it into a powerful two-way computer terminal. The system can then automatically dial-up and access remote computers, including the Prestel and Telecom Gold facilities.

With the Prestel Adaptor the BBC Micro can play new and important roles in the office or at home. For example it can tap the huge database of information and consumer information published by Prestel, send and receive instant (and secure) electronic mail via Telecom Gold, and handle the increasingly sophisticated Prestel services such as teleshopping, Micronet 800 and Viewfax information.

Acorn's Prestel adaptor features a number of useful facilities. For example, it can handle telephone numbers stored by the computer on disc or tape. Together with the autodial facility, this makes connection to frequently-used services fast and reliable. The adaptor also contains special built-in software to download telesoftware programs from the Micronet 800 database on Prestel.

The Prestel Adaptor is packaged in a cream-coloured case matching case matching the BBC Micro. It plugs into the RS423 port on the micro and the Type 600 BT telephone socket. The unit operates in full duplex mode, baud rate 1200 (receive)/75 (transmit), and conforms to the CCITT V.23 specification.

The Adaptor costs £113.85 including VAT, and comes complete with a viewdata telecomms ROM (which plugs into one of the spare sideways ROM sockets inside the micro) and comprehensive user guide. It is available only by mail order from Vector Marketing, London Road, Denington Estate, Wellingborough, North Hants NN8 2RL.



DISCS ON THE COUCH

The Disk Drive Analyser, a new Verbatim product from BFI Electronics Ltd, is a floppy-disc-based program designed to test and display a diagnostic report on certain disc drives as they are running. At present the analyzer may be used with Apple II, Apple IIE and IBM PC disc drives, but will be extended soon to cover other makes and formats. BFI claim that operation is so simple and quick that it will eventually become a popular accessory, particularly in high volume business environments where optimum performance is a prime requirement of heavily used drives.

The analyzer package includes a test disc and an operating guide providing simple step-by-step instructions. The disc is placed in the drive and the required test selected from a screen menu. A full test program may be carried out, including checking radial alignment, disc speed, disk clamping and read/write performance, or any one of these as an individual test.

The test disc generates a series of precision signals designed to ensure that if the disc is not centred correctly the read/write head picks up a substandard signal. The analyzer software analyses these results as good, fair or poor. In a similar fashion RPM may be monitored accurately from the frequency rate of the signals. Again, the screen display indicates speed accuracy as good, fair or poor.

The disc clamping mechanism is also monitored, as are the write/read functions. In the latter test, the drive is required to record and then play back a series of random numbers. The result in this case is simply a pass or fail.

The drive analyzer is designed to ensure peak drive performance at all times, and takes only a few minutes to run through all tests. The screen display and simplified menu are exceptionally user friendly, making it difficult even for beginners to make errors. It is backed by a full one year warranty.

More information can be obtained from BFI Electronics Ltd, 516 Walton Road, West Molesey, Surrey KT8 OQF (phone 01-941 4066 : telex 261395).

Now, the BBC

The BBC Micro has now taken a giant step into the world of business computing.

With the addition of its new Z80 second processor, it is the first computer at anywhere near its price to become fully compatible with CP/M software.

As most business computer users can verify, CP/M is the most widely used form of software in business today.

For £299, you're well and truly in business.

At £299, the Z80 adds 64K of usable RAM to the BBC Micro. And it allows you to use the CP/M 2.2 computer operating system.

It's extremely fast.

And besides giving you access to a vast new area of software, it enables you to use GSX graphics–based programs, the perfect complement to the BBC Micro's own superb graphics.

Free software and languages.

The Z80 second processor comes complete with five CP/M business programs.

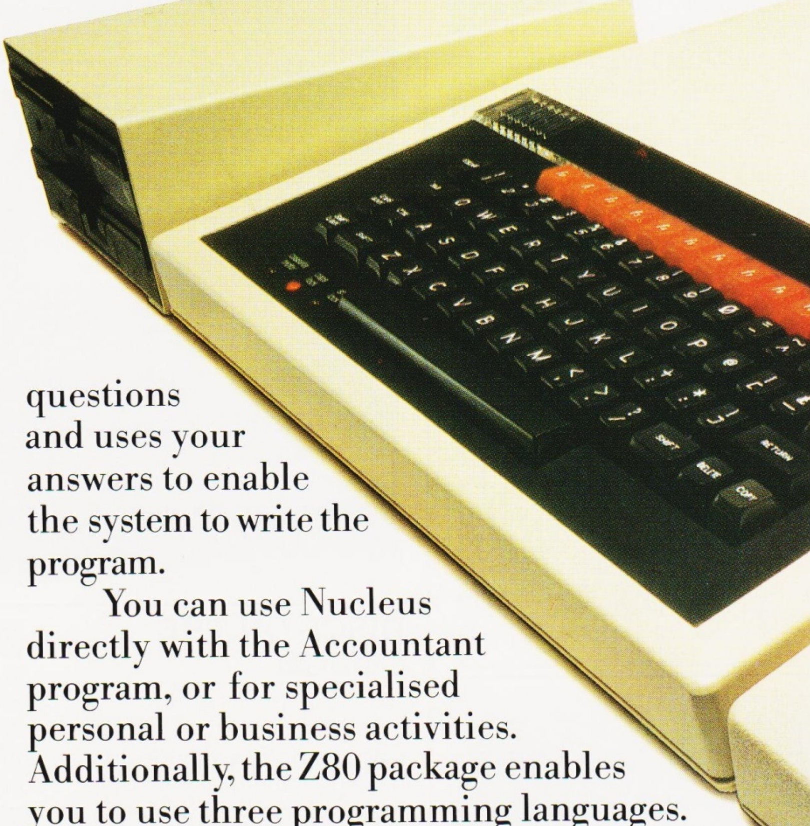
To handle your word processing, there's MemoPlan. It's a program with some highly sophisticated features, such as a safeguard against data loss through power cuts and the ability to show two documents simultaneously on the screen.

To form your CP/M personal database, there's FilePlan. It stores names, addresses, telephone numbers, stock listings and more. And if you use it with MemoPlan, you can generate personalised letters, labels and mail shots.

To produce forecasts and analyse groups of figures diagramatically, simply use the GraphPlan program. This is incredibly helpful in working out vital business calculations, converting them into graphs and charts.

Meanwhile, in the book-keeping department, there's the Accountant program. Use it to enter day-to-day transactions into the computer. Then, at any time, you can ask the computer to produce lists, summaries, reports, audit trails and trial balances. You can readily expand this package to a fully ledger based system, complete with payroll and more.

Finally, to help you to develop your own programs without having specialised experience, the Z80 comes with another software package called Nucleus. It's a system generator which asks you



questions and uses your answers to enable the system to write the program.

You can use Nucleus directly with the Accountant program, or for specialised personal or business activities. Additionally, the Z80 package enables you to use three programming languages.

Your BBC Micro instantly becomes multi-lingual.

To simplify writing your own software with the Z80, there's BBC BASIC.

For running professionally written business programs, there's Professional BASIC.

And then there's CIS COBOL, the leading microcomputer version of COBOL, the language used in mainframe computer applications throughout commerce and industry.

With CIS COBOL, the Z80 also gives you two sophisticated programming aids.

Macro.

One is Animator, an award winning debugging tool which enables you to identify programming errors quickly and easily.

The other is FORMS 2, which helps you to write your own interactive programs in COBOL.

With all these sophisticated features, the Z80 package is exceptional value for money. Indeed, bought separately the programs and languages could cost as much as £3,000.

<u>See the Z80 at work.</u>

The Z80 second processor is designed to be used with the BBC Micro Model B incorporating a Series 1.2 Machine Operating System and linked to a dual 80-track disc drive, a printer and monitor.

Ask your BBC Micro dealer to show you just how far it can go in the world of serious business computing.

For your nearest dealer, ring 01-200 0200.

<u>Technical specification.</u>

The Z80 has a 64K Random Access Memory, running CP/M 2.2 which provides approximately 55K bytes of RAM for user programs.

It operates at a clock rate of 6MHz.

Power supply is integral. Height, 70mm. Width, 210mm. Depth, 350mm.

The BBC Microcomputer System.

Designed, produced and distributed by Acorn Computers Limited.

IT'S SHOW-TIME

Suddenly it's exhibition season, with no less than four major shows within the space of a month. Your editor has tired feet. First off was Cetex at Earl's Court, a trade show which saw the first major public demonstration of the new range of MSX machines from the Japanese. JVC were demonstrating the video interfacing of their computer, which is one of the only ones I know that utilises the 'External Video Input' pin on the Texas 9918 display processor chip. Consequently it's possible for the JVC machine to mix its computer graphics with the output from a video recorder or disc player. The concept conjures up interesting possibilities for both games and training or educational applications.

Toshiba were exhibiting their entire range of electrical goods on a vast stand, with the MSX range tucked into a corner running some rather impressive games software on cartridge. The graphics quality was just about the highest of any machine I've yet seen that uses the Texas chip.

Mitsubishi's MSX offering was a bit on the weak side, with two demo computers doing nothing much at all while I was on the stand, and a brief leaflet. On the other hand, they had a most amazing demonstration of something we've waited a long time for. Hang-it-on-the-wall flat-screen TV is here. It isn't cheap and it has a 'pixelly' look but it works exceptionally well. It doesn't work like a normal LCD display — that wouldn't be bright enough — but uses fluorescent backlighting to produce a really luminous image. The panels are modular, so if you feel like it you could cover a whole wall: the version I saw was about 4' by 3', and together with the controlling computer, mixing desk, professional video camera and so on, costs about £250,000. Start saving now.

Dragon Data had a standful of goodies, like a C compiler for only £79.95, the new Dragon Professional (a bit like a smaller Dragon with twin Sony 3½" floppies on top and the OS9 operating system inside), the fairly hush-hush Dragon Beta with lots of memory, colours, bells and whistles, and a great deal of software. Two days later I read that they'd called the receiver in. . .

Star of the show for me, though, was Markplan's stand. They were exhibiting a remarkable gadget called called the Star Sculpture, which consists of a glass sphere about 12-15" in diameter sitting on a black base. Inside there's a near-vacuum of rare gases, and when it's turned on, 'living lightning' arcs from a central bulb to the inner face of the sphere. The colour of the plasma discharge seems to depend on the charge density, so the central bulb ripples with a red glow like the surface of a star, while blue-green fingers of fire stream outwards. As you run your hands over the sphere, the lightning alters to flow into patterns around your fingers. Microprocessor-controlled (of course), the thing is almost impossible to describe adequately, but I fell in love with it and I want one. Catch is, they cost about £2000 each. Now if everybody reading this sent in 5p. . .

Star of the Office Automation Show at the Barbican for me was the Apple stand, with the Lisa 2, Macintosh and Apple 2C all up and running (until I crashed a 2C. Harumph). I was told that Apple are still on schedule for their flat-screen 80 by 25 portable display to be ready by Autumn (can't wait) and then wandered down to the Macintosh classroom to play with the beast. After an hour's tuition by a slightly over-the-top chap complete with gown and mortar board, I entered the Win-an-Apple-Mac-a-day competition. Unfortunately I only got a consolation prize of a T-shirt covered in Apple advertising. Maybe I'll get it overprinted ("I went to the Macintosh stand and all I got was this lousy T-shirt"!).

Next on the agenda was the Fifth International Commodore Computer Show at the Novotel (formerly Cunard) Hotel. Commodore, as one of the few big companies who didn't make a massive loss last year, were understandably banging the drum with their "Number one and here to stay" message. Two new home computers were launched: the Commodore 16, a beginner's machine with 16K RAM, full-size keyboard, 121 (!? — Ed) colours and advanced BASIC: and the Plus/4. The Plus/4 straddles the gap between home and business computing, with 64 K RAM and four built-in application packages (definitely an idea whose time has come). The packages are the standard business set: word processor, spreadsheet, database and business graphics. There is a screen window facility so that you can view two packages simultaneously and the whole thing costs £249. The Commodore 16 is priced at

Centurians play APS's Fall of Rome at the Computer Fair. Is it in Latin?

Supersoft's Peter Calver sneaks a plug for his software at the Commodore Show.

The Commodore 16 is a new model for beginners.

The Plus/4 is Commodore's home/business model.

The Z-machine — censored by Commodore.

What lovely legs — and the girl's nice too.

£129.99 including a cassette unit, an Introduction to BASIC and four games.

Another Commodore launch at the show was Compunet, which will offer the full range of viewdata services for the price of a modem (£99.99) and a subscription. (The first year's subscription is free to purchasers). As someone pointed out to me, Commodore have manufactured more modems than Micronet already has subscribers. . .

Now, a tale of left and right hands. The blank photo is courtesy of Commodore. Later that day I attended Commodore's preview session of their two new business machines (still under development) — an IBM PC-compatible which Commodore hopes will make it the major competitor to IBM, and the Z-machine. This is the codename for a Lisa-like machine being developed by Commodore's Z-Team (no relation to George Peppard), although it was quite fascinating listening to the team leader trying to describe the computer without actually mentioning the Lisa, or Apple, or icons. The prototype, which features a 1024 by 1024 screen (graphics memory alone is 128K!), windows, mice and the rest, was housed in an old PET case because the "beautifully styled" case was still coming through Customs. I asked if I could get any photographs and was told I could come back at any time.

I did so on the Saturday. Having lugged my camera equipment from one side of London to the other, and being given the run-around for a couple of hours, I was told by a second individual that no Press photos were permitted; I didn't even get a peek at the thing. Commodore told me they were sorry: I wish I believed them. To paraphrase Lily Tomlin on Bell Telephones: "We don't care. We're Commodore. We don't have to."

I'm not sure why I should offer advice after that, but in my opinion the Z-machine, which is just the project name, should be marketed under that title. It has a really nice ring to it, but I expect the thing will go out with just a mouthful of numbers for identification.

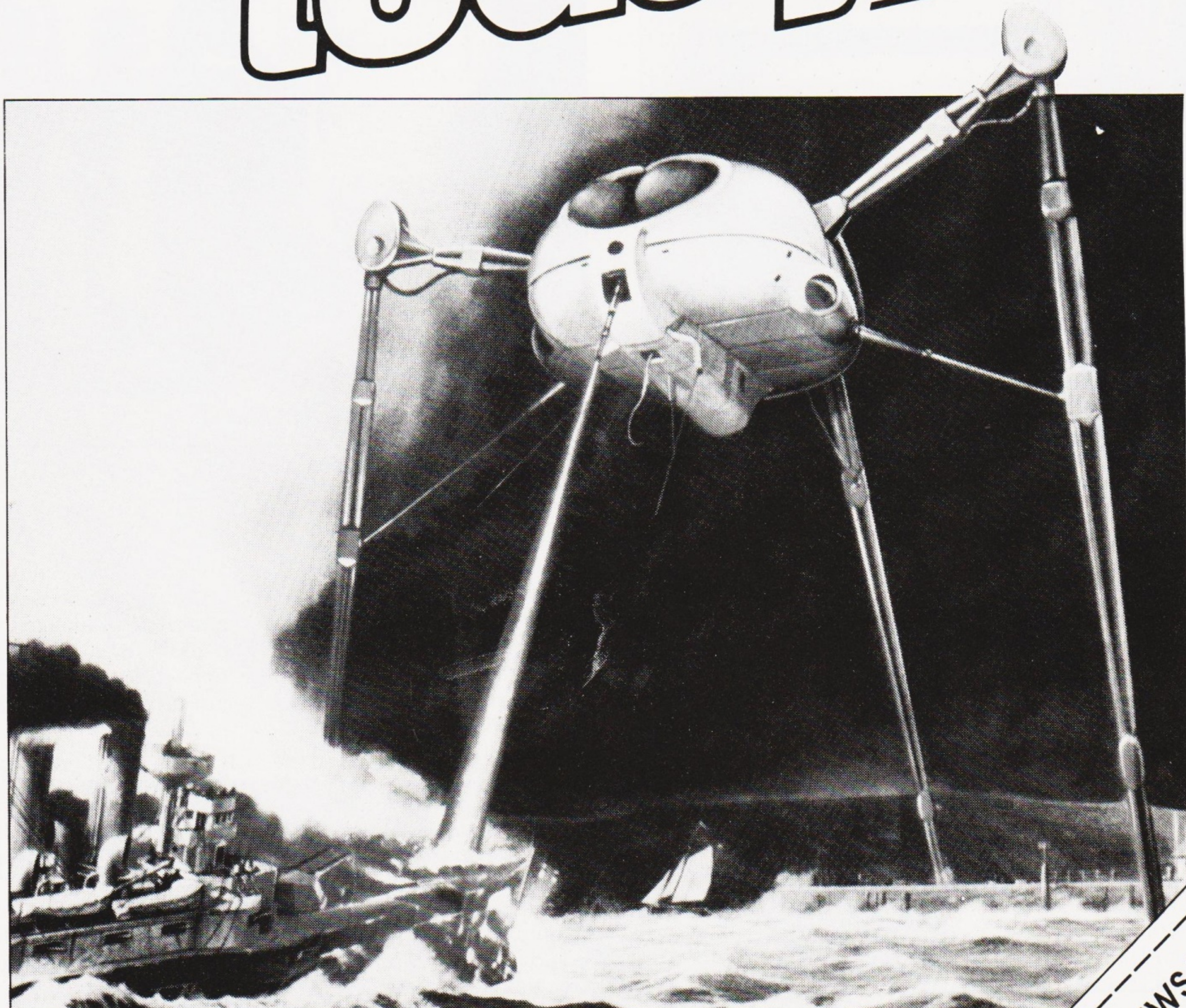
Finally we had the Computer Fair at Earl's Court, which was something of a disappointment with no new products being launched, and several major firms like Amstrad absent altogether. However, Dragon Data were at this show too, full of high spirits and confident that a rescue operation would salvage things. I hope so: it's been a long wait since the Dragon 32 but they they now have some good products coming along.

Of course, one of the attractions of the Press Preview at this show was seeing all the odd characters wandering around, such as the Hulk and Spiderman, a floppy disc with large feet, a robot, some Roman centurions and Roger Munford. . .

NEXT MONTH

computing today

SEPTEMBER ISSUE
ON SALE
AUGUST 10th



In next month's Computing Today our Adventure reviewer will be running for his life from the Martian heat rays in War of the Worlds, as well as learning about economics, hand-to-hand combat and downright low cunning in Parts 1 and 2 of the Ket trilogy. Control Universal's 6809 Second Processor for the BBC Micro will be on our testbench, as well as Commodore's portable SX-64. We'll be looking at a new range of machine code tutors for the BBC, 48K Spectrum, Atari and Commodore 64, and of course there will be useful programs for a variety of machines. Don't miss the September issue.

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

GIVE THIS TO YOUR NEWSAGENT

Please reserve me a regular copy of Computing Today

NAME ....

ADDRESS ....

POSTCODE ....

THE EXCELLENT XI

Simon Dismore

ACT, originally the UK vendors of the Sirius, decided to design their own Apricot machine to avoid problems with US supplies. Their first model, announced with much razamatazz in 1982, has been quietly joined by a bigger brother — is bigger more beautiful? The Apricot xi makes a powerful case.

The excitement of the original Apricot launch (notable for the use of a lithe blonde model and the subsequent charges of sexism which were levelled at ACT for months afterwards) seems hard to recapture now. The original machine had a very different specification from most of its competitors: housed in a gleaming white case was a full 256K of RAM, compared with the meagre 64K or 128K offered on most systems. The Sony 3½" disc drives seemed revolutionary, and the liquid crystal 'microscreen' with touch-sensitive function keys took away one's breath.

Today, the Apricot seems more 'different' than revolutionary. To a certain extent, disillusionment has set in. The Sony drives, though very reliable and comfortable to use, proved to be a little slow for the demands of 'serious' work. Many more systems are marketed with 256K RAM as standard nowadays — and there are even some doubts about the advantages of the liquid crystal screen.

The new Apricot xi is ACT's new chance in the fiercely competitive 16-bit business computer market. It incorporates two striking design changes. The first is entirely cosmetic — the system is encased in dark grey plastic instead of the gleaming white which ACT had made their trademark (ACT would no doubt call the colour 'anthracite' by comparison with the Cavalier CD and the Escort XR3i). The second change is genuinely exciting — one of the Sony floppy disc drives has been replaced with a 10M hard disc (a 5M configuration is also available).

REVELATIONS

Reviewing the Apricot in this new guise was a revelation. Fast disc access creates a feeling of confidence and smooth operation which is far less easily achieved on the floppy disc version, and (as you might expect) disc-bound operations like databases become far less onerous. ACT plan to support partitioning of the hard disc such that, for example, MS-DOS can have 4M and Concurent CP/M can have 6M. However, you should note that the CP/M and Concurrent CP/M versions of the hard disc are still under development.

The back panel of the xi. From left to right we have the keyboard port, printer port, RS232C port and monitor port.

The price of the 10M Apricot xi is £2995, compared with £1890 for the standard dual floppy model. The 5M configuration is currently priced at £2695, which represents rather a poor bargain compared with its bigger brother — £300 buys you an extra 5M, and any seasoned hard disc user will tell you that your disc requirements are the last place to economise.

As an operating system, MS-DOS is quite similar to CP/M in its interface to the user. "DIR" displays a directory, "TYPE" displays a text file, and so on. To the user, there are three really significant differences: disc access is considerably faster (the MS-DOS routines for allocating disc space seem to be far superior to CP/M, and are further improved by keeping much of the directory allocation in memory, rather than out on the edge of the floppy disc). The two other differences are borrowed from the Unix operating system, which is itself marketed by Microsoft for larger machines under the name 'Xenix'.

Unix (and likewise MS-DOS) permits the output of one program to become the input to another. Under MS-DOS, for example, there is a simple little utility called MORE which reads input and displays it on the screen, stopping at every screenful until a key is pressed. Of itself, this seems a fairly useless facility, but when (say) the directory is 'pipelined' through to MORE the advantages become obvious. No more messing around trying to press Control-S at the precise moment where you want to freeze the screen — let MORE do the work for you.

ACT describe a typical application of this in their User Guide. Given that DIR produces a list of files showing their size in column 14 of each line, and the SORT and MORE utilities, the command "DIR | SORT/+14 | MORE" produces a listing, a screenful at a time, of files sorted in order of size.

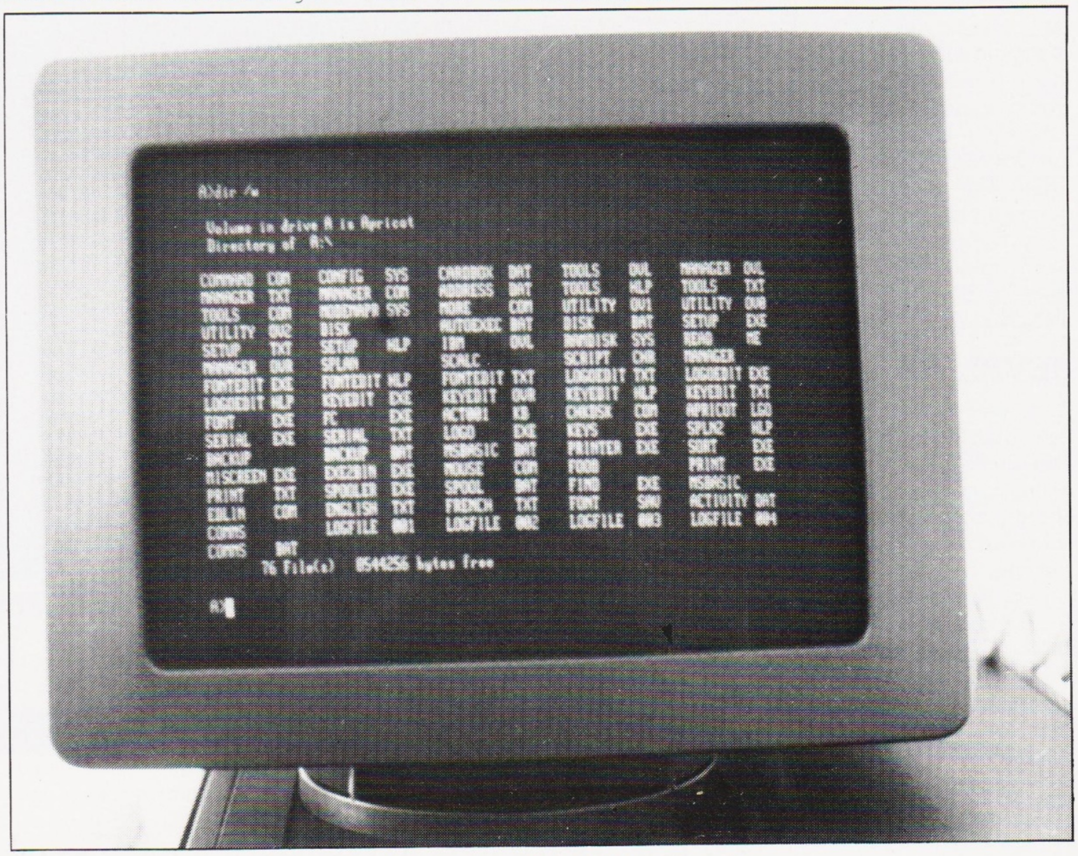
MS-DOS has also borrowed the Unix concept of directory paths, which are a boon when working on a hard disc which might easily contain over 500 files (see our description on the next page). Taken together, these three facilities certainly show that MS-DOS is more than just a one-for-one copy of CP/M. If only the benefits of MS-DOS and Concurrent CP/M could be combined in a single operating system!

HARD FACTS

We found that the 9" screen was not as hard to use as we had expected, though the news that ACT will shortly be supplying a 12" monitor may come as good news to those who have to use that system on a regular basis every day. An anti-glare finish combined with adequate, if not excellent, character designs ensured that the Apricot could be used in averagely bad lighting (and there are tools provided for designing your own character set if you prefer).

The keyboard unit caused unexpected frustration. ACT have not differentiated between the main keyboard and the special editing and cursor keys, which gives a pleasing compact impression to the eye but can often fool the fingers. Non-typewriter keys should have been finished in a different colour, like the function keys on the top row. In use, the keyboard tended to attract errors: it has a very 'soft' touch when typing and the close spacing of the keys meant that it was easy to press two or more at the same time. While this is to some extent a matter of personal preference and early training, it does seem that the keyboard is not the xi's strongest point.

The miniature LCD unit, with six touch-sensitive keys, was


The power of the hard disc. With 76 files stored 8,544,256 bytes are free.

MS-DOS DIRECTORY PATHS

MS-DOS version 2.0 introduced a concept from the increasingly fashionable Unix operating system — directory paths. This is a great advantage for a hard disc machine.

What does this mean? Imagine that you are a manufacturer who uses the Apricot xi for several different applications (for example, mailshots, quotations and a diary which keeps track of sales calls). You have 10 salesmen and two products, so there are (3 x 10 x 2 =) 60 different combinations of applications, salesman and product — only one of which will be correct in any given situation. Unfortunately, all your salesmen call their files SALES.LTR, as they find S2/W/IBO.LTR something of a mouthful after a hard lunch at the negotiating table.

After two or three weeks with the system, inexplicable errors start occurring — I.B. Ontarget's quotation for 2000 sprockets gets confused with the mailshot that C. A. Fastbuck is sending to his widget customers, with potentially embarrassing results. One solution is to give each salesman six diskettes — one for each product and each application.

This is scarcely a sensible approach to a computer which has space for millions of characters on a much faster internal disc, so MS-DOS lets you divide the disc into separate areas for each combination. Each directory can have other directories inside it, and two different directories can contain files with the same name without fear of confusion. MS-DOS mimics the Unix conventions for directory names, using the prefix '\' to indicate a directory within a command. Users can move between directories using the CD (Change Directory) command.

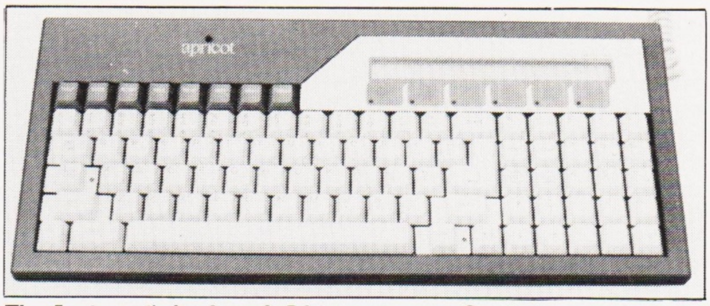
The most fundamental level of directory is called the 'root' directory, which is reached by issuing the command CD \ . You can imagine this to be either the trunk of a tree (with branches, twigs and leaves representing more specialised directories), or the root of a plant, with finer and finer roots descending from the main 'root' — most books on Unix seem to adopt the terminology of trees while printing diagrams which look like the roots of plants!

Each application is reached by 'paths' through the directories, which are invisible to other directories. So salesman Ian can have a file called SALES.LTR containing a quotation for sprockets (full pathname \IAN\SPROCKET\QUOTE\SALES.LTR) which will not be confused with Chris' general mailshot to all widget purchasers, also called SALES.LTR (the full pathname is entirely different: \CHRIS\WIDGET\MAILING\SALES.LTR).

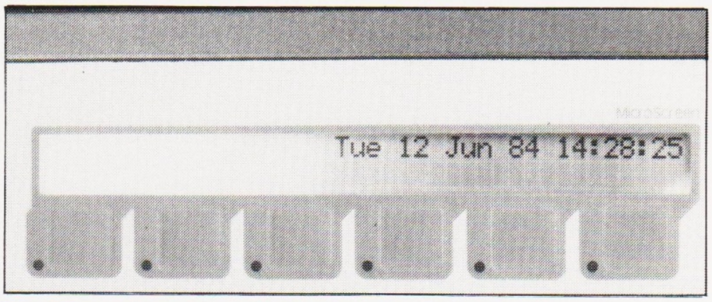
For convenience, MS-DOS can be told which paths to search when looking for programs. So if, for example, SUPER CALC is used to calculate quotations, only one copy need be kept on the hard disc (probably in the root directory). The quotations directories for each salesman are then instructed to look in the root directory for their software.

This all seems very confusing in theory, but the practice is simplicity itself. When, for example, Chris wants to use the system, he issues one command to change to the directory he wants (for example: CD \CHRIS\SPROCKETS\DIARY) and is then free to do whatever he wants without fear of confusion or corruption of other users' files.

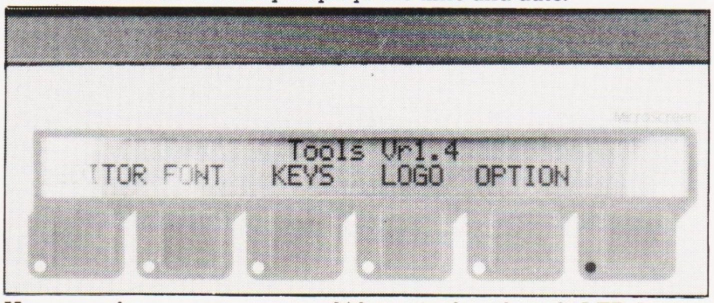
It is interesting to compare this approach with the more rudimentary facilities provided by Digital Research's CP/M operating system (also available for the Apricot). CP/M allows files to be incorporated in different 'user areas', numbered from 0 to 15. This means that up to 16 different combinations of user and application are possible, with any shared data or programs being stored in user area 0 where (if given a special attribute) they can be located by any user. This is much harder to use (users must memorise numbers, rather than shortened names for their applications) and, as we have seen above, even a simple sales office can easily exceed the arbitrary limit of 16 directories.



The Apricot xi's keyboard. A bit too compact?



The Microscreen normally displays the time and date.



Here some keys are programmed (the active keys have lit LEDs).

FACTSHEET	ACT Apricot xi 5M configuration — £2695 10M configuration — £2995
CPU	Intel 8086
Clock	5 MHz
RAM	256K standard, expandable to 768K
Bundled Languages	Microsoft BASIC with MS-DOS Personal BASIC with CP/M
Bundled Products	Manager menu environment Tools editor and reconfiguration utility Async communications software SuperCalc 3 and SuperPlanner
Dimensions	Display: 4.1 kg (10½" by 8½" by 10") System: 5.4 kg (16½" by 4" by 12½") Keyboard: 1.5 kg (16" by 2" by 7")
Display	80 columns by 24 lines Low resolution block graphics characters High Resolution 800 by 400 under GSX 256 User Defined Characters
I/O	RS-232C interface (female) Centronics Parallel interface (female) Microsoft Mouse interface (male) Integral 5 or 10M Hard Disc Integral 315K Sony Microfloppy Drive
OS	MS-DOS 2.11 with GSX bundled CP/M-86 free on request (not yet available) Concurrent CP/M-86 (not yet available)
Options	Expansion memory in 128K increments Intel 8087 floating point processor Asynchronous modem board Microsoft Mouse Colour Monitor (announced, but not yet available) 12" Monochrome Monitor (due in August 1984)

the greatest disappointment of all. Something about the contrast or the character definition seemed to be lacking, and one might in any case suggest that the best place for messages to appear is on the screen, rather than at the keyboard. Frankly, this seemed to be a gimmick which would have been better implemented with two or three additional screen lines, and fewer unconventional keys.

SUPER SOFT

The software that was provided was superbly documented. Supercalc behaved as expected, the asynchronous communications were supremely easy to operate and (in our tests) totally reliable, though it was surprising that the communications software could not detect the presence of the usual line signals (it was quite happy to transmit even when no other machine was present).

ACT also provide the Super-Planner electronic diary and address book with the machine. This really was a 'noddy' product, and it would be most unlikely to attract any serious users. It provides that simple facility to view a calendar, make appointments and so on, but provided no cross-referencing between appointments and addresses — and the facilities

THE GRAPHICS SYSTEM EXTENSION

Graphics represent one of the most machine-dependent applications it is possible to imagine. Some machines have low resolution colour screens while others offer high resolution monochrome output. Likewise, plotters and printers have a bewildering variety of features, each driven by incompatible instructions. Digital Research have attempted to solve this problem with a software extension to the operating system called GSX (Graphics System Extension).

This allows programs to make logical requests to the operating system for particular graphics operations. GSX interprets the logical operation using a module designed to drive the chosen device. So, for example, the logical operation CLEAR WORKSTATION causes a CRT screen to clear, but on a printer or plotter it ejects the sheet of paper and prompts for a new sheet to be inserted. The 'device drivers' are entirely independent of the machine on which the operating system runs, within the limits of common sense (printing a colour graph on a monochrome printer will be successful, but may be meaningless because all the colours look the same).

Applications programs can even ask GSX to describe the characteristics of the current device — number of colours available, range of text fonts, options of dotted lines, and so on. This sort of device-independence is still quite new, so there is not a great deal of software on the market that will take advantage of all the features of GSX, but it promises to be extremely useful in years to come. We have reproduced two examples of the 'same' graph created using GSX, one printed on a Hewlett-Packard 7470 plotter, the other on a Data Products 8010 printer — only one instruction was required to switch output between the two devices.

Device drivers are available for most popular microcomputers, plotters and printers, and even for more esoteric products like digitising pads, and GSX is available for CP/M, Concurrent CP/M (which is now being re-branded as 'Concurrent DOS') and the new version 2.11 of MS-DOS.

On the Apricot we reviewed, there was little scope to test the features of GSX. Only one device driver was supplied (DDACRT.SYS — the device driver for the Apricot's own screen), but there was a useful demonstration program in Microsoft Interpreted BASIC which showed how calls to GSX could be made without resorting to machine code routines. The documentation for GSX (dated August 1983) seemed to be insufficient for most normal users, and quite inadequate for programming: though listings were given for interfacing several languages (FORTRAN, Pascal, PL/1, Compiled BASIC and C) there was no description of the calls themselves and the parameters they require. We thought that ACT should take more advantage of this powerful feature (remember how popular the Sirius became for Computer Assisted Design?) by providing a wider range of device drivers and more software which would make use of them. If you are considering buying an Apricot for graphics, you should check to see which device drivers and applications are available at the time of purchase.

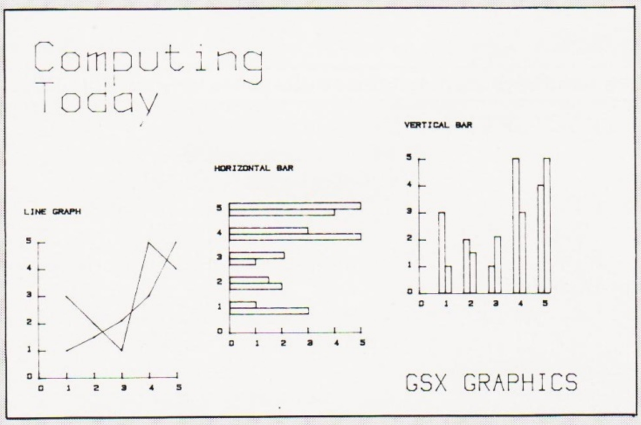


Fig. 1 A sample graph produced by GSX on a plotter.

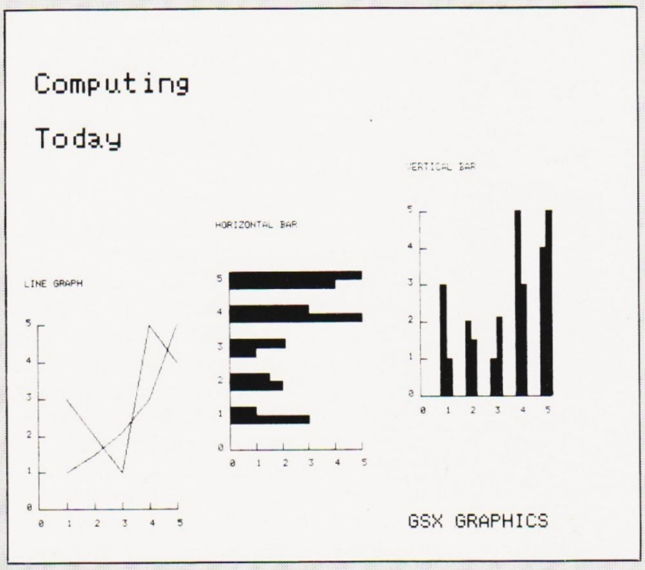


Fig. 2 The same graph sent to a printer.

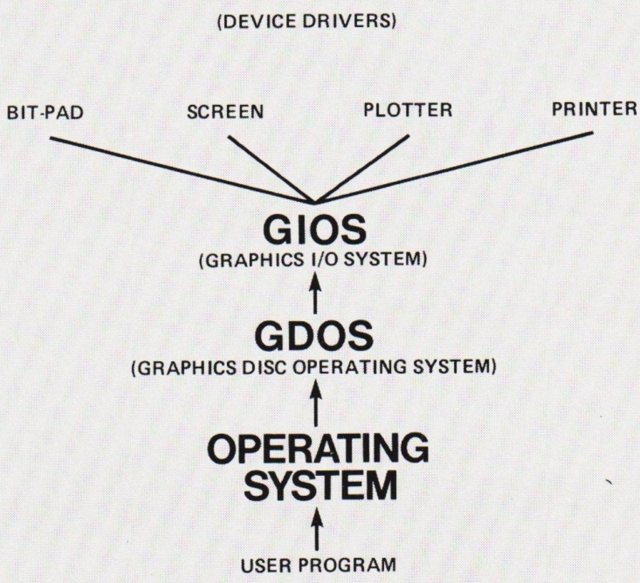


Fig. 3 The GSX system.

for rescheduling an appointment were very difficult to manage. Perhaps the authors should bear in mind that most people are on the telephone when they make their appointments, and do not have the opportunity to indulge in complicated manoeuvres at the keyboard. Still — it comes free with the machine, so it is perhaps uncharitable to complain too bitterly.

CONCLUSIONS

The Apricot xi represents very good value for money in a 16-bit hard disc system, and will be a benchmark for other manufacturers in price and performance terms. We were particularly impressed with the very high quality of the documentation, with copious use of illustrations and even full colour photographs. Though we had reservations about the keyboard, the lasting impression was of a very 'usable' system, with a lot of attention paid to the needs of inexperienced users.

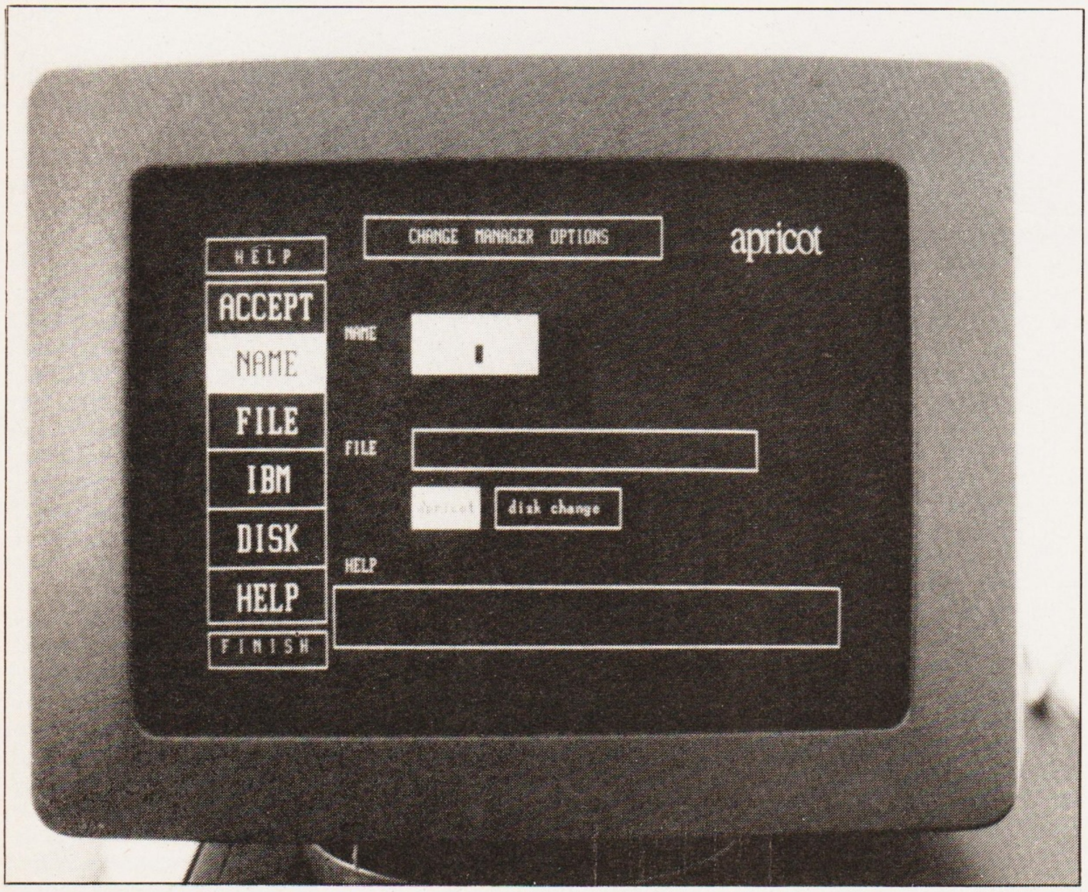
The Apricot front-end menu system deserves a special mention in this context. In essence, it is very simple: you highlight the name of a task you wish to perform and press the Return key, and the menu system locates the program and runs it for you.

The manager menu. Here we have selected 'change an option' (the white cells).

The design of the software is superb: names of tasks appear at four times their normal height, boxed together in a grid that clearly separates tasks (eg Communications or SuperCalc) from options (eg Finish or Help), and all of the cursor and editing keys can be used to move around the display.

The front-end comes as two programs: MANAGER is responsible for displaying options, help messages etc, and can easily be reconfigured by the user with the aid of the second, TOOLS, program. TOOLS is ACT's way of presenting all the unpleasant business of changing character sets, serial printer parameters and so on, in an unambiguous, easy-to-understand manner. The two together put most manufacturers' utilities to shame. Combined with MS-DOS's powerful 'batch file' facility, which allows sequences of commands to be grouped together and given a single name, the Manager makes it possible to produce a truly user-friendly front-end in a fraction of the time required under most operating systems.

Subject to some reservations about the xi keyboard, we felt that the entire system was well-designed for users of all levels of competence. Once CP/M and Concurrent CP/M are available for the hard disc, the Apricot xi will be a very attractive purchase for those who want the benefits of fast data access and high resolution monochrome graphics. ACT have designed a worthy successor to the Sirius.



Once into the 'Change options' file, a new menu is presented.

LEVEL 9'S ADVENTURES

Christopher Moss

Our intrepid reviewer has been feeling a bit jaded lately, but a supply of games from Level 9 have refreshed the parts that other Adventures cannot reach.

I have to confess that recently I'd become a bit jaded by Adventure-playing. Too many tapes were being loaded and run to reveal yet another variation on the theme of keys, lamps, swords, vicious dwarves and hungry beasts. Worse, I was drowning in a monotony of unimaginative descriptions. "You are in a room with stone walls. Exits north, south, west. There is a chest here." Yawn. Luckily I've been sampling some games from a prolific software house which have perked me up again. I thought Adventures had become dull until I discovered Level 9.

Before I go any further, I feel I should make a stand on the subject of text adventures versus graphics. I don't think graphics are worth the effort at the current stage of home computer technology. They eat up so much memory, both in terms of the screen RAM required and coding to draw the actual pictures that any graphics adventure must necessarily be limited in scope. Moreover, the current state-of-the-art of home computer graphics, makes it hard to create really exciting pictorial adventures that stimulate the emotions and generate moods in the player as he explores. It's like the difference between reading a comic book and a well-written novel — the images that the skilful author can create in your mind's eye through imaginative use of the written word have far more impact than the cartoon strip.

Unfortunately, just as a good novel can only come from the pen of a good author, as I pointed out above, the average level of imagination amongst adventure programmers is pretty low. But listen to this:

"You are at the eastern end of a long room with two pits in the floor. You are near the east pit, and the many thin stone slabs littering the room would make descending it simple. A path bypasses the pits to connect passages from east and west. There are holes all around, but the only large one is high above the west pit, out of easy reach".

BIG STUFF

That is the description of just *one* location in Level 9's Colossal Adventure. There are over 200 more locations included in this game, which is a full-size version of the original classic mainframe Adventure that started the whole genre rolling. Amazingly, to add a little something to the original, Level 9 have added a whole new end-game with 70 extra locations: and it all fits into a 32K BBC. This is a quite remarkable feat of programming, made possible because of the company's own 'adventure language' called 'a-code' (I love it when a plan comes together?), together with a 50% text compression technique. Stand up and take a bow, Messrs Pete and Mike Austin — your software impresses the hell out of me.

They have a nice line in humour, too: here's the response when you read the Spelunker Gazzette:

"The main headline is 'Don't go West'. The lead story is about the success of the Dwarven King who has added the heads of another two elves to his collection. The editorial denounces the perverted ways of 'Elvies' and page 3 features a female dwarf whose long grey beard has been positioned ingeniously. The rest is adverts, mainly for Witt Construction Plc (dungeons a speciality) and Acorn Forestry (oaks take a long time to grow — order now for your grandchildren)."

OK, I was complaining about repetitious dwarves and swords at the start of the article, but with text like that (and it is like that throughout the Adventure) the game is given a whole new lease of life. Lives, actually: Colossal Adventure is only the first of three parts of the Middle Earth trilogy, the others being Adventure Quest and Dungeon Adventure. Level 9 give 'recommended solving times' for these Adventures which are either wildly optimistic or assume you've got nothing better to do all day than play games. Their largest Adventure, Dungeon, is listed as an eight-week, (rather than a three-pipe!) problem, but it is *huge* and devious in the extreme. You certainly won't feel you've wasted your money on these games — hours and hours of pleasure are to be had, particularly since no crib sheets are included in the game (well, we reviewers get them, but the general public cannot be relied on to exhibit phenomenal will power!). Fortunately, if an unsolvable puzzle is driving you to distraction, Level 9 include an envelope with a 'clue voucher' which you can send off to them with a question(s) for help. You may well need it . . . but don't use it up too soon!

A MUSICAL INTERLUDE

The two latest adventures to join the Level 9 collection feature a welcome addition: on the BBC versions, anyway, which I got for the review. It takes a long while for all the program to load (after all, practically the whole of the RAM is being filled), so Level 9 have thoughtfully provided some musical relief. A short pre-loader program plays a classical piece of music while the main program is loading. Still showing great style, the tune is quite complex and I heard at least two voices playing in harmony, and though my classical knowledge isn't up to much, I assume the pieces have some relevance to the game titles.

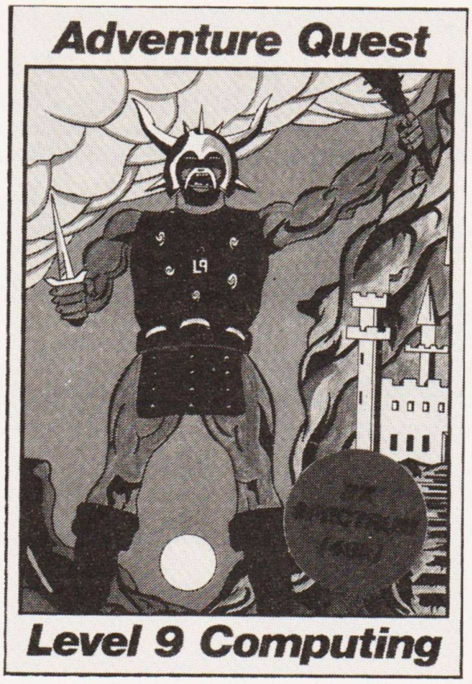
The first of the games is Lords of Time. In this game you have to travel through time in an appropriate but unusual vehicle, collecting various important objects from nine time zones in order to defeat the wicked plans of a bunch of evil timelords. This is something of an 'odd man out' amongst the range as it was not designed by the prolific Pete Austin but by a lady named Sue Gazzard. The Brothers Austin have worked their usual magic with it, though, and it's as complex as any of the others.

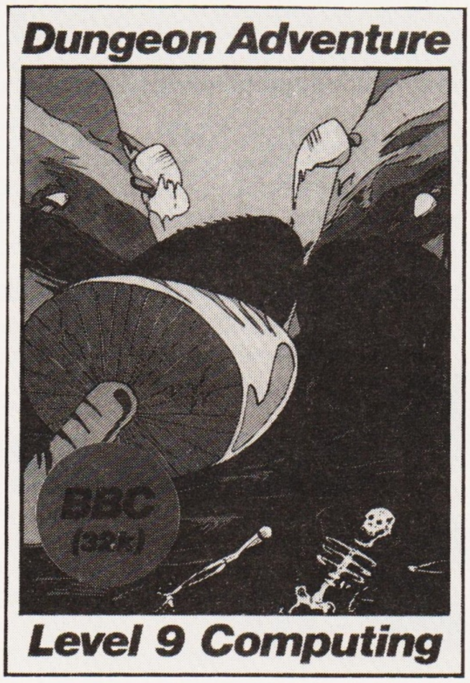
You get quite a bit of variety hopping through the time zones. There's a large mansion sporting a range of items from herb gardens to a Masai spear, a Viking longboat, a Roman town, and some dinosaurs with extremely anti-social habits. Don't try hopping around the time zones trying to get the flavour of the game, as I did, as you'll get nowhere fast: stick to the advice in the manual and visit the zones in numerical order.

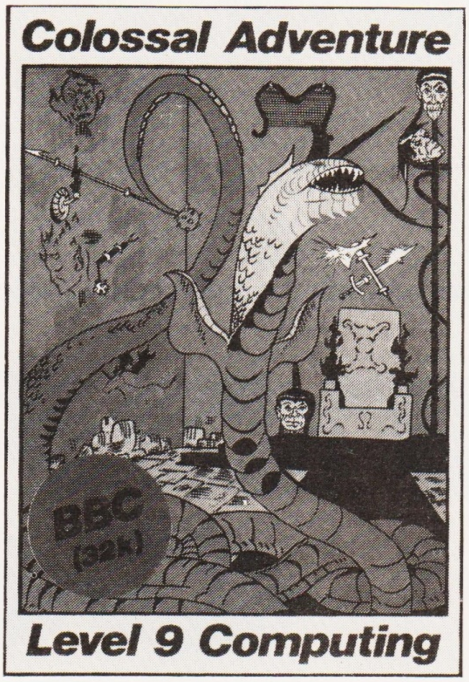
SNOW JOKE

The final game from Level 9 (currently, that is: more are in the pipeline), is Snowball. This is, without doubt, a huge Adventure: the authors claim over 7,000 locations. I'll have to take their word for it, because I haven't visited more than a fraction of that number so far.

The adventure is set aboard the Snowball 9 colony starship,







so named because for much of the journey its bulk consists of a layer of ammonia-ice surrounding the passenger quarters to provide flight-time fuel. It might also have something to do with the several hundred thousand frozen colonists residing in their freezer coffins awaiting revival in the brave new world to which they are headed. Unfortunately, the ship has been sabotaged and is heading for the destination star — literally! As Kim Kimberley (a female hero for a change, and probably no relation to Kimball Kinnison, the famous Lensman), you awake from your hibernation and are faced with the task of saving the whole starship. This is no mean feat as the resident robots, named Nightingales, have been reprogrammed, and far from being 'ministering angels', show more of a tendency to minister sudden death at the end of a syringe.

Apart from being so big, Snowball has a rather novel plot and, of course, the excellent text descriptions of all the other games. Control panels sport coloured indicator lights and buttons: can you work out what they do? How do you get up to the trapdoors in the ceiling (the Nightingales can't climb, apparently). What is a waldroid? (Hint: it's not a piece of confectionery).

I look forward to the two remaining parts of this trilogy, Return to Eden and The Worm in Paradise.

CONCLUSIONS

Looking back over this piece, I notice I haven't mentioned the extremely fast response to the user input for games of this complexity. Coupled with the very detailed background of the worlds in which each game is set, this makes Level 9's adventures more than just a cut above the rest. I can't remember enjoying any game as much since I played some of Peter Killworth's adventures from Acornsoft. I don't think I'd like to decide between these two for the person with the most devious plotting ability, but one thing is certain — a lot more of you can enjoy the games described here because they are available, not just for the BBC Model B, but for the Commodore 64, 48K Spectrum, 48K Lynx, 32K Nascom, 48K Oric, and 32K Atari.

Whichever machine you own, if you have the vaguest tendency towards adventure playing then you must try one of these games (unfortunately you'll probably end up wanting to buy the lot!). If I have one small criticism to make it's that Level 9 should let a competent proofreader give all their text strings the once-over before committing their game to the tape duplicators: there are several annoying spelling mistakes which crop up (it's mechanical, not machanical, for example). But then I'm just a perfectionist.

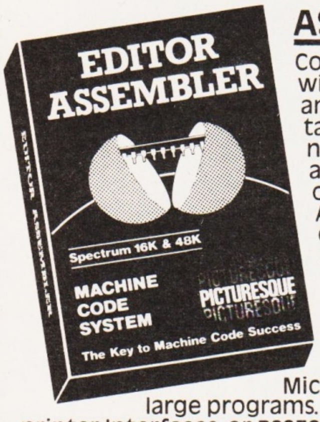
Level 9 adventures cost £9.90 each and can be obtained from them by mail order from 229 Hughenden Road, High Wycombe, Bucks HP13 5PG.

The Key to Spectrum Machine Code Success.

Picturesque's MACHINE CODE SYSTEM is used and recommended by professional software writers, yet the excellent documentation and the friendly, easy-to-use programs have been highly recommended for beginners.

You will only buy one Machine Code System, so buy the best, the one the professionals use.

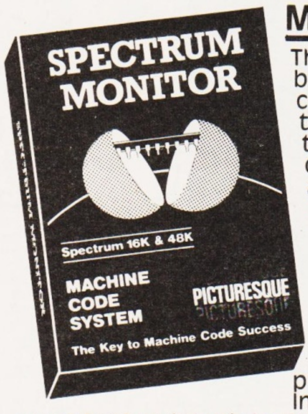
NEW IMPROVED PROGRAMS — MICRODRIVE COMPATIBLE



ASSEMBLER

Completely self-contained with its own line editor, giving an easy-to-read 40 column tabulated listing. Auto line numbering, line renumbering and auto tabulation make this one of the fastest and easiest Assemblers to use. 5-character label names. Decimal, Hex or ASCII constants. SAVE/LOAD/VERIFY both the listing and Machine Code to cassette/MICRODRIVE/NETWORK. Assemble from cassette or Microdrive into memory for very large programs. Customise to most Centronics printer interfaces, or RS232 (with Interface 1) for 80 column printout. FAST ASSEMBLY — 1k of code in 7 seconds. Assembler Directives:— ORG, END, DEFB, DEFW, DEFM, EQU, DEFL. (Microdrive and Centronics facilities only operate with 48K machines.)

£8.50
INCL. VAT & P&P.



MONITOR

The ideal tool to help the beginner get started, yet it contains all the commands for the experienced programmer to run and de-bug machine code programs. Inspect and alter memory contents in Hex or ASCII. Breakpoints and full Register display. NOW WITH SINGLE STEPPING through RAM or ROM. Disassemble any part of memory, RAM or ROM. Dec-Hex-Dec number converter. Printer output to ZX printer or via RS232 (with Interface 1) or customise to most Centronics printer interfaces. General memory management commands include Hex dump, Insert, Delete, Fill and Move. Can reside in memory with the Assembler (48K machines only) to give a complete Machine Code programming system.

£7.50
INCL. VAT & P&P.

Programs supplied on cassette with option to Save onto Microdrive (cartridge not supplied.)

Existing owners can obtain the new programs by returning the **cassette only** to Picturesque, along with a cheque/P.O. for **£1.50 per program** (inc. VAT & P&P). New cassettes will be supplied by return of post.

Available from the "SPECTRUM" chain of stores, branches of John Menzies and all good computer shops, or by mail order by sending cheque/PO to:
PICTURESQUE, 6 Corkscrew Hill, West Wickham, Kent, BR4 9BB. Send SAE for details.

STOP PLAYING AROUND GO TO 22 VIRGIN GAMES CENTRE 01·637 7911 OXFORD STREET LONDON W1.

for the widest range of software for the ZX Spectrum, Commodore 64, VIC 20, and Atari plus many titles for the Dragon, Apple, BBC and Electron.

New titles are stocked within hours of release – call our friendly knowledgeable staff for details.

VIRGIN GAMES CENTRE, 22 OXFORD ST., LONDON W1. 01·637 7911

MODE 7 SCREEN EDITOR

D. S. Peckett

Teletext mode on the BBC Micro is very memory-efficient and versatile, but oh-so-tricky to actually use. This program will let you edit the graphics and text with ease and store the results in a variety of ways.

Creating Mode 0-6 graphics displays on the BBC computer is easy, but things get rather trickier in Mode 7. This is something of a pity, because Teletext uses very little memory and gives a wide choice of colours. The problem is that, although the User Guide gives a good introduction to the mode and there have been lots of articles published on how to exploit it, the computer itself does not have any decent Mode 7 commands built-in.

As a result, therefore, you normally have to create Mode 7 displays pretty well by trial and error, using a series of PRINT statements. Since the display is both line and character oriented, this is a rather clumsy, tedious and error-prone approach.

In this article I will describe a program which makes the whole process much easier. It is a Mode 7 screen editor which gives you full control of what appears where on the screen, changing and moving blocks and characters as you wish. When the display is correct, the program will copy it to tape or disc, or generate suitable PRINT commands which you can insert in your own programs.

The program (SCRED7) will run in a 16K or 32K computer and is compatible with both BASIC I and II. It should also run in a disc-based system, although I have not actually tried that. It does, however, need O.S 1.0 or later.

THE APPROACH

When you first RUN the program, you will see a blank screen, with the cursor blinking at top left. You can move the cursor to any screen position by using the arrow keys normally and type or delete as usual at that point; in this way the program will act as a flexible screen editor.

However, it also makes very extensive use of the red function keys to select operating modes, to switch back and forth between text and graphics, to save and read the screen, and so on. By using these keys in conjunction with cursor movement and the usual alphanumeric keys, it is possible to create very complex displays remarkably quickly.

The key to using SCRED7 effectively is therefore the red keys (sorry) and Table 1 shows their functions. Each has up to four different meanings, depending on whether it is pressed alone or together with the Shift and/or CTRL keys.

The SHIFT/fn and CTRL/fn operations select coloured text and graphics respectively, as described on pages 154 and 155 of the User Guide, while SHIFT/CTRL/fn will choose the remaining Teletext commands. Pressing any of these keys will make all text after the control character on that line appear as text, graphics, coloured, flashing, double-height, and so on, as appropriate. If you select double-height, the program will automatically enter text on two lines for you, so that you will actually type the enlarged text or graphics characters.

With the exception of key f9, you can use the normal function keys to select the program's different support functions. However, f9 will supply CHR$255, which you otherwise cannot get directly from the keyboard; you need it when creating graphics in order to supply a complete 2x3 block of pixels.

IN USE

Let's now take a look at how to use SCRED7. I assume that you understand the Teletext approach of using control characters to affect the way that characters following them on that line are displayed. You will also know that, if you select a graphics control character, lower-case letters, numbers and punctuation appear as blocks of pixels; pages 486-489 of the User Guide show the relationships between graphics and alphanumerics.

I won't, therefore, go into detail about keys SHIFT/CTRL/f0-9, CTRL/f1-8 or SHIFT/f1-7; it's easy to understand what they do. The operation of the remaining keys is described below:

f0 — Copy to Tape Pressing this key writes a complete copy of the screen to tape or disc. The screen will clear as the display is saved in a buffer, and you will be prompted for a file name. Enter one, and you will go through the familiar writing routine, after which the screen will be restored. Once made, the copy can either be re-read by the program (using SHIFT/f0) or you can load it directly to the screen at any time with the command:

```
*LOAD "<filename>" 7C00
```

TABLE 1

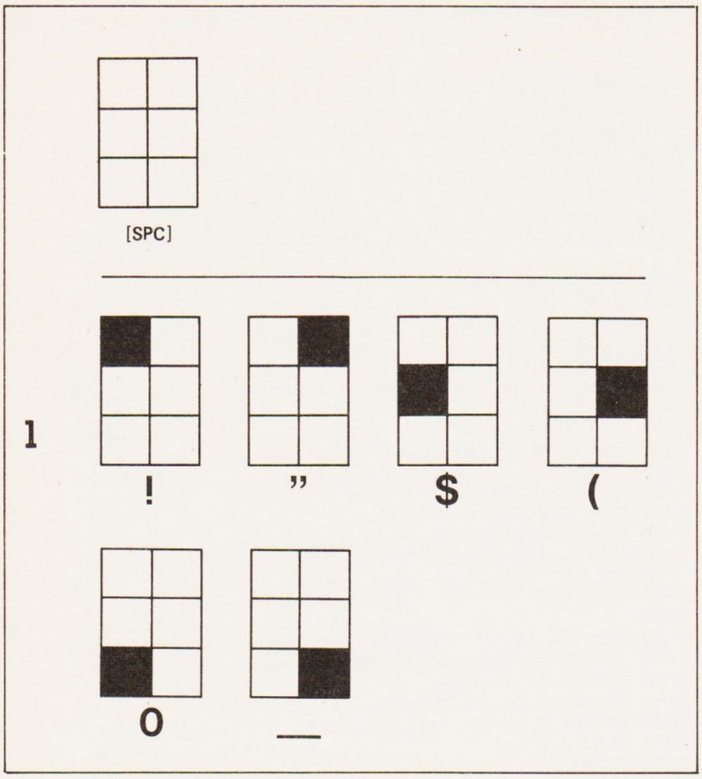
Use of Function Keys

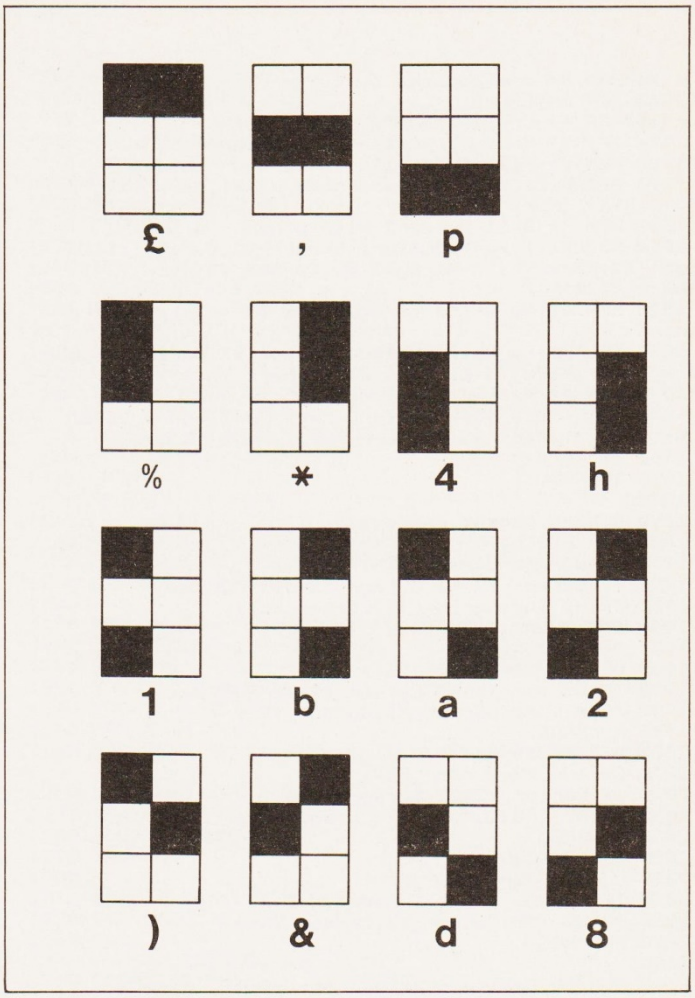
	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
SHIFT/ CTRL	Normal-Height	Double-Height	Contig. Graphics	Separate Graphics	Black B'gnd	New B'gnd	Hold Graphics	Release Graphics	Flash	Steady
CTRL (Graphics)	Read from PRINTs	Red	Green	Yellow	Blue	Magenta	Cyan	White	Conceal Graphics	—
SHIFT (Alpha)	Read from Tape	Red	Green	Yellow	Blue	Magenta	Cyan	White	—	—
Normal	Copy to Tape	Copy to PRINTs	Init.	Clear all Flags	Insert Mode	Overwrite Mode	Marker	Copy	Swap	CHR$ 255

Listing 1. The complete program for the Mode 7 Screen Editor.

```
   10 GOTO280
   20 PRINT "
"
   30 REM ** Plus another 24 lines just like line
 20, with 40 spaces in each
  260
  270 REM ** This will be the last such PRINT stat
ement
  280 ON ERROR GOTO 610
  290 DIM Mark_X%(1),Mark_Y%(1),Buffer1 1000,Buffe
r2 1000,String_Buff 40
  300 PROCInit
  310 REPEAT
  320 Key=GET
  330 REM ** Decode SHIFT/CTRL keys
  340 IF Key<13 THEN Key=100+ASC(MID$("()5689:;$%"
,Key-2,1))
  350 IF Key=24 THEN Key=255
  360 REM ** Must be normal character
  370 IF FNKey_In(32,126) OR FNKey_In(129,139) OR
FNKey_In(145,159) OR Key=255 PROCChar
  380 IF Key=13 PROCNew_Line
  390 IF Key=15 PROCTape_Save
  400 IF Key=16 PROCProg_Save
  410 IF Key=17 PROCClear
  420 IF Key=18 PROCClear_Flags
  430 IF Key=19 Insert=TRUE
  440 IF Key=20 Insert=FALSE
  450 IF Key=21 PROCMarker
  460 IF Key=22 PROCCopy
  470 IF Key=23 PROCSwap
  480 IF Key=27 PROCLeft
  490 IF Key=28 PROCRight
  500 IF Key=29 PROCDown
  510 IF Key=30 PROCUp
  520 IF Key=127 PROCDel
  530 IF Key=128 PROCTape_Read
  540 IF Key=140 PROCChar:Double_Ht=FALSE
  550 IF Key=141 Double_Ht=TRUE:PROCChar
  560 IF Key=144 PROCProg_Read
  570 *FX15,1
  580 UNTIL FALSE
  590
  600 REM ** Trap ESCAPE and shut down  program
  610 CLS
  620 IF ERR<>17 ON ERROR OFF:REPORT:PRINT " at li
ne ";ERL
  630 *FX4,0
  640 END
  650
  660 REM ** Set up system at start
  670 DEF PROCInit
  680 PROCClear
  690 REM ** Reset user-defined keys
  700 *FX18
  710 *FX4,2
  720 REM ** Set base addresses for     function k
eys
  730 *FX225,15
  740 *FX226,128
  750 *FX227,144
  760 *FX228,3
  770 P%=&7C00:REM ** Start of screen
  780 ENVELOPE 1,1,0,0,0,10,10,10,20,-5,-2,-1,120,
70
  790 ENDPROC
  800
  810 REM ** Main display routine
  820 DEF PROCChar
  830 IF Insert PROCSpace(Cursor_X%,Cursor_Y%)
  840 IF Cursor_X%=39 AND Cursor_Y%=24 THEN ?&7FE7
=Key ELSE PRINT TAB(Cursor_X%,Cursor_Y%);CHR$(Key)
;
  850 IF Double_Ht PROCSecond_Row
  860 PROCRight
  870 ENDPROC
  880
  890 REM ** Second row of double-height chars
  900 DEF PROCSecond_Row
  910 IF Insert AND Cursor_Y%<25 PROCSpace(Cursor_
X%,Cursor_Y%+1)
  920 X%=Cursor_X%:Y%=Cursor_Y%+1
  930 IF Y%>24 ENDPROC
  940 IF X%=39 AND Y%=24 THEN ?&7FE7=Key ELSE PRIN
T TAB(X%,Y%);CHR$(Key);
  950 ENDPROC
  960
  970 REM ** Go to start of next line
  980 DEF PROCNew_Line
  990 Cursor_X%=0
 1000 PROCDown
 1010 Double_Ht=FALSE
 1020 ENDPROC
 1030
 1040 REM ** Save screen to tape
 1050 DEF PROCTape_Save
 1060 REM ** Into buffer first
 1070 FOR I%=0 TO 999 STEP 4:Buffer1!I%=P%!I%:NEXT
 1080 CLS
 1090 REPEAT
 1100 INPUT TAB(5,8) "Save screen as which" TAB(5,
9) "file? " File_Name$
 1110 IF File_Name$="" PROCBleeps(6,2)
 1120 UNTIL File_Name$>""
 1130 PRINT ''
 1140 PROCOSCLI("SAVE """+File_Name$+""" "+STR$~(B
uffer1)+" +3E8")
 1150 REM ** Buffer back to screen
 1160 FOR I%=0 TO 999 STEP 4:P%!I%=Buffer1!I%:NEXT
 1170 ENDPROC
 1180
 1190 REM ** Save the screen to PRINT   statements
 1200 DEF PROCProg_Save
 1210 Q%=PAGE
 1220 VDU23,1,0;0;0;0;
 1230 FOR Y%=0 TO 24
 1240 PROCFind_Qts
 1250 FOR X%=0 TO 39 STEP 4:Q%!X%=P%!(40*Y%+X%):NE
XT
 1260 Q%=Q%+40
 1270 NEXT Y%
 1280 VDU23,1,1;0;0;
 1290 ENDPROC
 1300
 1310 REM ** Reset the system
 1320 DEF PROCClear
 1330 CLS
 1340 Cursor_X%=0:Cursor_Y%=0
 1350 PROCClear_Flags
 1360 ENDPROC
 1370
 1380 DEF PROCClear_Flags
 1390 Double_Ht=FALSE
 1400 Insert=FALSE
 1410 Markers=0
 1420 Block_Marked=FALSE
 1430 ENDPROC
 1440
 1450 REM ** Set the markers
 1460 DEF PROCMarker
 1470 IF Markers=2 PROCBleeps(3,10):ENDPROC:REM **
 Already set?
 1480 IF Markers=0 Mark_X%(0)=Cursor_X%:Mark_Y%(0)
=Cursor_Y%:PROCBleeps(1,15):Markers=1:Block_Marked
=FALSE:ENDPROC
 1490 IF FNMark_Bad PROCBleeps(5,5):ENDPROC
 1500 Mark_X%(1)=Cursor_X%:Mark_Y%(1)=Cursor_Y%:PR
OCBleeps(2,15):Markers=2
 1510 Block_Marked=TRUE
 1520 ENDPROC
 1530
 1540 REM ** Check 2nd marker below and to right o
f first
 1550 DEF FNMark_Bad
 1560 Flag=Markers=1
 1570 Flag=Flag AND Cursor_X%>=Mark_X%(0)
 1580 Flag=Flag AND Cursor_Y%>=Mark_Y%(0)
 1590 =NOT Flag
 1600
 1610 REM ** Copy the marked block to a new pos'n
 1620 REM ** Cursor shows top left of   new pos'n
 1630 DEF PROCCopy
 1640 REM ** Check there's room
 1650 IF NOT FNRoom PROCBleeps(5,5):ENDPROC
 1660 REM ** Fill Buffer1 with marked   block
 1670 PROCFill_Buff(Buffer1,Mark_X%(0),Mark_Y%(0),
Mark_X%(1),Mark_Y%(1))
 1680 REM ** Write to new posn
 1690 PROCWrite_Buff(Buffer1,Cursor_X%,Cursor_Y%,M
ark_X%(0),Mark_Y%(0),Mark_X%(1),Mark_Y%(1))
 1700 Markers=0
 1710 ENDPROC
 1720
 1730 REM ** Swap 2 blocks - a marked   one and
```

```
 1740 REM ** one located by cursor
 1750 DEF PROCSwap
 1760 REM ** Check there's room
 1770 IF NOT FNRoom PROCBleeps(5,5):ENDPROC
 1780 REM ** Fill Buffer1 with marked   block
 1790 PROCFill_Buff(Buffer1,Mark_X%(0),Mark_Y%(0),
Mark_X%(1),Mark_Y%(1))
 1800 REM ** Fill Buffer2 with other   block
 1810 PROCFill_Buff(Buffer2,Cursor_X%,Cursor_Y%,Cu
rsor_X%+Mark_X%(1)-Mark_X%(0),Cursor_Y%+Mark_Y%(1)
-Mark_Y%(0))
 1820 REM ** Re-write first block in    place of s
econd
 1830 PROCWrite_Buff(Buffer1,Cursor_X%,Cursor_Y%,M
ark_X%(0),Mark_Y%(0),Mark_X%(1),Mark_Y%(1))
 1840 REM ** Overwrite first with second
 1850 PROCWrite_Buff(Buffer2,Mark_X%(0),Mark_Y%(0)
,Mark_X%(0),Mark_Y%(0),Mark_X%(1),Mark_Y%(1))
 1860 Markers=0
 1870 ENDPROC
 1880
 1890 REM ** Cursor control
 1900 DEF PROCLeft
 1910 Cursor_X%=Cursor_X%-1
 1920 IF Cursor_X%=-1 Cursor_X%=39:PROCUp
 1930 PRINT TAB(Cursor_X%,Cursor_Y%);
 1940 ENDPROC
 1950 DEF PROCRight
 1960 Cursor_X%=Cursor_X%+1
 1970 IF Cursor_X%=40 Cursor_X%=0:PROCDown
 1980 PRINT TAB(Cursor_X%,Cursor_Y%);
 1990 ENDPROC
 2000 DEF PROCDown
 2010 Cursor_Y%=Cursor_Y%+1
 2020 IF Cursor_Y%=25 Cursor_Y%=0
 2030 PRINT TAB(Cursor_X%,Cursor_Y%);
 2040 ENDPROC
 2050 DEF PROCUp
 2060 Cursor_Y%=Cursor_Y%-1
 2070 IF Cursor_Y%=-1 Cursor_Y%=24
 2080 PRINT TAB(Cursor_X%,Cursor_Y%);
 2090 ENDPROC
 2100
 2110 REM ** Delete - allow for INSERT  mode
 2120 DEF PROCDel
 2130 IF Cursor_X%=0 ENDPROC
 2140 IF Insert THEN PROCDel1 ELSE PROCDel2
 2150 ENDPROC
 2160
 2170 REM ** If INSERT, back up row
 2180 DEF PROCDel1
 2190 FOR X%=Cursor_X% TO 39
 2200 P%?(40*Cursor_Y%+X%-1)=P%?(40*Cursor_Y%+X%)
 2210 NEXT
 2220 PRINT TAB(39,Cursor_Y%) " ";
 2230 IF Double_Ht AND Cursor_Y%<24 PROCDel11
 2240 PROCLeft
 2250 ENDPROC
 2260
 2270 REM ** If DOUBLE HEIGHT flag set, handle the
 next row
 2280 DEF PROCDel11
 2290 FOR X%=Cursor_X% TO 39
 2300 P%?(40*Cursor_Y%+X%+39)=P%?(40*Cursor_Y%+X%+
40)
 2310 NEXT
 2320 PRINT TAB(39,Cursor_Y%+1) " ";
 2330 ENDPROC
 2340
 2350 REM ** Non-INSERT deletion
 2360 DEF PROCDel2
 2370 PROCLeft
 2380 Key=32:REM ** Use space to o'write
 2390 PROCChar
 2400 PROCLeft
 2410 ENDPROC
 2420
 2430 REM ** Read a file back from tape
 2440 DEF PROCTape_Read
 2450 CLS
 2460 INPUT TAB(5,8) "Read back which file to" TAB
(5,9) "the screen? " File_Name$
 2470 PRINT ''
 2480 PROCOSCLI("LOAD """+File_Name$+""" "+STR$~(B
uffer1))
 2490 REM ** Move from buffer to screen
 2500 FOR I%=0 TO 999 STEP 4:P%!I%=Buffer1!I%:NEXT
 2510 ENDPROC
 2520
 2530 REM ** Copy PRINT lines
 2540 REM ** to the screen
 2550 DEF PROCProg_Read
 2560 Q%=PAGE
 2570 VDU23,1,0;0;0;0;
 2580 FOR Y%=0 TO 24
 2590 PROCFind_Qts
 2600 FOR X%=0 TO 39 STEP 4:P%!(40*Y%+X%)=Q%!X%:NE
XT
 2610 Q%=Q%+40
 2620 NEXT Y%
 2630 VDU23,1,1;0;0;0;
 2640 ENDPROC
 2650
 2660 REM ** Check that Key is in the   defined ra
nge
 2670 DEF FNKey_In(lo,hi)
 2680 =(Key>=lo) AND (Key<=hi)
 2690
 2700 REM ** Shift along row if INSERT  mode
 2710 DEF PROCSpace(x%,y%)
 2720 Start%=P%+y%*40+x%
 2730 FOR I%=Start%+39-x% TO Start% STEP -1:?I%=I%
?-1:NEXT
 2740 ENDPROC
 2750
 2760 REM ** Sound Bleeps
 2770 DEF PROCBleeps(qty,length)
 2780 FOR i%=1 TO qty
 2790 SOUND 1,1,120,length
 2800 SOUND 2,1,121,length
 2810 NEXT
 2820 ENDPROC
 2830
 2840 REM ** Ensure enough room for move
 2850 REM ** not to go off screen edges
 2860 REM ** and that things are set formove
 2870 DEF FNRoom
 2880 Flag=Block_Marked
 2890 Flag=Flag AND Cursor_X%+Mark_X%(1)-Mark_X%(0
)<40
 2900 Flag=Flag AND Cursor_Y%+Mark_Y%(1)-Mark_Y%(0
)<25
 2910 =Flag
 2920
 2930 REM ** Fill a buffer
 2940 DEF PROCFill_Buff(buff_no,x1,y1,x2,y2)
 2950 Ptr%=0
 2960 FOR X%=x1 TO x2
 2970 FOR Y%=y1 TO y2
 2980 buff_no?Ptr%=P%?(40*Y%+X%)
 2990 Ptr%=Ptr%+1
 3000 NEXT
 3010 NEXT
 3020 ENDPROC
 3030
 3040 REM ** Write a buffer to screen
 3050 DEF PROCWrite_Buff(buff_no,x0,y0,x1,y1,x2,y2
)
 3060 Ptr%=0
 3070 FOR X%=x0 TO x0+x2-x1
 3080 FOR Y%=y0 TO y0+y2-y1
 3090 P%?(40*Y%+X%)=buff_no?Ptr%
 3100 Ptr%=Ptr%+1
 3110 NEXT
 3120 NEXT
 3130 ENDPROC
 3140
 3150 REM ** Find quotes at the start
 3160 REM ** of the next PRINT
 3170 DEF PROCFind_Qts
 3180 REPEAT
 3190 Q%=Q%+1
 3200 UNTIL ?Q%=ASC("""")
 3210 Q%=Q%+1:REM** First free space
 3220 ENDPROC
 3230
 3240 REM ** Send contents of "string$" to
 3250 REM ** Command Line Interpreter
 3260 DEF PROCOSCLI(string$)
 3270 $String_Buff=string$
 3280 X%=String_Buff MOD 256
 3290 Y%=String_Buff DIV 256
 3300 CALL &FFF7
 3310 ENDPROC
```





f1 — Load to PRINT Statements This key will translate the screen directly into PRINT statements which you can then incorporate into your own programs. For it to work, there MUST be 25 dummy PRINT lines at the start of the program, as shown in the listing. Each line must have precisely 40 spaces between the double quotes; after you use f1, these spaces will be filled with the characters on the screen and you can exit from the program and delete all the program except those lines. The function can be used as often as you like while SCRED7 is running — the PRINTs will always contain the last screen you saved.

f2 — Initialize pressing this key will clear the screen, put the cursor at top-left and set the starting conditions of 'Overwrite' and 'Single-height'.

f3 — Clear Flags Key f3 will set the program to use Overwrite and Single-Height input modes, and will clear the markers, but will not alter the display on the screen or move the cursor. It is particularly useful when (if?) you get confused, since it sets things back to a known state.

f4 — Insert Mode SCRED7's normal operating mode is 'Overwrite', in which anything you type overwrites whatever is on the screen at the cursor position. If, however, you select 'Insert' by pressing f4, the program will shift everything to the right of the cursor one space right each time you type a character. Anything which "falls off the end" will be lost — it does not wrap-round to the next line. If you are in Double-Height mode, both parts of the affected characters will be shifted.

f5 — Overwrite Mode Not surprisingly, key f5 will put the system back to its normal entry mode.

f6 — Set Marker The screen markers are used in conjunction with the block copy and exchange operations, and define an 'active block'. If no markers are set, pressing this key records the current cursor position, defining the top-left corner of a block on the screen. A single tone will sound. If one marker is already set, another press will set the cursor position as the block's bottom-right corner; two bleeps sound. The second corner MUST NOT be above or to the left of the first. If you make an error, or try to set more than two active markers, the key is ignored and you will hear multiple bleeps. The markers are phantoms — they do not actually appear on the screen.

f7 — Copy Once a block has been marked, you can move the cursor to almost any position and, by pressing key f7, make a copy of the block. The cursor defines the copy's top left corner. The only limitation is that there must be room to fit in the copy below and to the right of the cursor — the copy will not wrap-round the edges of the screen. Once a block has been marked, it can be copied as many times as you like without its needing to be redefined.

f8 — Swap Blocks This operation works in a very similar way to Copy, except that the marked and cursor-defined areas are exchanged completely. As before, there is no wrap-round.

SHIFT/f0 — Read from Tape or Disc If you press the SHIFT and f0 keys together, the program will read in a previously-saved copy of the screen and display it. In the normal BBC way, you will be asked for the title of the file you wish to load.

CTRL/f0 — Read from PRINT Statements If you press this key, the program will use whatever is held in the PRINT statements at the start of the program to form the screen display. If they are in their initial blank state, the screen will be cleared — otherwise you will restore the last screen saved via key f1.

DELETE The Delete key operates in its normal way, but with alterations to match the special needs of Mode 7. In the normal 'Overwrite' mode, the key will replace the character at its left with a blank and move left. Characters to the right will not be affected. If, however, you are in 'Insert' mode, the characters to its right will be dragged left with the cursor. If you delete in the middle of entering double-height characters, then both rows will be handled; this will not necessarily happen if you move the cursor from a 'single' to a 'double' area before deletion.

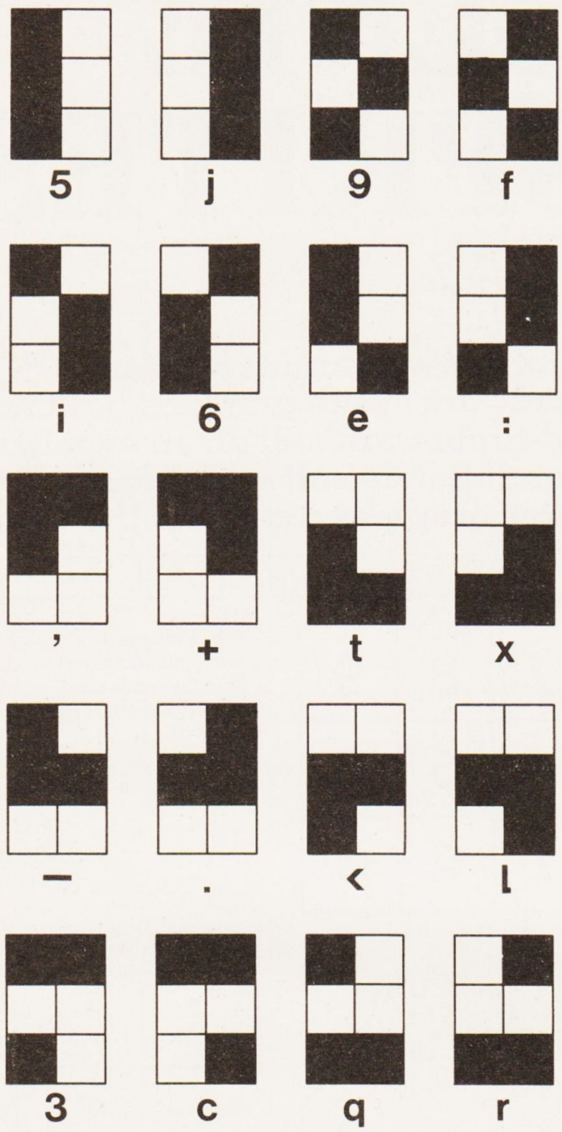
ESCAPE Pressing the ESCAPE key will shut the program down.

THE PROGRAM

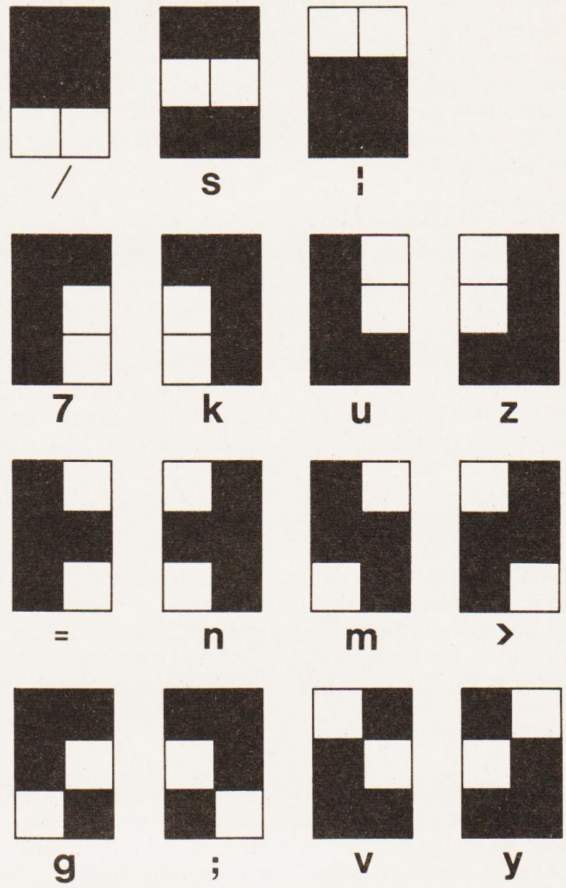
Listing 1 is a printout of SCRED7. At the start of the program is the area reserved for the dummy PRINT statements, which are skipped past by the GOTO at line 10. Make sure that the statements are in this place and that they are exactly right, or else the program is likely to get very confused.

Line 290 reserves space for

3



4



5

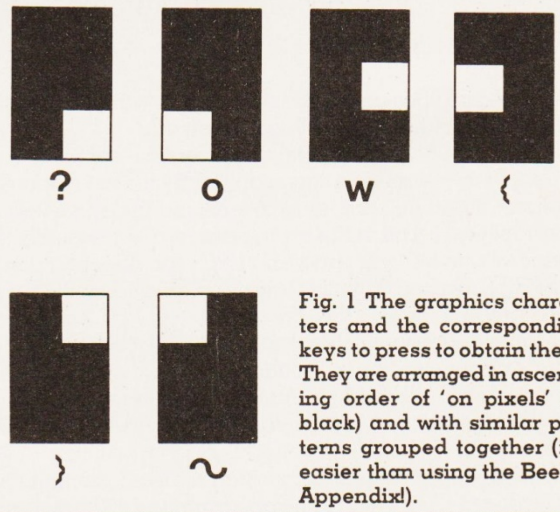


Fig. 1 The graphics characters and the corresponding keys to press to obtain them. They are arranged in ascending order of 'on pixels' (in black) and with similar patterns grouped together (it's easier than using the Beeb's Appendix!).

the program's arrays and, most importantly, the two buffers used by the copy, swap and I/O functions. PROCInit (lines 670-790) then sets up the system and allocates suitable codes to the function keys.

The heart of SCRED7 is the REPEAT . . . UNTIL loop at lines 310-580. This reads a key and, if necessary, decodes it to set a Teletext control code. In particular, line 340 converts the "SHIFT/CTRL/f" keys, which return values from 3 to 12, to the range of values needed for double-height, flash, new background, etc. The main loop then gives what amounts to a CASE function to select the action required of each keypress.

Each of the program's functions is served by one of many PROCedures which, together, form the bulk of the program. Each is, I hope fairly understandable and uses meaningful variable names. If you really wanted or needed to, you could cut down the size of SCRED7 quite substantially by shortening these names.

When displayable characters are typed into the program, the Beeb's normal PRINT TAB() statement is used to put them onto the screen. However, all the screen manipulation and storage routines address the screen memory directly. They use P%, which is set to &7C00, as a pointer to the start of the screen. This approach does mean that SCRED7 would not work down the Tube, but does contribute greatly to its speed of operation.

The tape and disc I/O routines use ★LOAD and ★SAVE commands to read and write the data to and from Buffer1. Lines 1140 and 2480 then invoke the computer's string functions and Command Line Interpreter to pass BASIC variables to the OS via PROCOSCLI at lines 3260-3310. If you have BASIC II, you can alter the PROCOSCLIs in lines 1140 and 2480 to OSCLI, and delete the PROCedure.

CONCLUSION

In this article, I have described a very useful Mode 7 screen editor. The program makes life much easier when it comes to designing a Teletext display, writing Mode 7 routines or simply exploring just how the Beeb's least-understood mode works.

The many function key operations may make the program look forbidding but, in practice, it is not. A few minutes playing with SCRED7 will show you that it really is very simple to use and give you an idea of just how powerful Mode 7 can be.

I hope that you find it useful.

GENIE SPRITES

Andrew Howard

Many new computers offer hardware sprite graphics but the old-timers can still do an excellent simulation in software. Here's some graphics magic for the Genie.

Sprite is a short, simple program that provides the facility for BASIC programs to move objects of any size around the screen at speed via five extra commands. It is not compatible with Level III BASIC or DOS.

The program loads at top of memory and automatically executes upon loading. It can be loaded with the system command or any other load routine: for details see Table 1. The ORG column shows the operand that should be on line 390 of the listing for the relevant memory size.

THE FACILITIES

Up to 20 sprites can be defined, each of any size. They can be moved around the screen in any direction by a single command.

● **KILL** This command resets all defined sprites to the start of the screen, sets the sprite size to one character by one character and clears the GET buffer. It is used at the start of programs before new sprites are defined. Format of KILL : KILL. For example:

10 CLEAR 50:KILL:DEFSTR A-F,P,R

● **FIELD** This command sets the sprite size and holds for all 20 possible sprites. They can be any size — from one character by one character to 64 by 16. Format of FIELD : FIELD(x,y) where x = length of the sprite (from 1 to 64) and y = height of the sprite (from 1 to 16). For example:

20 KILL:FIELD(17,5)

● **NAME** This command defines a sprite to be at a certain screen location. The position supplied is stored in the sprite buffer within the program and the position is updated every time that particular sprite moves. Note that after the KILL command, all 20 sprites are defined to be at 15360. Format of NAME : NAMEn,p where n = sprite number (from 1 to 20) and p = screen position (from 15360 to 16383). For example:

20 KILL:FIELD (6,6):NAME5,15570

The three previous commands were initialisation operations. The final two perform actual visible operations.

● **PUT** This command actually moves the sprites around the screen. Four parameters are supplied by the user: sprite number, direction, number of places and the erase flag. Consider Fig 2. This is the arrangement of keys usually found on a separate numeric keypad and forms the direction parameter for SPRITE. Taking 8 as up, 2 as down and so on then a single digit number provides the direction for the PUT command.

Figure 3, however, shows the movement each digit represents. For example, 9 causes a movement of up and right. Format of PUT : PUTn,d,t,s where n = sprite number (from 1 to 20), d = direction as explained above (1 to 9), t = number of places to move (from 1 onwards) and s = erase flag — s=0 if sprite is to be left at original position and s=1 if original sprite is to be erased before moving. For example:

40 PUT1,6,1,1

This command will move sprite number one to the right and erase the old image whereas:

320 PUT7,3,5,0

will move sprite number seven five places down and five places right and will not erase the old one.

Listing 1 is a demonstration program using the PUT command. The user types a key from 1 to 9 to move the block around — each time it moves three places in the specified direction. Note that if the t parameter is greater than one, the user will actually see the sprite move and not just disappear and reappear in its new place.

● **GET** This is a special command (totally separate from the

7	8	9
4	5	6
1	2	3

Fig. 1 Numeric keypad configuration — the numbers are used as the sprite direction parameter.

7 U/L	8 UP	9 U/R
4 LEFT	5 NONE	6 RIGHT
1 D/L	2 DOWN	3 D/R

Fig. 2 The movement resulting from each 'direction digit'.

```
1Ø CLS: PRINT CHR$(188) STRING$(15,14Ø) CHR$(188):
PRINT CHR$(191) "  Video Genie  " CHR$(191): PRINT
CHR$(191) "  Soft-Sprite  " CHR$(191): PRINT
STR$(17,131)

2Ø KILL: FIELD 17,5: NAME1,1536Ø

3Ø A$=INKEY$: IF A$="" THEN 3Ø ELSE A=VAL(A$): IF A=Ø
THEN 3Ø

4Ø PUT 1,A,3,1: GOTO 3Ø
```

Listing 1. Demonstration program using PUT.

```
1Ø CLS: PRINT CHR$(188) STRING$(15,14Ø) CHR$(188):
PRINT CHR$(191) "  Video Genie  " CHR$(191): PRINT
CHR$(191) "  Soft-Sprite  " CHR$(191): PRINT
STR$(17,131)

2Ø KILL: FIELD(17,5): P=1536Ø: GETP,Ø: CLS

3Ø FOR Y=1 TO 3: T=P: FOR X=1 TO 3: GETP,1: P=P+2Ø:
NEXT X: P=T+(5*64): NEXT Y

4Ø GOTO 4Ø
```

Listing 2. Sprite duplication program.

Listing 3. The assembler listing fro Genie Sprite.

```
3C00                        ORG   3C00H
3C00  4C                    DEFM  'LOADING S
PRITE....'
3C01  4F
3C02  41
3C03  44
3C04  49
3C05  4E
3C06  47
3C07  20
3C08  53
3C09  50
3C0A  52
3C0B  49
3C0C  54
3C0D  45
3C0E  2E
3C0F  2E
3C10  2E
3C11  2E
4016                        ORG   4016H
4016  00  70                DEFW  SPRITE
401E                        ORG   401EH
401E  00  70                DEFW  SPRITE
7000                        ORG   7000H
7000  21  1B  30    SPRITE  LD    HL,301BH
7003  22  16  40            LD    (4016H),HL
7006  21  58  04            LD    HL,458H
7009  22  1E  40            LD    (401EH),HL
700C  3E  C3                LD    A,195
700E  32  7F  41            LD    (417FH),A
7011  32  82  41            LD    (4182H),A
7014  32  8E  41            LD    (418EH),A
7017  32  91  41            LD    (4191H),A
701A  32  7C  41            LD    (417CH),A
701D  21  9B  F6            LD    HL,KILL
7020  22  92  41            LD    (4192H),HL
7023  21  9E  F6            LD    HL,NAME
7026  22  8F  41            LD    (418FH),HL
7029  21  B7  FF            LD    HL,GET
702C  22  80  41            LD    (4180H),HL
702F  21  DA  FE            LD    HL,PUT
7032  22  83  41            LD    (4183H),HL
7035  21  1D  F6            LD    HL,FIELD
7038  22  7D  41            LD    (417DH),HL
703B  CD  4A  1B            CALL  1B4AH
703E  CD  6E  F6            CALL  CLEAR
7041  21  4A  70            LD    HL,TITLE
7044  CD  75  2B            CALL  2B75H
7047  C3  72  00            JP    72H
704A  53            TITLE   DEFM  'SPRITE VI
DEO GENIE SOFT-SPRITES PROGRAM'
704B  50
704C  52
704D  49
704E  54
704F  45
704C  52
704D  49
704E  54
704F  45
7050  20
7051  56
7052  49
7053  44
7054  45
7055  4F
7056  20
7057  47
7058  45
7059  4E
705A  49
705B  45
705C  20
705D  53
705E  4F
705F  46
7060  54
7061  2D
7062  53
7063  50
7064  52
7065  49
7066  54
7067  45
7068  53
7069  20
706A  50
706B  52
706C  4F
706D  47
706E  52
706F  41
7070  4D
7071  0A                    DEFB  10
7072  56                    DEFM  'VERSION 1.
0 BY ANDREW HOWARD'
7073  45
7074  52
7075  53
7076  49
7077  4F
7078  4E
7079  20
707A  31
707B  2E
707C  30
707D  20
707E  42
707F  59
7080  20
7081  41
7082  4E
7083  44
7084  52
7085  45
7086  57
7087  20
7088  48
7089  4F
708A  57
708B  41
708C  52
708D  44
708E  0A                    DEFB  10
708F  28                    DEFM  '(C) APRIL
 1984.'
7090  43
7091  29
7092  20
7093  41
7094  50
7095  52
7096  49
7097  4C
7098  20
7099  31
709A  39
709B  38
709C  34
709D  2E
709E  0A  00                DEFW  10
F61D                        ORG   0F61DH
F61D  CF            FIELD   RST   8
F61E  28                    DEFM  '('
F61F  CD  02  2B            CALL  2B02H
F622  7B                    LD    A,E
F623  B7                    OR    A
F624  CA  4A  1E            JP    Z,1E4AH
F627  FE  41                CP    65
F629  D2  4A  1E            JP    NC,1E4AH
F62C  32  44  F6            LD    (XSIZE),A
F62F  CF                    RST   8
F630  2C                    DEFM  ,
F631  CD  02  2B            CALL  2B02H
F634  7B                    LD    A,E
F635  B7                    OR    A
F636  CA  4A  1E            JP    Z,1E4AH
F639  FE  11                CP    17
F63B  D2  4A  1E            JP    NC,1E4AH
F63E  32  45  F6            LD    (YSIZE),A
F641  CF                    RST   8
F642  29                    DEFM  ')'
F643  C9                    RET
F644  00            XSIZE   NOP
F645  00            YSIZE   NOP
F646                SPRTBL  DEFS  40
F66E  E5            CLEAR   PUSH  HL
F66F  21  CF  FA            LD    HL,BUFFER
F672  11  D0  FA            LD    DE,BUFFER+1
F675  01  FF  03            LD    BC,1023
F678  36  20                LD    (HL),32
F67A  ED  B0                LDIR
F67C  21  00  3C            LD    HL,15360
F67F  DD  21  46  F6        LD    IX,SPRTBL
F683  06  14                LD    B,20
F685  DD  75  00    RESCUR  LD    (IX+0),L
```

```
F688  DD  74  01              LD    (IX+1),H
F68B  DD  23                  INC   IX
F68D  DD  23                  INC   IX
F68F  10  F4                  DJNZ  RESCUR
F691  3E  01                  LD    A,1
F693  32  44  F6              LD    (XSIZE),A
F696  32  45  F6              LD    (YSIZE),A
F699  E1                      POP   HL
F69A  C9                      RET
F69B  C3  6E  F6      KILL    JP    CLEAR
F69E  CD  02  2B      NAME    CALL  2B02H
F6A1  7B                      LD    A,E
F6A2  B7                      OR    A
F6A3  CA  4A  1E              JP    Z,1E4AH
F6A6  FE  15                  CP    21
F6A8  D2  4A  1E              JP    NC,1E4AH
F6AB  3D                      DEC   A
F6AC  87                      ADD   A,A
F6AD  5F                      LD    E,A
F6AE  16  00                  LD    D,0
F6B0  E5                      PUSH  HL
F6B1  21  46  F6              LD    HL,SPRTBL
F6B4  19                      ADD   HL,DE
F6B5  E3                      EX    (SP),HL
F6B6  DD  E1                  POP   IX
F6B8  CF                      RST   8
F6B9  2C                      DEFM  ','
F6BA  CD  02  2B              CALL  2B02H
F6BD  7A                      LD    A,D
F6BE  FE  3C                  CP    3CH
F6C0  DA  4A  1E              JP    C,1E4AH
F6C3  FE  40                  CP    40H
F6C5  D2  4A  1E              JP    NC,1E4AH
F6C8  DD  73  00              LD    (IX+0),E
F6CB  DD  72  01              LD    (IX+1),D
F6CE  C9                      RET
F6CF                  TEMP    DEFS  1024
FACF                  BUFFER  DEFS  1024
FECF  3F              MOVTBL  DEFB  63
FED0  40                      DEFB  64
FED1  41                      DEFB  65
FED2  FF                      DEFB  -1
FED3  00                      NOP
FED4  01                      DEFB  1
FED5  BF                      DEFB  -65
FED6  C0                      DEFB  -64
FED7  C1                      DEFB  -63
FED8  00              COUNT   NOP
FED9  00              ERASE   NOP
FEDA  CD  02  2B      PUT     CALL  2B02H
FEDD  7B                      LD    A,E
FEDE  B7                      OR    A
FEDF  CA  4A  1E              JP    Z,1E4AH
FEE2  FE  15                  CP    21
FEE4  D2  4A  1E              JP    NC,1E4AH
FEE7  3D                      DEC   A
FEE8  87                      ADD   A,A
FEE9  5F                      LD    E,A
FEEA  16  00                  LD    D,0
FEEC  E5                      PUSH  HL
FEED  21  46  F6              LD    HL,SPRTBL
FEF0  19                      ADD   HL,DE
FEF1  E3                      EX    (SP),HL
FEF2  DD  E1                  POP   IX
FEF4  CF                      RST   8
FEF5  2C                      DEFM  ','
FEF6  CD  02  2B              CALL  2B02H
FEF9  7B                      LD    A,E
FEFA  B7                      OR    A
FEFB  CA  4A  1E              JP    Z,1E4AH
FEFE  FE  0A                  CP    10
FF00  D2  4A  1E              JP    NC,1E4AH
FF03  3D                      DEC   A
FF04  5F                      LD    E,A
FF05  16  00                  LD    D,0
FF07  E5                      PUSH  HL
FF08  21  CF  FE              LD    HL,MOVTBL
FF0B  19                      ADD   HL,DE
FF0C  E3                      EX    (SP),HL
FF0D  FD  E1                  POP   IY
FF0F  CF                      RST   8
FF10  2C                      DEFM  ','
FF11  CD  02  2B              CALL  2B02H
FF14  7B                      LD    A,E
FF15  B7                      OR    A
FF16  CA  4A  1E              JP    Z;1E4AH
FF19  32  D8  FE              LD    (COUNT),A
FF1C  CF                      RST   8
FF1D  2C                      DEFM  ','
FF1E  CD  02  2B              CALL  2B02H
```

```
FF21  7B                      LD    A,E
FF22  FE  02                  CP    2
FF24  D2  4A  1E              JP    NC,1E4AH
FF27  32  D9  FE              LD    (ERASE),A
FF2A  E5              PUTLP   PUSH  HL
FF2B  DD  6E  00              LD    L,(IX+0)
FF2E  DD  66  01              LD    H,(IX+1)
FF31  11  CF  F6              LD    DE,TEMP
FF34  3A  45  F6              LD    A,(YSIZE)
FF37  47                      LD    B,A
FF38  E5              STORE   PUSH  HL
FF39  C5                      PUSH  BC
FF3A  3A  44  F6              LD    A,(XSIZE)
FF3D  47                      LD    B,A
FF3E  CD  A8  FF      STORE2  CALL  PUT7
FF41  28  0A                  JR    Z,STORE3
FF43  7E                      LD    A,(HL)
FF44  12                      LD    (DE),A
FF45  3A  D9  FE              LD    A,(ERASE)
FF48  B7                      OR    A
FF49  28  02                  JR    Z,STORE3
FF4B  36  20                  LD    (HL),32
FF4D  23              STORE3  INC   HL
FF4E  13                      INC   DE
FF4F  10  ED                  DJNZ  STORE2
FF51  C1                      POP   BC
FF52  21  40  00              LD    HL,64
FF55  EB                      EX    DE,HL
FF56  E3                      EX    (SP),HL
FF57  19                      ADD   HL,DE
FF58  D1                      POP   DE
FF59  10  DD                  DJNZ  STORE
FF5B  E1                      POP   HL
FF5C  E5                      PUSH  HL
FF5D  DD  6E  00              LD    L,(IX+0)
FF60  DD  66  01              LD    H,(IX+1)
FF63  FD  7E  00              LD    A,(IY+0)
FF66  5F                      LD    E,A
FF67  16  00                  LD    D,0
FF69  B7                      OR    A
FF6A  F2  6F  FF              JP    P,PUT2
FF6D  16  FF                  LD    D,-1
FF6F  19              PUT2    ADD   HL,DE
FF70  DD  75  00              LD    (IX+0),L
FF73  DD  74  01              LD    (IX+1),H
FF76  11  CF  F6              LD    DE,TEMP
FF79  3A  45  F6              LD    A,(YSIZE)
FF7C  47                      LD    B,A
FF7D  E5              PUT3    PUSH  HL
FF7E  C5                      PUSH  BC
FF7F  3A  44  F6              LD    A,(XSIZE)
FF82  47                      LD    B,A
FF83  CD  A1  FF      PUT4    CALL  PUT5
FF86  23                      INC   HL
FF87  13                      INC   DE
FF88  10  F9                  DJNZ  PUT4
FF8A  C1                      POP   BC
FF8B  21  40  00              LD    HL,64
FF8E  EB                      EX    DE,HL
FF8F  E3                      EX    (SP),HL
FF90  19                      ADD   HL,DE
FF91  D1                      POP   DE
FF92  10  E9                  DJNZ  PUT3
FF94  E1                      POP   HL
FF95  3A  D8  FE              LD    A,(COUNT)
FF98  3D                      DEC   A
FF99  32  D8  FE              LD    (COUNT),A
FF9C  B7                      OR    A
FF9D  C2  2A  FF              JP    NZ,PUTLP
FFA0  C9                      RET
FFA1  CD  A8  FF      PUT5    CALL  PUT7
FFA4  C8                      RET   Z
FFA5  1A                      LD    A,(DE)
FFA6  77                      LD    (HL),A
FFA7  C9                      RET
FFA8  7C              PUT7    LD    A,H
FFA9  FE  3C                  CP    3CH
FFAB  38  08                  JR    C,PUT8
FFAD  FE  40                  CP    40H
FFAF  30  04                  JR    NC,PUT8
FFB1  3E  01                  LD    A,1
FFB3  B7                      OR    A
FFB4  C9                      RET
FFB5  AF              PUT8    XOR   A
FFB6  C9                      RET
FFB7  CD  02  2B      GET     CALL  2B02H
FFBA  EB                      EX    DE,HL
FFBB  CD  A8  FF              CALL  PUT7
FFBE  CA  4A  1E              JP    Z,1E4AH
FFC1  EB                      EX    DE,HL
```

```
FFC2  D5             Ø254Ø          PUSH  DE
FFC3  CF             Ø255Ø          RST   8
FFC4  2C             Ø256Ø          DEFM  ','
FFC5  CD  Ø2  2B     Ø257Ø          CALL  2BØ2H
FFC8  7B             Ø258Ø          LD    A,E
FFC9  FE  Ø2         Ø259Ø          CP    2
FFCB  D2  4A  1E     Ø26ØØ          JP    NC,1E4AH
FFCE  11  CF  FA     Ø261Ø          LD    DE,BUFFER
FFD1  E3             Ø262Ø          EX    (SP),HL
FFD2  32  D9  FE     Ø263Ø          LD    (ERASE),A
FFD5  3A  45  F6     Ø264Ø          LD    A,(YSIZE)
FFD8  47             Ø265Ø          LD    B,A
FFD9  E5             Ø266Ø GET2     PUSH  HL
FFDA  C5             Ø267Ø          PUSH  BC
FFDB  3A  44  F6     Ø268Ø          LD    A,(XSIZE)
FFDE  47             Ø269Ø          LD    B,A
FFDF  CD  A8  FF     Ø27ØØ GET3     CALL  PUT7
FFE2  28  ØC         Ø271Ø          JR    Z,GET5
FFE4  3A  D9  FE     Ø272Ø          LD    A,(ERASE)
FFE7  B7             Ø273Ø          OR    A
FFE8  28  Ø4         Ø274Ø          JR    Z,GET4
FFEA  1A             Ø275Ø          LD    A,(DE)
FFEB  77             Ø276Ø          LD    (HL),A
FFEC  18  Ø2         Ø277Ø          JR    GET5
FFEE  7E             Ø278Ø GET4     LD    A,(HL)
FFEF  12             Ø279Ø          LD    (DE),A
FFFØ  23             Ø28ØØ GET5     INC   HL
FFF1  13             Ø281Ø          INC   DE
FFF2  1Ø  EB         Ø282Ø          DJNZ  GET3
FFF4  C1             Ø283Ø          POP   BC
FFF5  21  4Ø  ØØ     Ø284Ø          LD    HL,64
FFF8  EB             Ø285Ø          EX    DE,HL
FFF9  E3             Ø286Ø          EX    (SP),HL
FFFA  19             Ø287Ø          ADD   HL,DE
FFFB  D1             Ø288Ø          POP   DE
FFFC  1Ø  DB         Ø289Ø          DJNZ  GET2
FFFE  E1             Ø29ØØ          POP   HL
FFFF  C9             Ø291Ø          RET
7ØØØ                 Ø292Ø          END   SPRITE
ØØØØØ  TOTAL ERRORS

BUFFER   FACF        PUT4     FF83
CLEAR    F66E        PUT5     FFA1
COUNT    FED8        PUT7     FFA8
ERASE    FED9        PUT8     FFB5
FIELD    F61D        PUTLP    FF2A
GET      FFB7        RESCUR   F685
GET2     FFD9        SPRITE   7ØØØ
GET3     FFDF        SPRTBL   F646
GET4     FFEE        STORE    FF38
GET5     FFFØ        STORE2   FF3E
KILL     F69B        STORE3   FF4D
MOVTBL   FECF        TEMP     F6CF
NAME     F69E        TITLE    7Ø4A
PUT      FEDA        XSIZE    F644
PUT2     FF6F        YSIZE    F645
PUT3     FF7D
```

SUMMARY OF COMMANDS

KILL	Reset all sprite variables
FIELD	Define sprite size
NAMEn,p	Define sprite n to be at position p
PUTn,d,t,s	Move sprite n direction d, t places: s=erase flag
GETp,m	Get from position p to buffer or vice versa according to m

TABLE 1

Capacity	Protect at	Start address	End address	Entry address	ORG operand
48K	63004	63005	65535	28672	0F61DH
32K	46620	46621	49152	28672	0B61DH
16K	30236	30237	32767	28672	0761DH

PUT command) in which the user supplies a screen position. According to a flag, a block of the current sprite dimensions can be read from the screen into a buffer, or the buffer can be read back onto the screen. Format of GET: GETp,m where p = screen position (from 15360 to 16383) and m = move flag — m = 0 reads the block from the screen to the buffer, m = 1 reads the block from the buffer to the screen. Suppose we wanted to make eight more copies of the sprite in the previous program: see Listing 2.

ACCESS TO THE SPRITE BUFFERS

The current sprite size can be obtained by:

PEEK (−2492) for the length x and
PEEK (−2491) for the height y

The sprite position table starts at −2490, the formula for accessing the position of sprite n is:

entry addr = −2490 + (2*(n−1))

For example, consider sprite number three. The buffer position is −2490 + (2*2) = −2486. Therefore, upon execution of:

X = PEEK (−2436)+PEEK (−2485)*256

X will contain the position on the screen of sprite number three. The GET buffer starts at −1329 and is a 1024 (1K) byte area of memory. The top left hand corner of the current sprite size starts at the first position in the buffer.

The MOVTBL table starts at −305 and contains nine bytes of information regarding the directions 1 to 9. For example, changing the 63 in −305 to 128 will case a '1' direction to move two lines straight down, as opposed to one line down and one character to the left.

PROGRAM NOTES

During loading, a message is displayed on the screen and the keyboard and screen vectors are changed to effect automatic program execution. The entry and initialisation routine SPRITE resets the I/O vectors and enforces the five commands. The NEW routine is called, then title control returns to the **READY** message. Note that the keyboard is reset to 301BH. For those without the extra ROMs in 3000H onwards, this should be changed to 3E3H.

The FIELD routine is self explanatory. The call to 2B02H evaluates the expression pointed to by HL and places the integer result in DE. The KILL routine again is self evident — it fills the GET buffer with spaces and resets all 20 sprites to 15360, the size being 1 by 1. The NAME routine first evaluates the sprite number and then calculates its entry in the position buffer. The position is then evaluated and if correct is then stored in the appropriate buffer. The PUT command first evaluates the sprite number, then calculates its entry address in the sprite position buffer. Second, the position is evaluated and the position in the move tabel MOVTBL calculated. Lastly, the count parameter is evaluated and stored and the erase value is evaluated and set.

The subroutine from line 1680 to 1960 reads the sprite from the screen and stores it temporarily in the PUT buffer, TEMP. Note that if any portion of the sprite is not within screen memory, it will not be read from/into the buffer. The old sprite is erased or not as it is read, according to the erase flag. The subroutine from lines 1970 to 2080 obtains the new address according to the direction and stores it in the sprite buffer, replacing the old address. Lines 2090 to 2270 contain the subroutine that stores the sprite on the screen at its new position, calling the subroutine PUT5 which only stores a portion of the sprite if it is within screen memory.

The GET command first evaluates the screen position, then the move indicator m which is stored in the erase variable, the subroutine from line 2640 to 2910 performs the actual move. Finally, the code in lines 2720 to 2790 performs the data transfer in the appropriate direction.

NEW TEXT FOR OLD

Peter Green

The main problem with Modes 2 and 5 on the BBC Micro and the Electron is that large amounts of text tend to be unreadable. Here's a proportional spacing routine that will change all that.

Owners of the BBC Micro and the Electron will know that they are brimful of graphics modes. These modes allow the user to decide on his priorities: lots of colours, high resolution or plenty of program space (memory being what it is the Acorn machines, ie sparse, you can't have all three at once).

You can have text written to the screen in any mode, and text characters are displayed by setting screen pixels to the foreground colour in the shape of the required leter — that is to say, the screen is bit-mapped. This doesn't apply to Mode 7 on the BBC, which works by storing the ASCII codes of the displayed characters in screen memory and using a special character generator to produce the screen output. This makes Mode 7 very memory-efficient — eight times more efficient than the nearest mode up — but less flexible, since the character definitions are fixed and inaccessible.

The character definitions for the other seven modes are stored as part of the ROM: eight bytes per character, since the characters are defined on an eight-by-eight grid and there are fixed characters occupy nearly three-quaters of a kilobyte in memory, so one set of definitions has to suffice for all types of mode.

In consequence, the physical shape of the letters on the screen depends on the screen resolution, which affects the shape of the pixels for the selected mode. So text appearance has to be a compromise. The design of the characters is optimised for 40-column display — whether over 25 or 32 lines — and is quite pleasant. In the two 80-column modes, there are twice as many pixels to the screen-width and hence twice as many characters can be fitted onto one line, but each character is still only eight pixels wide so the text appears a bit squashed, though still legible.

On the other hand, in 20-column modes there are only half the number of pixels across the screen relative to 40-column, so the letters are stretched horizontally. This renders them, in my opinion at least, almost unreadable, especially when there are several lines of text. So, is there anything we can do to improve the situation?

WEIGHT LOSS

Since the characters are bit-mapped, we can make them any shape we like by re-defining them using the VDU23 command. To get 40 characters per line in Modes 2 and 5, which are 160 pixels wide, we need to make each character four pixels wide, which means only three pixels when the gap between letters is taken into account. Letters like M and W cannot be made to look right in this width, so I decided to compromise and design each letter on a 'proportional spacing' basis. Each letter is made as wide as is necessary for legibility, from one pixel (plus a one-pixel gap) for the exclamation mark, to five pixels plus a gap for the likes of M and W.

If we print our new characters on to the screen using the conventional PRINT statement and the text cursor, each letter will still be spaced eight pixels apart, so the appearance of the letters will be improved but there will still be only 20 to the line and the spacing between them will look rather disjointed. The answer is print the text using proportional spacing, by linking the text and graphics cursors with the VDU5 command. After printing each character, the graphics cursor is moved along the line a distance depending on the width of the character just printed. Using this technique, an average of about 33 characters per line is obtained.

If a full eight bytes of data was used to define each new character, plus a byte for the width so that the cursor could be repositioned correctly, the data for each character would be nine bytes long. For a printing speed comparable to the normal PRINT statment, I wanted the routine to be in machine code, and it's much easier to multiply by eight in machine code (using three left shifts of the binary number) that to multiply by nine to obtain the correct position in the data table for each character. Since the bottom row of almost all the characters is blank, my redefined set has a zero byte for the bottom row of every character (so it can be left out of the data table): this only affects the comma and semicolon, which are moved up by one pixel to accommodate this system.

HOW IT WORKS

Listing 1 shows the setting up of the machine code and character data, the procedure that calls the assembled machine code and a short demonstration routine.

Let's consider the machine code part first. In this example 1000 bytes (more than we need, actually: this is the development program) is reserved by the DIM statement in line 5010. The first 472 locations contain the width and redefinition data bytes for the

Listing 1. The complete proportional text program, plus a sample output routine.

```
   10 REM ** PROPORTIONAL TEXT IN MODES 2 AND 5
   20 PROCsetup
   30 MODE5:VDU19,0,1;0;
   40 PROCprint(" !""#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ",256,256)
   50 PRINT" !""#$%&'()*+,-./0123456789:;<=>?@ABCD
EFGHIJKLMNOPQRSTUVWXYZ"
   60 PROCprint("THE QUICK BROWN FOX JUMPED OVER T
HE LAZY DOG'S BACK.",700,600)
   70 END
 4000 DEFPROCprint(string$,xstart%,ystart%)
 4005 VDU5
 4010 ?xcurs=xstart% MOD 256: xcurs?1=xstart% DIV
256
 4020 ?ycurs=ystart% MOD 256: ycurs?1=ystart% DIV
256
 4030 CALL code,string$
 4035 VDU4
 4040 ENDPROC
 5000 DEFPROCsetup
 5010 DIM Q% 1000
 5020 code=Q%+480: base=&75: ?base=Q% MOD 256: bas
e?1=Q% DIV 256
 5030 curmov=Q%+472: ?curmov=25: curmov?1=4: curmo
v?6=23: curmov?7=224
 5040 xcurs=curmov+2: ycurs=curmov+4
 5050 par=&600: block=&70: string=&72: length=&74:
 char=&77: temp=&79: oswrch=&FFEE
 5060 FOR I=0 TO 3 STEP 3
 5070 P%=code
 5080 [
 5090 OPT I
 5100 LDA par+1          \get parameters
 5110 STA block
 5120 LDA par+2
 5130 STA block+1
 5140 LDY #0
 5150 LDA (block),Y
 5160 STA string
 5170 INY
 5180 LDA (block),Y
 5190 STA string+1
 5200 INY
 5210 INY
 5220 LDA (block),Y
 5230 STA length
 5240 LDY #0
 5250 CPY length          \is string empty?
 5260 BNE start           \if not, start
 5265 JMP end             \else end
 5270 .start LDA base     \reset char,char+1
 5280 STA char            \to initial values
 5290 LDA base+1
 5300 STA char+1
 5310 LDA (string),Y      \get next character
 5320 SEC
 5330 SBC #32             \subtract 32 from ASCII
code
 5340 ASL A               \and multiply by 8
 5350 ASL A
 5360 ASL A
 5370 BCC nocarry
 5380 INC char+1          \add in carry if it exis
ts
 5390 CLC
 5400 .nocarry ADC char   \add offset to char
 5410 STA char
 5411 BCC nocarry1
 5412 INC char+1          \add in carry if it exis
ts
 5420 .nocarry1 STY temp  \save pointer to current
 character
 5430 LDY #0
 5440 LDA (char),Y        \get width of current ch
ar
 5450 ADC xcurs           \and add it to the x cur
sor position
 5460 BCC setcurs
 5470 LDA xcurs+1
 5480 CLC
 5490 ADC #1
 5500 CMP #5              \check if high byte of x
 cursor is 5
 5510 BNE setcurs         \if not, can print chara
cter
 5520 LDA #0
 5530 STA xcurs           \otherwise reset x curso
r
 5540 STA xcurs+1         \to left of screen
 5550 LDA ycurs
 5560 SEC
 5570 SBC #40             \and lower y cursor by 1
 line
 5580 STA ycurs
 5590 BCS setcurs
 5600 DEC ycurs+1
 5610 .setcurs LDY #0
 5620 .loop LDA curmov,Y  \move the graphics curso
r
 5630 JSR oswrch          \(and send start of VDU2
3 command)
 5640 INY
 5650 CPY #8
 5660 BNE loop
 5670 LDY #1
 5680 .read LDA (char),Y  \read in definition byte
s
 5690 JSR oswrch          \and send them out
 5700 INY
 5710 CPY #8
 5720 BNE read
 5730 LDA #0              \finish with a zero
 5740 JSR oswrch          \for all bottom rows
 5750 LDA #224            \and print the redefined
 character
 5760 JSR oswrch          \at the current graphics
 cursor
 5770 LDY #0
 5780 LDA (char),Y        \get width byte again
 5790 CLC
 5800 ADC xcurs           \and add it to the x cur
sor
 5810 STA xcurs           \(no need to check for r
ight
 5820 BCC nocarry2        \of screen, we already k
now it
 5830 INC xcurs+1         \must fit in)
 5840 .nocarry2 LDY temp  \get back pointer to str
ing position
 5850 INY                 \move pointer up one
 5860 CPY length          \end of string?
 5870 BEQ end             \end if yes
 5875 JMP start           \else loop back
 5880 .end RTS            \back to BASIC
```

59 characters that I dealt with (ASCII codes 32 to 90; space to capital Z), while the next locations contain the string of bytes for the graphics cursor shift and start of the VDU 23 command (ie 25, 4 (PLOT absolute), four bytes for the absolute X and Y coordinates to which the graphics cursor is to move, a 23 and a 224 (each character to be printed is redefined as character 224, then printed)). So the machine code proper starts at Q% + 480 ('code').

Various page zero addresses are required: base, the actual address in memory of the start of the data bytes; par, block, string and length, to retrieve the parameters passed by the machine code CALL and find the string in memory; char, the address of the data corresponding to the current character; and temp, a temporary storage location for the pointer to the current string character.

The machine code starts by moving the parameters from the BASIC workspace into page zero memory and then using them to locate the start of the string and its length. A check is made to see whether the string is empty, and if so, a jump is made to the end of the routine. Otherwise, the two-byte pointer char is reset to the start of the data block in memory (ie to the value of base) and the character to be printed is loaded into the accumulator using post-indexed indirect addressing. The ASCII code has 32 subtracted from it (we will not be printing any control characters with this routine) and the result is multiplied by 8 (using three left shifts) to give the offset of the required eight data bytes into the data table.

The largest number that can be in the accumulator is 58 (ASCII 90, capital Z, since I have not redefined the lower case letters), which is 00111010 in binary. Multiply-

```
5890 ]
5900 NEXT
5910 address=Q%
5920 REPEAT
5930 READ data
5940 IF data<>999 THEN ?address=data: address=add
ress+1
5950 UNTIL data=999
5960 ENDPROC
7032 DATA32,0,0,0,0,0,0,0
7033 DATA16,128,128,128,128,128,0,128
7034 DATA32,160,160,0,0,0,0,0
7035 DATA48,80,80,248,80,248,80,80
7036 DATA40,32,112,80,64,224,64,240
7037 DATA40,144,176,32,96,64,208,144
7038 DATA48,96,144,144,96,152,144,120
7039 DATA24,64,192,128,0,0,0,0
7040 DATA32,96,192,128,128,128,192,96
7041 DATA32,192,96,32,32,32,96,192
7042 DATA48,32,168,112,32,112,168,32
7043 DATA32,0,0,64,64,224,64,64
7044 DATA24,0,0,0,0,64,64,128
7045 DATA32,0,0,0,0,224,0,0
7046 DATA16,0,0,0,0,0,128,128
7047 DATA40,16,48,32,96,64,192,128
7048 DATA32,64,224,160,160,160,224,64
7049 DATA32,64,192,64,64,64,64,224
7050 DATA40,96,144,16,32,64,128,240
7051 DATA40,96,144,16,32,16,144,96
7052 DATA40,32,96,96,160,240,32,32
7053 DATA40,240,128,224,16,16,144,96
7054 DATA40,112,192,192,240,144,240,96
7055 DATA40,240,16,16,32,32,64,64
7056 DATA40,96,144,144,96,144,144,96
7057 DATA40,96,240,144,144,112,48,224
7058 DATA16,0,0,128,128,0,128,128
7059 DATA24,0,64,64,0,64,64,128
7060 DATA40,16,32,64,128,64,32,16
7061 DATA32,0,0,224,0,224,0,0
7062 DATA40,128,64,32,16,32,64,128
7063 DATA32,64,160,32,96,64,0,64
7064 DATA40,96,144,176,176,176,128,112
7065 DATA32,64,224,160,160,224,160,160
7066 DATA40,224,176,176,224,176,176,224
7067 DATA48,112,216,128,128,128,216,112
7068 DATA40,224,176,144,144,144,176,224
7069 DATA40,240,128,128,224,128,128,240
7070 DATA40,240,128,128,224,128,128,128
7071 DATA48,112,216,128,184,136,216,112
7072 DATA32,160,160,160,224,160,160,160
7073 DATA32,224,64,64,64,64,64,224
7074 DATA40,112,32,32,32,160,224,64
7075 DATA40,144,176,224,192,224,176,144
7076 DATA32,128,128,128,128,128,128,224
7077 DATA48,216,248,168,168,136,136,136
7078 DATA40,144,144,208,240,176,144,144
7079 DATA40,96,240,144,144,144,240,96
7080 DATA40,224,176,176,224,128,128,128
7081 DATA40,96,144,144,144,176,160,112
7082 DATA40,224,176,176,224,160,176,144
7083 DATA48,112,216,192,112,24,216,112
7084 DATA32,224,64,64,64,64,64,64
7085 DATA32,160,160,160,160,160,160,224
7086 DATA32,160,160,160,160,160,224,64
7087 DATA48,136,136,136,136,168,248,80
7088 DATA32,160,160,64,64,64,160,160
7089 DATA32,160,160,160,224,64,64,64
7090 DATA32,224,32,32,64,128,128,224
7091 DATA999
```

ing by 8 means that we only have to check, after the third shift, whether the carry bit has become set, and if so add it to the high byte of the two-byte address char. Then we add what's left in the accumulator to the low byte of char, again checking to see if there is a carry and incrementing the high byte if necessary.

Now we are going to need to use the Y register for more post-indexed indirect addressing, so the current value of the pointer to the position we have reached in the string is saved in location temp (a string cannot be longer than 256 characters on the BBC, so a single byte suffices). Now that char, char + 1 contain the start in memory of the eight data bytes required, we can load the width of the current character into the accumulator (line 5440) and add it to the current x postion of the graphics cursor. (Note that this addition is done in the accumulator: the actual memory locations remain unaltered). The BBC graphics modes treat the screen as being 1280 points wide, which is &0500 in hex, so we can simply check whether the character will fit onto the current line by checking if the high byte of the x cursor has reached 5. If it has, lines 5520-5540 set the x cursor to zero, that is the extreme left-hand edge of the screen, and lines 5550-5600 move the y cursor down the screen to the next line (graphics coordinates increase *up* the screen, so to move down we subtract).

Notice that the value subtracted in line 5570 is larger than you might think correct. However, although the screen is actually 160 by 256 pixels, the graphics system treats the screen as 1024 by 1280, so all pixel offsets have to be multiplied by 8 horizontally and 4 vertically to achieve the correct results on-screen. This is why the first number in each data statement (lines 7032-7090) is eight times the actual width: the remaining numbers in each line are the graphics data.

Next the graphics cursor is set to the correct position to print the next character. Lines 5620-5660 use the operating system call OSWRCH to send the sequence of eight bytes starting at curmov: 25,4 to indicate 'absolute move', the four bytes containing the x and y cursor positions, and a 23,224 to begin the character redefinition. Then the remaining seven bytes of data for the character in question are sent out by lines 5680-5720, followed by a final zero in lines 5730-5740 which reresents the bottom row of all new characters. Then the character 224 is printed.

Finally lines 5770-5830 get the width of the current character again and add it to the x cursor, this time storing it back in the correct memory locations so that the next letter will be printed in the right place. The pointer to the string is recalled from temp, incremented, and if the end of the string hasn't been reached, the program loops back for the next character.

The 'leapfrogging' jumps at the start and end of the program (5260-5255 and 5870-5875) are not very elegant but as the main body of the machine code is longer than 128 bytes when assembled, relative branching is unfortunately impossible.

Once the code is assembled, the data is POKEd into the memory block of 472 bytes left reserved for it. In fact the machine code itself only occupies 170 bytes, so the DIM statement in line 5010 could be altered to DIM Q% 642 to save memory.

USING THE CODE

A procedure called PROCprint is used to implement the new routine: the parameters to be passed are the string to be printed, and the x and y coordinates at which you want the first letter to appear. The procedure links the text and graphics cursors with VDU 5, loads the x and y positions into the correct curmov locations as low-byte, high-byte pairs (lines 4010-4020), then CALLs the machine code. When the string has been printed, the cursors are set to normal with a VDU 4.

Note that this routine has certain limitations. Like the ordinary PRINT routine, it will not prevent words from breaking at the end of a line (although letters will not be broken). If you want to rewrite the program for word-wrapping, the techniques were covered in an article in the May issue of Computing Today.

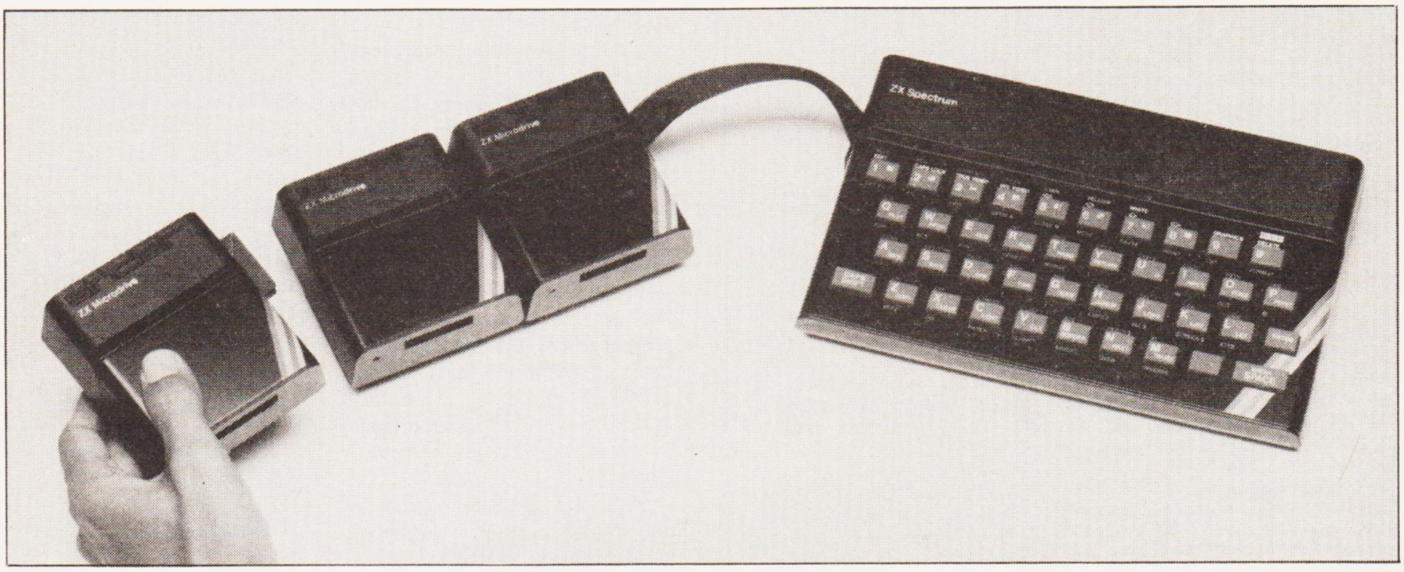
Because the text is being written at the graphics cursor, the graphics colour rules set by any previous GCOL statement will be obeyed. Text will be superimposed on anything currently at the printing location, so you'll have to clear the required area first. The screen will not scroll if you try to write over the bottom: text will simply disappear into the nether world of negative coordinates.

For reasons of space I've only redefined letters up to Z but the adventurous among you could try redefining the lower-case letters too. To reduce the memory overhead, which is considerable, you could assemble the code and data, save it as a block of machine code and load it in as convenient for use by other programs, thus saving the space occupied by the assembler portion and the BASIC DATA statements: a considerable saving.

MICRODRIVE FILE LINE EDITOR

W. F. Barnard

With the eventual availability of the ZX Microdrives, who needs a mainframe? Well, maybe life isn't quite that simple . . .



The ZX microdrives give the home micro user the ability to create his own files just like those on a mainframe computer. This program allows the user to create and edit these files.

Some of the terms used will be familiar to ICL users. The only failing with the Microdrive facility is that a program will crash if it tries to read off the end of a file. This editor, when creating a file will always put a final line in the file of four stars (★★★★), enabling the editor to recognise the end of a file.

The editor works by reading the file you wish to edit, the input file, one line at a time and allows you to modify each line, if required, before writing the line to an output file. If no input file is specified, then the editor automatically goes into input mode. Input mode allows you to create an output file containing as many lines as you wish. You terminate input mode by typing four stars as the final line in the file, ie ★★★★.

There is a facility to merge one other file (a MERGE file) with the current input file and another facility to allow editing instructions to be read from a file (a USE file) instead of from the keyboard.

Each line read from the input file is displayed on the screen in the form:

Line number(Line length)__ the actual line.
eg 1(9)__Hello mum

The command line that you type in must not contain leading or trailing spaces as the program is not designed to ignore these. The edit commands, which may be typed in upper or lower case are:

- C__ Close the current input file (may be the MERGE file) and reopen it at the beginning (eg C).
- E__ End the edit by Transcribing all the lines in the input file(s) to the output file and close them (eg E).
- I__ Insert the string between the delimiters in front of the current line or go into input mode (eg I/hello/will insert the line 'hello' into the output file. IN will make the editor go into input mode, ie every line typed in will be written to the output file. Input mode is terminated with ★★★★).
- M__ Open a second file for editing. This file is closed by the X command (eg M3fred will open file 'fred' on Microdrive 3).
- P__ Move the Pointer over so many lines, ie read lines from the input file and do not write them to the output file. This in effect deletes lines. The command will stop itself if the ★★★★ at the end of the input file is read (eg P4 will skip over four lines. P/The/ will skip over lines until it finds a line beginning with the characters 'The'. PC/and/ will skip over lines until it finds a line containing the characters 'and').
- Q__ Quit the edit. This will close all the files and then erase the output file (eg Q).
- R__ Replace a 'find' string with a 'replace' string in the current line (eg R/the/there/ replaces the characters 'the' with the characters 'there' in the current line).
- T__ Transcribe (ie copy) so many lines from the input file to the output file. This command works with the same parameters as the P command explained above. Like the P command, the T command will stop if the file terminator ★★★★ is read from the input file (eg T3 will copy 3 lines of the input file to the output file. T/We/will copy lines up to the line beginning with the characters 'We'. TC/the/ will copy lines up to the line containing the characters 'the').
- U__ Use a file for further editing instructions. The USE file should contain the Z command to get further commands from the keyboard (eg U2fred will open the file 'fred' on Microdrive 2 and execute the editing commands contained in it).
- X__ Close the MERGE file (eg X). X will close the MERGE file, if one is open, and display the current line from the original input file.)
- Z__ Close the USE file if one is open and go back to the keyboard for further editing commands (eg Z).
- H__ Help. Print a list of editing command instructions.
- CAT__ The keyword CAT (ie extended mode, symbol shift 9) will show the catalogue of a Microdrive (eg CAT2 will catalogue Microdrive 2).
- ★__ An emergency command in case a USE file executes the ★★★★ file terminator.

Example 2 : EDITING THE NEW FILE

RUN the editor program. Type ENTER to the first question. To the next two questions, type the filename and then the Microdrive number of the file that you wish to create. Type each line followed each time by ENTER. Type ★★★★ to end input mode and close your new file. You can check that your file has been created by using the MOVE command, ie to list your file called fred on Microdrive 1 on the screen type:

MOVE "m";1;"fred" TO #2

The following test file, call it fred on Microdrive 1, may be created for editing later in Example 2:

The first line of the file.
The second line of the file.
This id the thrid lin wiv erros init.
This is the last line.
★★★★

Example 1 : CREATING A NEW FILE

RUN the editor program. Type the following answers to the first four questions, comments are in braces{}:

fred	{The input file}
1	{Its Microdrive number}
fred2	{The output file, you can't use the name fred unless it's on another Microdrive}
1	{The same Microdrive}

Type in the following commands:

T1	{Transcribe one line}
P1	{Move the Pointer to the next line}
R/d/s/	{Replace 'd' with 's' so that 'id' becomes 'is'}
R/ri/ir/	{Replace 'ri' with 'ir' so 'thrid' becomes 'third'}
R/in/ine/	{Replace 'in' with 'ine' so 'lin' becomes 'line'}
R/v/th/	{Replace 'v' with 'th' so 'wiv' becomes with'}
R/erros/no errors/	{As above}
R/init/in it/	
T1	{Transcribe one line}
C	{Close and reopen the input file at the beginning}
T2	{Transcribe two lines}
P1	{Move the Pointer to the next line}
M1 fed	{Merge the file 'fred' on Microdrive 1, you can merge the same file like this if you want}
P2	{Move the Pointer over two lines}
R/This id//	{Replace 'This id' with nothing, ie delete the text}
T1	{Transcribe one line}
X	{Close the MERGE file}
E	{Copy the rest of the input file to the output file and close the files}

Type the following command:

MOVE "m";1;"fred2" TO #2

The resulting output file should look like this:

The first line of the file.
This is the third line with no errors in it.
The first line of the file.
The second line of the file.
the thrid lin wiv erros init.
This is the last line.
★★★★

Example 3 : THE USE FILE

Suppose that you have a large file and you wish to correct one spelling mistake in line 134 and then end the edit. Create a file, as in Example 1, called 'use' containing the following edit commands:

T133	{Transcribe down to line 134}
R/Fred/Freda/	{Correct the spelling mistake}
E	{End the edit by transcribing the rest of your file to the new version}
★★★★	{End input mode when creating 'use'

RUN the editor again, as in Example 2, and when you are asked for a command type:

U1 use

This will take the editing instructions from the file 'use' on Microdrive 1. If you had a large file which takes a while to edit and transcribe all the lines, the USE file will allow you to edit the file while you are having coffee or making a phone call.

Listing 1 Complete program for Microdrive File Line Editor.

```
10 REM **************************
11 REM * Microdrive File Editor *
12 REM *    W.F.Barnard B.Sc.   *
13 REM *       April 1984       *
14 REM **************************
15
20 GO SUB 100: REM init
25 GO SUB start
30 GO SUB openfiles
35 IF inputonly THEN GO SUB input: GO TO 65
40
45 GO SUB getcommand
50 GO SUB command
55 IF NOT end THEN GO TO 45
60
65 GO SUB closefiles
70 GO SUB synopsis
75 GO TO 9999
99
100 REM **************
101 REM * Initialise *
102 REM **************
103
110 LET a$="": REM command string
115 LET c$="": REM current line
120 LET f$="": REM find string
125 LET r$="": REM replace string
130 LET i$="": REM insert string
135 LET t$="": LET s$="": REM temp stores for c$
140 LET d$="Command_": REM command display string
145 LET m$="": REM string delimiter
150 LET nocoms=14: REM no. commands
155 DIM z$(nocoms,6): REM commands + line no.s
160 RESTORE 160
175 FOR i=1 TO nocoms
180  READ z$(i)
185 NEXT i
190 DATA "Cc1300","Ee1400"
191 DATA "Ii1500","Mm1600"
192 DATA "Pp1700","Qq2000"
193 DATA "Rr2100","Tt2300"
194 DATA "Uu2600","Xx2700"
195 DATA "Zz2800","Hh3100"
196 DATA " CAT CAT 2900","**3000"
200 DIM b$(4,11): REM microdrive no.s + filenames
205 DIM l(3): REM linecounts for each file
206 LET index=1: REM index to linecounts,1=i/p file,2=merge file
210 LET comstrm=0: REM command stream(0=kbd,6=USE file)
215 LET instrm=4: REM input file stream(5=merge file)
220 LET start=500
225 LET openfiles=700
230 LET input=800
232 LET putline=850
235 LET getline=900
237 LET printc=940
240 LET getcommand=1000
245 LET illegal=1200
250 LET tooshort=1250
255 LET delimerr=1280
257 LET findf=1950
260 LET closefiles=3400
262 LET endfile=3460
265 LET synopsis=3500
266 LET bp1=.1: REM Beep parameters
267 LET bp2=20
270 LET TRUE=1: REM boolean values
275 LET FALSE=0
280 LET inputonly=FALSE
285 LET end=FALSE
290 LET merge=FALSE
300 LET use=FALSE
310 LET found=FALSE
400 RETURN
498
499 REM start:
500 REM *********
501 REM * Start *
502 REM *********
503
510 CLEAR #: CLS #
515 FOR i=1 TO 5: PRINT PAPER 1;"                                ": NEXT i
520 PRINT INK 7; PAPER 2;AT 1,5;"Microdrive File Editor";AT 2,8;"W.F.Barnard B.
Sc.";AT 3,11;"April 1984"
521
525 PRINT '''"What is the name of the file to be edited? (just press ENTER fori
nput mode only)"
527 BEEP bp1,bp2
530 INPUT LINE a$: LET lena=LEN a$
535 IF lena>10 THEN GO TO 525
540 IF lena=0 THEN LET inputonly=TRUE: GO TO 565
541
545 PRINT '"Which microdrive is it on (1-8)?"
547 BEEP bp1,bp2
550 INPUT num
555 IF num<1 OR num>8 THEN GO TO 545
560 LET b$(1)=STR$ num+a$
561
565 PRINT '"What is the name of your output file?"
567 BEEP bp1,bp2
570 INPUT LINE a$: LET lena=LEN a$
575 IF lena=0 OR lena>10 THEN GO TO 565
576
580 PRINT '"Which microdrive is it on (1-8)?"
582 BEEP bp1,bp2
585 INPUT num
590 IF num<1 OR num>8 THEN GO TO 580
591
595 LET b$(2)=STR$ num+a$
600 IF b$(1)=b$(2) THEN PRINT '"Output file same as input file": GO TO 565
605 PRINT
610 RETURN
698
699 REM openfiles:
700 REM **************
701 REM * Open files *
702 REM **************
703
705 PRINT '"Opening File(s)"
710 IF NOT inputonly THEN OPEN #instrm;"m";VAL b$(1,1);b$(1,2 TO ): GO SUB getl
ine
720 OPEN #7;"m";VAL b$(2,1);b$(2,2 TO )
730 RETURN
798
799 REM input:
800 REM **************
801 REM * Input mode *
802 REM **************
803
805 PRINT '"Input Mode (Terminate with ****)"''
810 BEEP bp1,bp2: INPUT #comstrm; LINE c$
815 POKE 23692,0: PRINT c$
820 IF c$="****" THEN RETURN
830 GO SUB putline
840 GO TO 810
848
849 REM putline:
850 REM *************************
851 REM * Put c$ to output file *
852 REM *************************
853
860 PRINT #7;c$
870 LET l(2)=l(2)+1
880 RETURN
898
899 REM getline:
900 REM **************************************
901 REM * Get current line from input stream *
902 REM **************************************
903
920 INPUT #instrm; LINE c$
930 LET l(index)=l(index)+1
931
935 REM printc:
940 LET lenc=LEN c$
950 POKE 23692,0
960 PRINT 'l(index);"(";lenc;")_";c$
970 RETURN
998
999 REM getcommand:
1000 REM ***********************************
1001 REM * Get command from command stream *
1002 REM ***********************************
1003
1010 IF NOT use THEN BEEP bp1,bp2
1015 INPUT #comstrm;(d$); LINE a$
1020 LET lena=LEN a$
1035 IF lena=0 THEN GO TO 1010
1036
1040 POKE 23692,0
1045 PRINT '"Command_";a$
1050 REM find command
1055 FOR i=1 TO nocoms
1060  IF a$(1)=z$(i,1) OR a$(1)=z$(i,2) THEN LET command=VAL z$(i,3 TO ): RETURN

1065 NEXT i
1070 LET command=illegal
1075 RETURN
1198
1199 REM illegal:
1200 REM *******************
1201 REM * Illegal command *
1202 REM *******************
1203
1210 POKE 23692,0
1215 BEEP bp1+bp1,bp2
1220 PRINT FLASH 1;'"ILLEGAL"
1225 BEEP bp1+bp1,bp2
1230 RETURN
1231
1240 REM tooshort:
1250 POKE 23692,0
1260 PRINT INVERSE 1;'"Command too short"
1270 GO TO 1220
1271
1275 REM delimerr:
1280 POKE 23692,0
1290 PRINT INVERSE 1;'"Delimiters do not match"
1295 GO TO 1220
1298
1299
1300 REM *************
1301 REM * C command *
1302 REM *************
1303 REM * Format C  ********************
1304 REM * Close current input file and *
1305 REM * reopen at beginning of file  *
1306 REM ********************************
1307
1310 CLOSE #instrm
1330 OPEN #instrm;"m";VAL b$(index,1);b$(index,2 TO )
1340 LET l(index)=0
1350 GO SUB getline
1360 RETURN
1399
1400 REM *************
1401 REM * E command *
1402 REM *************
1403 REM * Format E  ************************
1404 REM * Transcribe rest of input file(s) *
1405 REM * to output file and signal end.   *
1406 REM ************************************
1407
1410 IF c$="****" THEN GO TO 1440
1420 GO SUB putline
1430 GO SUB getline
1435 GO TO 1410
1436
1440 IF NOT merge THEN LET end=TRUE: RETURN
1445
1450 GO SUB 2730: REM X command
1460 GO TO 1410
1499
1500 REM *************
1501 REM * I command *
1502 REM ***************************
1503 REM * Format I/string/ or IN ********************
1504 REM * Insert string in front of current line or *
1505 REM * go into input mode.                       *
1506 REM *******************************************
1507
1510>IF a$="IN" OR a$="in" THEN LET s$=c$: GO SUB input: LET c$=s$: GO TO printc
1515 IF lena<3 THEN GO TO tooshort
1520 IF lena=3 THEN LET i$="": GO TO 1530
1525 LET i$=a$(3 TO lena-1)
1530 IF a$(2)<>a$(lena) THEN GO TO delimerr
1540 LET s$=c$
1550 LET c$=i$
1560 GO SUB putline
1570 LET c$=s$
1580 RETURN
1599
1600 REM *************
1601 REM * M command *
1602 REM *****************
1603 REM * Format M2fred *************************
1604 REM * Open file called fred on microdrive 2 *
1605 REM * for input. Use X command to close it. *
1606 REM ****************************************
1607
1610 IF merge THEN PRINT '"Merge file already open": GO TO illegal
1615 IF lena=1 THEN GO TO illegal
1620 IF a$(2)<"0" OR a$(2)>"8" THEN PRINT '"Microdrive number missing": GO TO il
legal
1630 LET b$(3)=a$(2 TO )
1635 LET instrm=5
1640 OPEN #instrm;"m";VAL b$(3,1);b$(3,2 TO )
1645 LET index=3
1646 LET l(3)=0
1650 LET merge=TRUE
1660 LET t$=c$: REM save current line
1670 GO SUB getline
1680 RETURN
1699
1700 REM *************
1701 REM * P command *
1702 REM ******************************************
1703 REM * Format P4 or P/string/ or PC/string/ **********
1704 REM * Read & ignore so many lines from input stream *
1705 REM ***********************************************
```

```
1706
1710 IF lena=1 THEN GO TO illegal
1720 IF a$(2)="C" OR a$(2)="c" THEN GO TO 1850
1730 IF a$(2)<"0" OR a$(2)>"9" THEN GO TO 1790
1740 FOR i=1 TO VAL a$(2 TO )
1750  IF c$="****" THEN GO TO endfile
1760  GO SUB getline
1770 NEXT i
1780 RETURN
1785
1786 REM find line beginning with f$
1790 IF lena<4 THEN GO TO tooshort
1795 LET f$=a$(3 TO lena-1): LET lenf=LEN f$
1800 IF a$(2)<>a$(lena) THEN GO TO delimerr
1801
1810 IF c$="****" THEN GO TO endfile
1820 GO SUB findf: IF found AND pos=1 THEN RETURN
1830 GO SUB getline
1840 GO TO 1810
1841
1845 REM find line containing f$
1850 IF lena<5 THEN GO TO tooshort
1860 LET f$=a$(4 TO lena-1): LET lenf=LEN f$
1870 IF a$(3)<>a$(lena) THEN GO TO delimerr
1871
1880 IF c$="****" THEN GO TO endfile
1890 GO SUB findf
1900 IF found THEN RETURN
1910 GO SUB getline
1920 GO TO 1880
1948
1949 REM findf:
1950 REM *****************
1951 REM * Find f$ in c$ *
1952 REM *****************
1953
1960 LET found=FALSE
1970 FOR i=1 TO lenc-lenf+1
1980  IF f$=c$(i TO i+lenf-1) THEN LET pos=i: LET found=TRUE: RETURN
1985 NEXT i: LET pos=0
1990 RETURN
1999
2000 REM *************
2001 REM * Q command *
2002 REM *************
2003 REM * Format Q  ************************************
2004 REM * Quit edit, close files & erase output file *
2005 REM ********************************************
2006
2010 GO SUB closefiles
2020 PRINT '"Erasing Output File"
2030 ERASE "m";VAL b$(2,1);b$(2,2 TO )
2040 LET end=TRUE
2050 GO TO 9999
2099
2100 REM *************
2101 REM * R command *
2102 REM ***************************
2103 REM * Format R/find/replace/ *****************
2104 REM * Replace find string with replace string *
2105 REM * in current line.                        *
2106 REM * Replace string may be blank.            *
2107 REM *****************************************
2108
2109 IF c$="****" THEN GO TO endfile
2110 IF lena<5 THEN GO TO tooshort
2115 LET m$=a$(2)
2120 IF m$<>a$(lena) THEN GO TO delimerr
2130 LET a$=a$(3 TO lena-1): LET lena=LEN a$
2135 REM find middle delimiter
2140 FOR i=1 TO lena
2150  IF a$(i)=m$ THEN GO TO 2180
2160 NEXT i
2170 GO TO delimerr
2175
2180 LET f$=a$( TO i): LET lenf=LEN f$
2190 LET r$=a$(i TO ): LET lenr=LEN r$
2200 IF lenf=1 THEN PRINT '"Empty find string": GO TO illegal
2210 LET f$=f$( TO lenf-1): LET lenf=LEN f$
2220 IF lenr=1 THEN LET r$="": LET lenr=0: GO TO 2230
2225 LET r$=r$(2 TO ): LET lenr=LEN r$
2230 GO SUB findf
2240 IF NOT found THEN PRINT '"String ";f$;" not found": GO TO illegal
2250 IF lenf=lenc THEN LET c$=r$: GO TO printc
2260 IF pos=1 THEN LET c$=r$+c$(pos+lenf TO ): GO TO printc
2270 IF pos=lenc-lenf+1 THEN LET c$=c$( TO pos-1)+r$: GO TO printc
2280 LET c$=c$( TO pos-1)+r$+c$(pos+lenf TO )
2290 GO TO printc
2299
2300 REM *************
2301 REM * T command *
2302 REM ****************************************
2303 REM * Format T5 or T/string/ or TC/string/ *
2304 REM * Transcribe so many lines from input  *
2305 REM * stream to output stream              *
2306 REM ****************************************
2307
2310 IF lena=1 THEN GO TO tooshort
2320 IF a$(2)="C" OR a$(2)="c" THEN GO TO 2480
2330 IF a$(2)<"0" OR a$(2)>"9" THEN GO TO 2400
2335 REM transcribe n lines
2340 FOR i=1 TO VAL a$(2 TO )
2350  IF c$="****" THEN GO TO endfile
2360  GO SUB putline
2370  GO SUB getline
2380 NEXT i
2390 RETURN
2395
2399 REM transcribe up to line beginning with f$
2400 IF lena<4 THEN GO TO tooshort
2410 LET f$=a$(3 TO lena-1): LET lenf=LEN f$
2420 IF a$(2)<>a$(lena) THEN GO TO delimerr
2425
2430 GO SUB findf: IF found AND pos=1 THEN RETURN
2440 IF c$="****" THEN GO TO endfile
2450 GO SUB putline
2460 GO SUB getline
2470 GO TO 2430
2475
2479 REM transcribe up to line containing f$
2480 IF lena<5 THEN GO TO tooshort
2490 LET f$=a$(4 TO lena-1): LET lenf=LEN f$
2500 IF a$(3)<>a$(lena) THEN GO TO delimerr
2505
2510 IF c$="****" THEN GO TO endfile
2520 GO SUB findf
2530 IF found THEN RETURN
2540 GO SUB putline
2550 GO SUB getline
2560 GO TO 2510
2599
2600 REM *************
2601 REM * U command *
2602 REM *****************
2603 REM * Format U2fred ***********************
2604 REM * Use file called fred on microdrive 2 *
2605 REM * for further edit commands. This file *
2606 REM * should contain the Z command to come *
2607 REM * back to the keyboard.                *
2608 REM ****************************************
2609
2610 IF lena=1 THEN GO TO illegal
2615 IF use THEN PRINT '"Already using USE file": GO TO illegal
2620 IF a$(2)<"1" OR a$(2)>"8" THEN PRINT '"Missing microdrive number": GO TO il
legal
2630 LET b$(4)=a$(2 TO )
2635 LET d$=""
2640 LET comstrm=6
2650 OPEN #comstrm;"m";VAL b$(4,1);b$(4,2 TO )
2660 LET use=TRUE
2670 RETURN
2699
2700 REM *************
2701 REM * X command *
2702 REM *************
2703 REM * Format X  ********
2704 REM * Close merge file *
2705 REM ******************
2706
2710 IF lena<>1 THEN GO TO illegal
2720 IF NOT merge THEN PRINT '"No merge file open": GO TO illegal
2730 CLOSE #instrm
2740 LET instrm=4
2750 LET c$=t$: REM retrieve current line
2760 LET merge=FALSE
2770 LET b$(3)="          "
2775 LET index=1
2780 GO TO printc
2799
2800 REM *************
2801 REM * Z command *
2802 REM *************
2803 REM * Format Z  ******************************
2804 REM * Close USE file and go back to keyboard *
2805 REM * for editing commands.                  *
2806 REM ******************************************
2807
2810 IF lena<>1 THEN GO TO illegal
2820 IF NOT use THEN PRINT '"No USE file open": GO TO illegal
2830 CLOSE #comstrm
2840 LET comstrm=0
2850 LET b$(4)="          "
2855 LET d$="Command_"
2860 LET use=FALSE
2870 RETURN
2899
2900 REM ***************
2901 REM * CAT command *
2902 REM ***************
2903 REM * Format CAT n ***********
2904 REM * Catalogue microdrive n *
2905 REM ************************
2906
2910 IF lena<>2 THEN GO TO illegal
2920 IF a$(2)<"1" OR a$(2)>"8" THEN PRINT '"Invalid microdrive number": GO TO il
legal
2930 CAT VAL a$(2)
2940 GO TO printc
2999
3000 REM *************
3001 REM * * Command *
3002>REM *************
3003 REM * Format *  ******************************
3004 REM * Emergency command in case a USE file   *
3005 REM * runs off the end & executes the ****   *
3006 REM ****************************************
3007
3009 BEEP bp1+bp1,bp2
3010 PRINT FLASH 1;'"End of USE file??"
3019 BEEP bp1+bp1,bp2
3020 PRINT FLASH 1;"Closing USE file if open!!"
3030 GO TO 2820: REM Z command
3099
3100 REM *************
3101 REM * H command *
3102 REM *************
3103 REM * Format H  *******
3104 REM * Print Help page *
3105 REM *****************
3106
3110 IF lena<>1 THEN GO TO illegal
3120 CLS #
3125 PRINT INVERSE 1;"List of commands."
3130 RESTORE 3130
3140 FOR i=1 TO nocoms
3150 READ h$
3160 PRINT '"-";z$(i,1);"-"'h$
3170 NEXT i
3180 PRINT
3190 GO TO printc
3199
3200 DATA "Close current input file and    reopen at beginning. (C)"
3205 DATA "Transcribe rest of input file(s)to output file and end edit. (E)"
3210 DATA "Insert a line in front of        current line or go into input   mode.
 (I/string/ or IN)"
3215 DATA "Merge a 2nd input file (M1file2)"
3220 DATA "Move Pointer over so many lines of input file. (P4 or P/string/ or PC
/string/)"
3225 DATA "Quit. Close files & erase outputfile. (Q)"
3230 DATA "Replace find string with replacestring in current line.(R/f/r/)"
3235 DATA "Transcribe so many lines from   input file to output file (T6 orT/str
ing/ or TC/string/)"
3240 DATA "Use a file for editing commands.(U1newfile)"
3245 DATA "Close merged file. (X)"
3250 DATA "Close USE file. (Z)"
3255 DATA "Print this Help list (H)"
3260 DATA "Catalogue a microdrive. ( CAT 2)"
3265 DATA "Emergency command in case a USE file executes the **** at the   end o
f the file."
3398
3399 REM closefiles:
3400 REM ***************
3401 REM * Close files *
3402 REM ***************
3403
3405 PRINT '"Closing Files"
3410 IF merge THEN GO SUB 2730: REM X command
3420 IF use THEN GO SUB 2830: REM Z command
3425 IF NOT inputonly THEN CLOSE #instrm
3430 PRINT #7;"****": REM file terminator
3435 LET l(2)=l(2)+1
3440 CLOSE #7
3450 RETURN
3458
3459 REM endfile:
3460 REM **************************
3461 REM * Report end of i/p file *
3462 REM **************************
3463
3470 PRINT '"You have reached the end of yourinput file!"
3480 RETURN
3499
3500 REM ************
3501 REM * Synopsis *
3502 REM ************
3503
3510 PRINT 'l(1);" lines read from input file "
3520 PRINT l(2);" lines written to output file "
3530 PRINT l(3);" lines read from merge file "
3540 RETURN
3999
4000 ERASE "m";1;"Editor"
4010 SAVE *"m";1;"Editor" LINE 10
```

SCREEN SCROLLER

D. Garvin

Graphics windows are useful features to enhance the output capabilities of your computer. If your micro doesn't have them, fear not — this article will provide them for any Z80 based machine.

Recently, for display purposes I needed a routine to scroll the VDU screen both vertically and horizontally, accompanied by complete wrap-around in both cases. This article describes how this was achieved and developed further on a TRS 80 Model I with 48K of RAM (changes are given for those using 16K). The general principles however, can probably be applied to any micro that has a memory-mapped display.

Before going into details it is worth mentioning the two features of the Model I which make screen manipulation relatively easy.

- **The VDU contents have a permanent home but will travel.** The screen is made up of 16 lines, each line being 64 one-byte characters in length. Thus, the entire display takes up 1024 bytes of RAM. As shown in Fig 1, this chunk of bytes is *permanently* resident at addresses 15360 to 16383 in the memory map. Because the screen can be treated like any other RAM memory, it is possible to PEEK at a specific location and if necessary, POKE the value elsewhere in RAM including a new location on the screen. This method is fine if only a few bytes have to be relocated but shifting 1024 bytes around is very tedious. The solution is to use machine code to juggle the contents of the screen.

- **The keyboard can be scanned easily.** The addresses 14337 (3801H) to 14464 (3880H) are given over the keyboard. Figure 1 shows the arrangement for the eight rows of keys including row seven which has the arrow keys. As these particular keys are used extensively in the routines, it is important to understand what happens when they are pressed.

The following small program should help.

```
10 CLS
20 REM KEYBOARD PEEKING
30 A$=INKEY$
40 IF A$="" THEN 30
50 P=PEEK(14400)
60 PRINT P;
70 GOTO 30
```

If the keys from ENTER to SPACEBAR (excluding BREAK) are pressed in turn, the values 1,2,8,16,32,64 and 128 should be displayed. Each time a key is pressed in this row, which is mapped to the single byte address 14400, the bit pattern of that byte changes. These changes are used to decode which of the eight keys was pressed. The "resting state" of the byte is 00000000 but pressing UP ARROW, say, causes BIT 3 to become a 1 and the byte now equals decimal 8. Similarly, pressing SPACEBAR causes BIT 7 to become a 1 and the byte equals 128. So it is easy to tell which key has been pressed in this row by PEEKing at 14400 immediately following the press (Line 50 above).

But that's not the end of it. Shortly after the key has been released, the bit pattern of byte 14400 reverts back to 00000000 again, ready for the next key press. To see this happen, remove lines 30 and 40 above and note that the cavalcade of zeros across the screen can only be interrupted by pressing one of the seven keys in the block represented by 14400. Other keys have no effect as they are mapped to other addresses. Because the whole process is so fast, a regular scan of the keyboard can be accomplished either by PEEKing (14400), or by performing a machine code equivalent LD A, (14400), searching all the time for an input from one of the ARROW keys. Pressing any one of the four arrow keys, can be used to signal that a screen movement is required. Depending on which key is detected, the screen can then be moved sideways or up and down.

HOW IT WORKS

In the Z80 Instruction Set there are two useful block transfer commands which can quickly copy the contents of one set

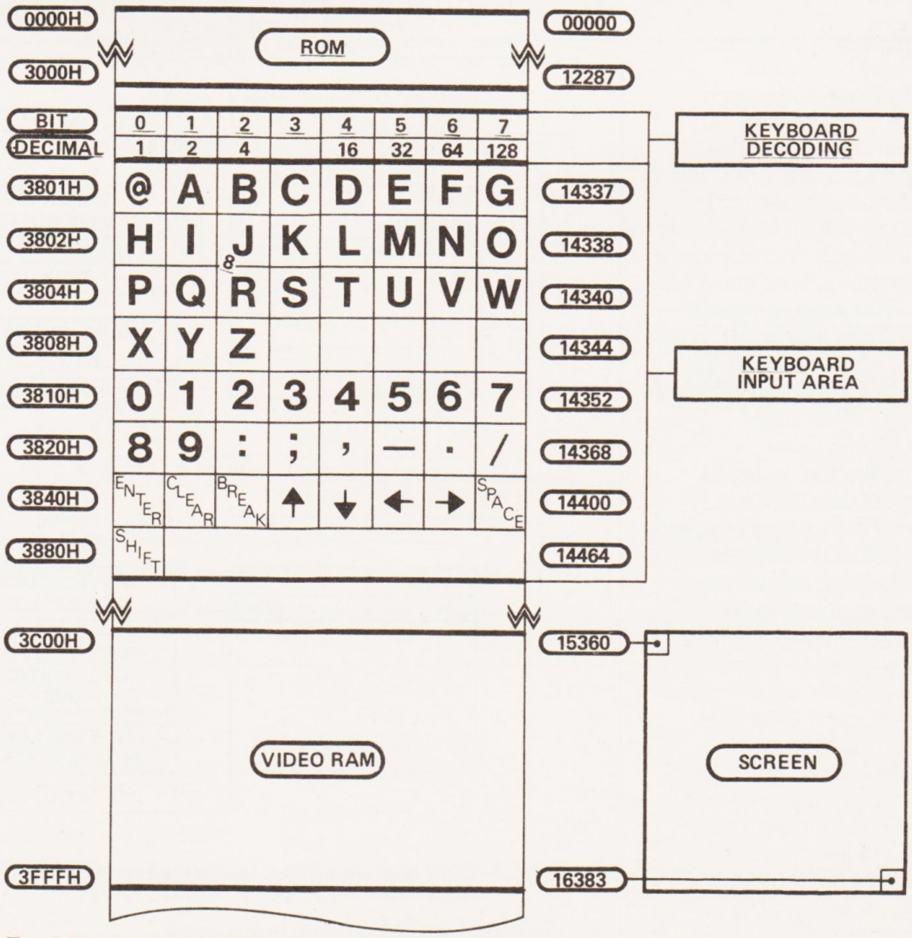


Fig. 1 Part of the TRS-80 memory map showing keyboard and video addresses

Listing 1

```
                00100 ;*****************************************************
                00110 ;LISTING 1
                00120 ;WINDOW SCROLLING AND ROTATION ASSEMBLER VERSION
                00130 ;TRS80 MODEL I/III
                00140 ;(C) T.A.ITHELL 1984
                00150 ;FULL AND PARTIAL SCREEN SCROLLING SUPPORTED
                00160 ;UP/DOWN/RIGHT/LEFT ARROWS MOVE SCREEN
                00170 ;CLEAR KEY TO RETURN TO BASIC
                00180 ;*****************************************************
                00190 ;
FE82            00200          ORG    65154          ;48K SYSTEMS
                00210 ;
FE82 3A4038     00220 SCAN     LD     A,(14400)      ;ARROW KEYS ROW ADDRESS
FE85 FE02       00230          CP     2              ;CLEAR KEY PRESSED?
FE87 C8         00240          RET    Z              ;YES-SO BACK TO BASIC
FE88 FE08       00250          CP     8              ;UP ARROW PRESSED?
FE8A 2812       00260          JR     Z,UP           ;YES-GO PROCESS UP
FE8C FE10       00270          CP     16             ;DOWN ARROW PRESSED?
FE8E 2851       00280          JR     Z,DOWN         ;YES-GO PROCESS DOWN
FE90 FE20       00290          CP     32             ;LEFT ARROW PRESSED?
FE92 CA2CFF     00300          JP     Z,LEFT         ;YES GO PROCESS LEFT
FE95 FE40       00310          CP     64             ;RIGHT ARROW PRESSED?
FE97 CA5AFF     00320          JP     Z,RIGHT        ;YES-GO PROCESS RIGHT
FE9A 18E6       00330          JR     SCAN           ;IGNORE REST:BACK TO SCAN
FE9C 00         00340          NOP
FE9D 00         00350          NOP
                00360 ;*****************************************************
FE9E 3AB2FF     00370 UP       LD     A,(SCROLL)     ;CHECK FOR SINGLE LINE
FEA1 FE01       00380          CP     1              ;SCROLL WHICH IS ILLEGAL
FEA3 28DD       00390          JR     Z,SCAN         ;YES-SO IGNORE KEY
FEA5 2AA0FF     00400          LD     HL,(START1)    ;WINDOW TOP LEFT ADDRESS
FEA8 11BAFF     00410          LD     DE,BUFFER      ;64 BYTE LINE BUFFER
FEAB ED4BAEFF   00420          LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FEAF EDB0       00430          LDIR                  ;COPY TOPLINE TO BUFFER
FEB1 ED4BB2FF   00440          LD     BC,(SCROLL)    ;NUMBER OF SCROLL LINES
FEB5 C5         00450          PUSH   BC             ;SAVE A COPY OF COUNTER
FEB6 ED5BA0FF   00460          LD     DE,(START1)    ;WINDOW TOP LEFT ADDRESS
FEBA 2AA4FF     00470          LD     HL,(START3)    ;WINDOW 2ND LINE ADDRESS
FEBD ED4BAEFF   00480 LOOP1    LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FEC1 EDB0       00490          LDIR                  ;LINE N+1 UP -) LINE N
FEC3 C1         00500          POP    BC             ;GET LINE COUNTER
FEC4 CD8AFF     00510          CALL   ZERO           ;GO CHECK IF COUNTER=0
FEC7 2809       00520          JR     Z,ENDIT1       ;YES-SO OFF TO TIDY UP
FEC9 C5         00530          PUSH   BC             ;NO-SO SAVE LINE COUNTER
FECA CD8EFF     00540          CALL   SUMS1          ;GO CALCULATE NEXT LINES
FECD CD99FF     00550          CALL   DELAY          ;SLOW THINGS DOWN
FED0 18EB       00560          JR     LOOP1          ;AND ONTO NEXT LINE
FED2 ED5BAAFF   00570 ENDIT1   LD     DE,(BOTLN1)    ;WINDOW BOTTOMLEFT ADDRES
FED6 21BAFF     00580          LD     HL,BUFFER      ;64 BYTE LINE BUFFER
FED9 ED4BAEFF   00590          LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FEDD EDB0       00600          LDIR                  ;COPY BUFFER TO BOTTOMLINE
FEDF 18A1       00610          JR     SCAN           ;BACK TO KEYBOARD INPUT
                00620 ;*****************************************************
FEE1 3AB2FF     00630 DOWN     LD     A,(SCROLL)     ;CHECK FOR SINGLE LINE
FEE4 FE01       00640          CP     1              ;SCROLL WHICH IS ILLEGAL
FEE6 289A       00650          JR     Z,SCAN         ;YES-SO IGNORE KEY
FEE8 2AAAFF     00660          LD     HL,(BOTLN1)    ;WINDOW BOTTOMLEFT ADDRESS
FEEB 11BAFF     00670          LD     DE,BUFFER      ;64 BYTE LINE BUFFER
FEEE ED4BAEFF   00680          LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FEF2 EDB0       00690          LDIR                  ;COPY BOTTOM LINE-)BUFFER
FEF4 ED4BB2FF   00700          LD     BC,(SCROLL)    ;NUMBER OF SCROLL LINES
FEF8 C5         00710          PUSH   BC             ;SAVE A COPY OF COUNTER
FEF9 ED5BAAFF   00720          LD     DE,(BOTLN1)    ;WINDOW BOTTOMLEFT ADDRESS
FEFD 2AACFF     00730          LD     HL,(BOTLN3)    ;WINDOW PENULTIMATE LINE
FF00 ED4BAEFF   00740 LOOP2    LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FF04 EDB0       00750          LDIR                  ;LINE N-1 DOWN-) LINE N
FF06 C1         00760          POP    BC             ;GET LINE COUNTER
FF07 CD8AFF     00770          CALL   ZERO           ;GO CHECK IF COUNTER=0
FF0A 2810       00780          JR     Z,ENDIT2       ;YES-SO OFF TO TIDY UP
FF0C C5         00790          PUSH   BC             ;NO-SO SAVE LINE COUNTER
FF0D ED4BB6FF   00800          LD     BC,(DIFFR1)    ;CALCULATE ADDRESSES
FF11 ED42       00810          SBC    HL,BC          ;OF NEXT LINES
FF13 EB         00820          EX     DE,HL
FF14 ED42       00830          SBC    HL,BC
FF16 EB         00840          EX     DE,HL
FF17 CD99FF     00850          CALL   DELAY          ;SLOW THINGS DOWN
FF1A 18E4       00860          JR     LOOP2          ;AND ONTO NEXT LINE
FF1C ED5BA0FF   00870 ENDIT2   LD     DE,(START1)    ;WINDOW TOP LEFT ADDRESS
FF20 21BAFF     00880          LD     HL,BUFFER      ;64 BYTE LINE BUFFER
FF23 ED4BAEFF   00890          LD     BC,(LINLEN)    ;LINE LENGTH TO SCROLL
FF27 EDB0       00900          LDIR                  ;COPY BUFFER TO TOP LINE
FF29 C382FE     00910          JP     SCAN           ;BACK TO KEYBOARD INPUT
                00920 ;*****************************************************
FF2C 3AAEFF     00930 LEFT     LD     A,(LINLEN)     ;CHECK FOR SINGLE BYTE
FF2F FE01       00940          CP     1              ;SCROLL WHICH IS ILLEGAL
FF31 CA82FE     00950          JP     Z,SCAN         ;YES-SO IGNORE KEY
FF34 ED4BB2FF   00960          LD     BC,(SCROLL)    ;NUMBER OF SCROLL LINES
FF38 C5         00970          PUSH   BC             ;SAVE A COPY OF COUNTER
FF39 2AA2FF     00980          LD     HL,(START2)    ;BYTE 2 TOP LINE
FF3C ED5BA0FF   00990          LD     DE,(START1)    ;BYTE 1 TOP LINE
FF40 ED4BB0FF   01000 LOOP3    LD     BC,(SHLINE)    ;SCROLL LINE LENGTH - 1
FF44 1A         01010          LD     A,(DE)         ;SAVE FIRST BYTE IN LINE
```

of addresses to another. Originally these commands (LDIR/LDDR) were used, but subsequently were dropped in favour of the code given in the Assembler Listing (Listing 1) and in another form in the Basic Listing (Listing 2). In the original only the whole screen could be scrolled. This present version allows any size of window to be specified, while retaining the whole screen option. By defining several sets of the same parameters in a three-dimensional array, multiple windows are also possible.

Figure 2 shows what happens when the ARROW keys are pressed. In brief, sideways movement is achieved by transferring a single byte from one end of a line to the other and then repeating the process for however many lines have been requested. Vertical scrolling is done on a line by line basis. The top line becomes the bottom line or vice versa according to the scrolling direction specified, again repeating for the number of lines requested.

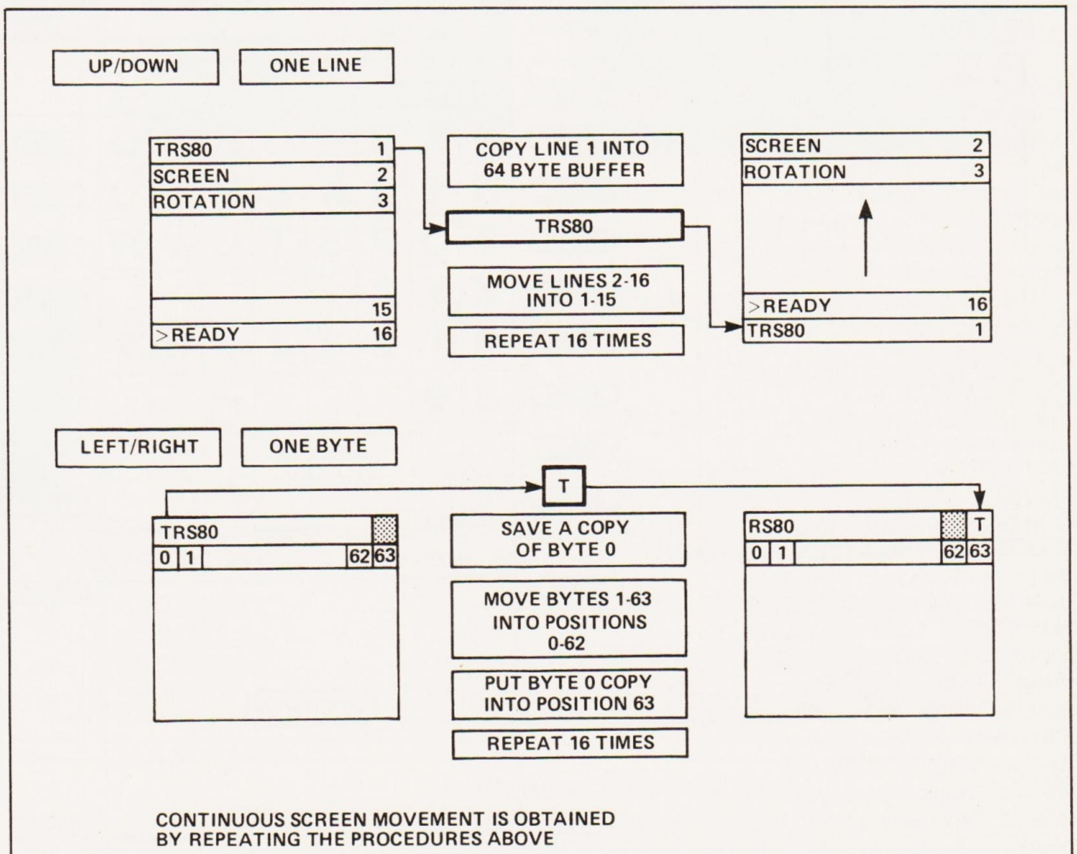


Fig. 2 How screen rotation is achieved

```
FF45 EDB0     01020        LDIR                 ;SHIFT LINE 1 BYTE LEFT
FF47 12       01030        LD    (DE),A         ;FIRST BYTE->LAST BYTE
FF48 CD99FF   01035        CALL  DELAY          ;SLOW THINGS DOWN
FF4B C1       01040        POP   BC             ;GET LINE COUNTER
FF4C CD8AFF   01050        CALL  ZERO           ;GO CHECK IF COUNTER=0
FF4F CA82FE   01060        JP    Z,SCAN         ;YES-SO BACK TO K/B INPUT
FF52 C5       01070        PUSH  BC             ;NO-SO SAVE LINE COUNTER
FF53 CD8EFF   01080        CALL  SUMS1          ;GO CALCULATE NEXT LINES
FF56 13       01090        INC   DE             ;ADD 1 TO DE TOTAL
FF57 23       01100        INC   HL             ;ADD 1 TO HL TOTAL
FF58 18E6     01120        JR    LOOP3          ;AND ONTO NEXT LINE
              01130 ;**************************************************
FF5A 3AAEFF   01140 RIGHT  LD    A,(LINLEN)     ;CHECK FOR SINGLE BYTE
FF5D FE01     01150        CP    1              ;SCROLL WHICH IS ILLEGAL
FF5F CA82FE   01160        JP    Z,SCAN         ;YES-SO IGNORE KEY
FF62 ED4BB2FF 01170        LD    BC,(SCROLL)    ;NUMBER OF SCROLL LINES
FF66 C5       01180        PUSH  BC             ;SAVE A COPY OF COUNTER
FF67 2AA8FF   01190        LD    HL,(LNEND2)    ;BYTE 62 TOP LINE
FF6A ED5BA6FF 01200        LD    DE,(LNEND1)    ;BYTE 63 TOP LINE
FF6E ED4BB0FF 01210 LOOP4  LD    BC,(SHLINE)    ;SCROLL LINE LENGTH - 1
FF72 1A       01220        LD    A,(DE)         ;SAVE LAST BYTE IN LINE
FF73 EDB8     01230        LDDR                 ;SHIFT LINE 1 BYTE RIGHT
FF75 12       01240        LD    (DE),A         ;LAST BYTE->FIRST BYTE
FF76 CD99FF   01245        CALL  DELAY          ;SLOW THINGS DOWN
FF79 C1       01250        POP   BC             ;GET LINE COUNTER
FF7A CD8AFF   01260        CALL  ZERO           ;GO CHECK IF COUNTER=0
FF7D CA82FE   01270        JP    Z,SCAN         ;YES-SO BACK TO K/B INPUT
FF80 C5       01280        PUSH  BC             ;NO-SO SAVE LINE COUNTER
FF81 ED4BB8FF 01290        LD    BC,(DIFFR2)
FF85 CD92FF   01300        CALL  SUMS2          ;GO CALCULATE NEXT LINES
FF88 18E4     01320        JR    LOOP4          ;AND ONTO NEXT LINE
              01330 ;**************************************************
FF8A 0B       01340 ZERO   DEC   BC             ;LINECOUNT-1
FF8B 78       01350        LD    A,B            ;B=C=0?
FF8C B1       01360        OR    C
FF8D C9       01370        RET                  ;BACK TO CALLING ROUTINE
FF8E ED4BB4FF 01380 SUMS1  LD    BC,(DIFFR0)    ;CALCULATE NEXT LINE
FF92 ED4A     01390 SUMS2  ADC   HL,BC          ;ADDRESSES
FF94 EB       01400        EX    DE,HL
FF95 ED4A     01410        ADC   HL,BC
FF97 EB       01420        EX    DE,HL
FF98 C9       01430        RET                  ;BACK TO CALLING ROUTINE
FF99 010000   01440 DELAY  LD    BC,0           ;POKE DELAY HERE
FF9C CD6000   01450        CALL  60H            ;ROM DECREMENT COUNT
FF9F C9       01460        RET                  ;BACK TO CALLING ROUTINE
              01470 ;**************************************************
              01480 ;BUFFERS FOR WINDOW PARAMETERS PASSED FROM BASIC
FFA0 0000     01490 START1  DEFW  0             ;WINDOW TOP LEFT ADDRESS
FFA2 0000     01500 START2  DEFW  0             ;WINDOW TOP LEFT ADDRESS+1
FFA4 0000     01510 START3  DEFW  0             ;WINDOW TOPLEFT ADDRESS+64
FFA6 0000     01520 LNEND1  DEFW  0             ;WINDOW TOPRIGHT ADDRESS
FFA8 0000     01530 LNEND2  DEFW  0             ;WINDOW TOPRIGHT ADDRESS-1
FFAA 0000     01540 BOTLN1  DEFW  0             ;WINDOW BOTTOMLEFT ADDRES
FFAC 0000     01550 BOTLN3  DEFW  0             ;WINDOW BOTTOMLEFT-64
FFAE 0000     01560 LINLEN  DEFW  0             ;LINE LENGTH TO SCROLL
FFB0 0000     01570 SHLINE  DEFW  0             ;LINE LENGTH-1
FFB2 0000     01580 SCROLL  DEFW  0             ;NUMBER OF SCROLL LINES
FFB4 0000     01590 DIFFR0  DEFW  0             ;CONSTANTS TO ADD/SUBTRACT
FFB6 0000     01600 DIFFR1  DEFW  0             ;DEPENDING ON SCROLL
FFB8 0000     01610 DIFFR2  DEFW  0             ;DIRECTION REQUIRED
FFBA          01620 BUFFER  DEFS  64            ;64 BYTE LINE BUFFER
FFFA 00       01630         NOP
FFFB 00       01640         NOP
FFFC 0000     01650 POKEIT  DEFW  0             ;VALUE=AUTOSCROLL DIRECTN
FE82          01660         END   SCAN
00000 TOTAL ERRORS
BOTLN1  FFAA
BOTLN3  FFAC
BUFFER  FFBA
DELAY   FF99
DIFFR0  FFB4
DIFFR1  FFB6
DIFFR2  FFB8
DOWN    FEE1
ENDIT1  FED2
ENDIT2  FF1C
LEFT    FF2C
LINLEN  FFAE
LNEND1  FFA6
LNEND2  FFA8
LOOP1   FEBD
LOOP2   FF00
LOOP3   FF40
LOOP4   FF6E
POKEIT  FFFC
RIGHT   FF5A
SCAN    FE82
SCROLL  FFB2
SHLINE  FFB0
START1  FFA0
START2  FFA2
START3  FFA4
SUMS1   FF8E
SUMS2   FF92
UP      FE5E
ZERO    FF8A
```

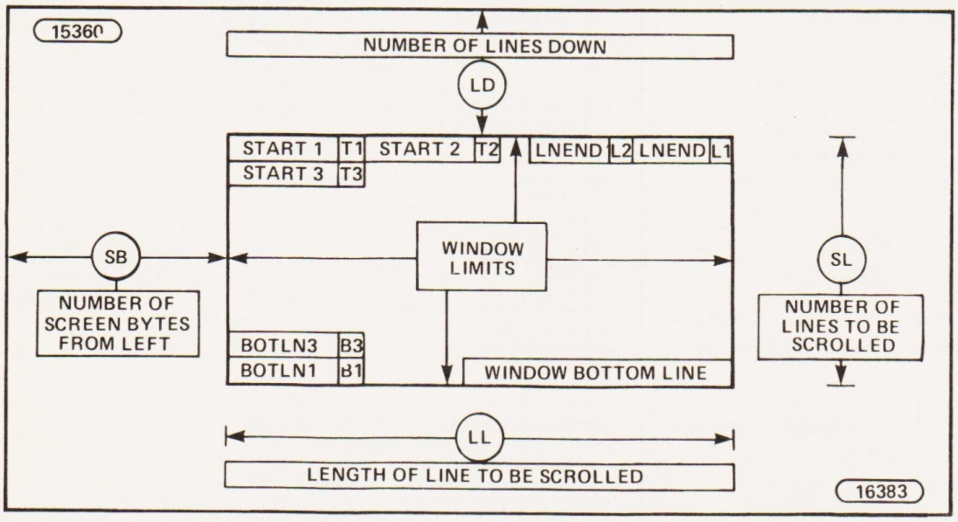


Fig. 3 Window parameters

TABLE 1

Window parameter limits to the user input

LD	1 - 16
LL	1 - 64
SB	0 - 63
SL	1 - 16
DE (Delay)	1 - 65535

Note: the program calculates all the other values from these inputs.

Listings 1 and 2 have detailed comments about individual sections of the program. To work properly, the program requires that four parameters be defined. These numbers can describe either the whole screen or any regularly shaped part. Referring to Fig 3:

LINES DOWN	(LD)	gives the position of the window top line
BYTES LEFT	(SB)	gives the extreme left of the window
LINE LENGTH	(LL)	gives the width of the window
SCROLL LINES	(SL)	gives the depth of the window

For example, the whole screen is defined by:

LINES DOWN	= 1
BYTE LEFT	= 0
LINE LENGTH	= 64
SCROLL LINES	= 16

To slow the scrolling enough to make the display readable or for slow-motion animation, a DELAY is also specified. The fastest speed needs a DELAY = 1. It is also worthwhile requesting 65535, the slowest allowed, because the workings of the routine become almost painfully clear as successive bytes or lines are ponderously hauled around the screen.

As shown in Table 1, there are some limitations on the minimum and maximum size of window allowed. The code in lines 350 to 440 is designed to filter out the more esoteric requests, such as a window of 0 lines!! Error trapping, however, is not complete except for lines 350 to 440 which protect the machine code.

Another important point is that columns one byte wide can only be scrolled horizontally. The comments in Listing 1 give the details of how this is achieved. Finally, only square or rectangular windows are supported. Ovals, diamonds, circles, keyholes and the like are definitely out!

Once input, the values of these window parameters are stored in a three-dimensional array called window ready to be accessed when called. Additional information required by the machine code program is also calculated from the four inputted values and stored in the array.

If the multiple window option is being used, the number and direction of moves required must also be supplied (see later). When a particular window is to be scrolled, the relevant parameters are passed from the array into a set of buffers. Finally, to move the screen, a USR call is executed and the display should shift as directed by the user.

With a minimum number of changes, the program can be adapted for a variety of uses. Examples are given below. It is important to realise that much of the extra coding is for the purpose of demonstration only. Suggestions are made in the text whenever the modifications would justify a separate saving of the program.

Essentially, the options are combinations of :

- Manual/Program control of scrolling
- Single or Multiple Window Scrolling
- Up/Down/Right/Left Scrolling
- Defining any size of window (2-1024 bytes)

ENTERING AND USING THE PROGRAM

Like many programs, once the general ideas had been worked out, it became clear that with a few changes several useful spin-offs could be produced. Listing 2 contains the core program which will be modified to produce various optional extras.

The listings are for a 48K machine, the few changes required for 16K implementation are given in Listing 3. These changes are required because the machine code is not relocatable and the buffer addresses will also be different. A list of important addresses is given in Table 2.

The Assembler Listing 1 is to allow easy relocation by typing in the source code and specifying another origin. However, Listing 2 contains all the code in a form which can be poked into memory. If you do decide to relocate the code then allow for the following:

- BUFFER at the end of the code is 64 bytes long and allowances for this and the other buffers must be made when defining a new origin.
- Data will be poked into these buffers from BASIC. Relocating the code will change the buffer addresses and you must alter the BASIC Listings accordingly.

Begin by setting MEM SIZE to 65150 to protect the machine code, then type in and save a copy of Fig 3. After initialisation, the prompts below will appear. The simplest window to specify is full screen. Enter the values given.

```
NUMBER OF WINDOWS REQUIRED? 1 (Enter)
(Keep it simple!)
PARAMETERS FOR WINDOW 1
NUMBER OF LINES DOWN FROM TOP (1-16 LINES)  1 (Enter)
(VDU Line 1 = Top line)
NUMBER OF BYTES ACROSS SCREEN FROM LEFT MARGIN (Ø-63
BYTES)?
LENGTH OF LINE TO BE SCROLLED (1-64 BYTES)? 64 (Enter)
(64 bytes = 1 line)
NUMBER OF LINES TO SCROLL (1-16 LINES)? 16 (Enter)
(16 = whole screen)
ROTATION DELAY (1-65535)? 1 (Enter)
(Very fast!)
```

If this program is eventually used as a subroutine in larger projects, it is important that you do not exceed the parameter limits given in each prompt above ...otherwise the dreaded MEM SIZE? will almost certainly materialise.

After a short wait while the program generates some garbage strings to fill the screen, you will be prompted to press the ARROW keys or CLEAR. The screen should move in the direction

Listing 2

```
10 REM LISTING 2
20 REM WINDOW SCROLLING AND ROTATION BASIC VERSION
30 REM 48K TRS80 MODEL I/III (C) T.A.ITHELL 1984
40 REM POKING MACHINE CODE INTO MEMORY FROM 65154
50 DEFINTA-Z:CLEAR200:CLS:PRINT"POKING DATA INTO MEMORY"
60 FORT=382TO97STEP-1:READN:POKE-T,N:NEXT
70 DATA58,64,56,254,2,200,254,8,40,18,254,16,40,81
80 DATA254,32,202,44,255,254,64,202,90,255,24,230,0
90 DATA0,58,178,255,254,1,40,221
100 DATA42,160,255,17,186,255,237,75,174,255,237
110 DATA176,237,75,178,255,197,237,91,160,255,42,164,255,237,75
120 DATA174,255,237,176,193,205,138,255,40,9,197,205,142,255,205
130 DATA153,255,24,235,237,91,170,255,33,186,255,237,75,174,255
140 DATA237,176,24,161,58,178,255,254,1,40,154,42,170,255,17
150 DATA186,255,237,75,174,255,237
160 DATA176,237,75,178,255,197,237,91,170,255,42,172,255,237,75
170 DATA174,255,237,176,193,205,138,255,40,16,197,237,75,182,255
180 DATA237,66,235,237,66,235,205,153,255,24,228,237,91,160,255
190 DATA33,186,255,237,75,174,255,237,176,195,130,254,58,174
200 DATA255,254,1,202,130,254,237,75,178
210 DATA255,197,42,162,255,237,91,160,255,237,75,176,255,26,237
220 DATA176,18,205,153,255,193,205,138,255,202,130,254,197,205,142,255,19,35
230 DATA24,230,58,174,255,254,1,202,130,254,237,75
240 DATA178,255,197,42,168,255,237,91
250 DATA166,255,237,75,176,255,26,237,184,18,205,153,255,193,205,138,255,202
260 DATA130,254,197,237,75,184,255,205,146,255,24
270 DATA228,11,120,177,201,237,75,180,255,237,74,235,237,74,235
280 DATA201,1,0,0,205,96,0,201
290 REM INPUT NUMBER/SIZE OF WINDOWS REQUIRED
300 CLS
310 INPUT"NUMBER OF WINDOWS REQUIRED";NW
320 DIMWI(NW,50,2)
330 FORR=1TONW
340 PRINT"PARAMETERS FOR WINDOW";R
350 INPUT"NUMBER OF LINES DOWN FROM TOP (1-16 LINES)";LD
360 IFLD(1ORLD)16THEN350
370 INPUT"NUMBER OF BYTES ACROSS SCREEN FROM LEFT MARGIN (0-63 BYTES)";SB
380 IFSB(0ORSB)63THEN370
390 INPUT"LENGTH OF LINE TO BE SCROLLED (1-64 BYTES)";LL
400 IFLL(1ORLL)64ORLL+SB)64THEN390
410 INPUT"NUMBER OF LINES TO SCROLL (1-16 LINES)";SL
420 IFSL(1ORSL=)17ORSL+LD)17THEN410
430 INPUT"ROTATION DELAY (1-65535)";DE
440 IFDE(1ORDE)65535THEN430
450 CLS
460 C=1
470 REM FOR EACH WINDOW CALCULATE REST OF WINDOW PARAMETERS
480 REM FROM INPUTS AND THEN LOAD THEM INTO ARRAY WI
490 T1=15360+((LD-1)*64)+SB:M=INT(T1/256):L=T1-(M*256):GOSUB710
500 T2=T1+1:M=INT(T2/256):L=T2-(M*256):GOSUB710
510 T3=T1+64:M=INT(T3/256):L=T3-(M*256):GOSUB710
520 L1=T1+(LL-1):M=INT(L1/256):L=L1-(M*256):GOSUB710
530 L2=T1+(LL-2):M=INT(L2/256):L=L2-(M*256):GOSUB710
540 B1=15360+(((LD+SL)-2)*64)+SB:M=INT(B1/256):L=B1-(M*256):GOSUB710
550 B3=B1-64:M=INT(B3/256):L=B3-(M*256):GOSUB710
560 SH=LL-1
570 D0=64-LL
580 D1=64+LL
590 D2=64+SH
600 WI(R,15,0)=LL
610 WI(R,17,0)=SH
620 WI(R,19,0)=SL
630 WI(R,21,0)=D0
640 WI(R,23,0)=D1
650 WI(R,25,0)=D2
660 DM=INT(DE/256):DL=DE-(DM*256)
670 WI(R,27,0)=DL
680 WI(R,28,0)=DM
690 NEXTR
700 GOTO720
710 WI(R,C,0)=L:WI(R,C+1,0)=M:C=C+2:RETURN
720 REM DEMONSTRATION SCREEN LOAD AND PROGRAM RUN
730 CLS:SW=1:REM SW=1 MEANS WINDOW 1
740 REM GENERATE A SCREENFUL OF GARBAGE
750 FORF=0TO95:FF=RND(159)+32:F$=F$+CHR$(FF):NEXT
760 FORT=0TO896STEP48:PRINT@T,F$;:NEXT
770 PRINT@140,"*** SCROLLING-PRESS UP/DOWN/RIGHT/LEFT ARROWS ***";
780 PRINT@204,"*** TO RETURN-PRESS CLEAR KEY ***";
790 REM PUT THE PARAMETERS FOR WINDOW 1 INTO BUFFERS
800 ME=96:FORC=1TO26:POKE-ME,WI(SW,C,0):ME=ME-1:NEXT
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
820 REM CALL MACHINE CODE
830 POKE16526,130:POKE16527,254:Z=USR(0)
840 CLS:PRINT"DONE"
```

Listing 3

```
60 FORT=32385TO32571:READN:POKET,N:NEXT
70 DATA58,64,56,254,2,200,254,8,40,18,254,16,40,81
80 DATA254,32,202,44,127,254,64,202,90,127,24,230,0
90 DATA0,58,178,127,254,1,40,221
100 DATA42,160,127,17,186,127,237,75,174,127,237
110 DATA176,237,75,176,127,197,237,91,160,127,42,164,127,237,75
120 DATA174,127,237,176,193,205,138,127,40,9,197,205,142,127,205
130 DATA153,127,24,235,237,91,170,127,33,196,127,237,75,174,127
140 DATA237,176,24,161,58,178,127,254,1,40,154,42,170,127,17
150 DATA186,127,237,75,174,127,237
160 DATA176,237,75,178,127,197,237,91,170,127,42,172,127,237,75
170 DATA174,127,237,176,193,205,138,127,40,16,197,237,75,182,127
180 DATA237,66,235,237,66,235,205,153,127,24,228,237,91,190,127
190 DATA33,186,127,237,75,174,127,237,176,195,130,126,58,174
200 DATA127,254,1,202,130,126,237,75,178
210 DATA127,197,42,162,127,237,91,160,127,237,75,176,127,26,237
220 DATA176,18,205,153,127,193,205,138,127,202,130,126,197,205,142,127,19,35
230 DATA24,230,58,174,127,254,1,202,130,126,237,75
240 DATA178,127,197,42,168,127,237,91
250 DATA166,127,237,75,176,127,26,237,184,18,205,153,127,193,205,138,127,202
260 DATA130,126,197,237,75,184,127,205,146,127,24
270 DATA228,11,120,177,201,237,75,180,127,237,74,235,237,74,235
280 DATA201,1,0,0,205,96,0,201

800 ME=32672:FORC=1TO26:POKEME,WI(SW,C,0):ME=ME+1:NEXT
810 POKE32666,WI(SW,27,0):POKE32667,WI(SW,28,0)
820 REM CALL MACHINE CODE
830 POKE16526,130:POKE16527,126:Z=USR(0)
```

of the ARROW key pressed. The blank line at the bottom of the display gives a point of reference. Once you are happy that the program is working satisfactorily, press CLEAR to return BASIC and then reRUN using different parameters. Table 3 gives a list of suggestions you might like to try to get the feel of the program. For the moment though, always answer the NUMBER OF WINDOWS prompt with 1.

This simple rotation and screen shifting is extremely useful when creating animation sequences. A small movement of the whole screen or part of it, one way or the other can save hours of redrawing.

● **Modification 1** In its present form, the only way of getting out of the machine code loop is to press CLEAR (or if really lost the RESET button). Obviously it would be more flexible if the program could return to BASIC of its own accord. Listing 4 has the required changes, the new values are underlined. They simply replace JumP to keyboard SCAN with RETurn to BASIC instructions, BREAK will work normally again. Change the BASIC lines as shown in Listing 4 and RUN this new version. Again, use Table 3 for screen parameter examples. Although nothing dramatically different happens, after each ARROW key press the routine goes back to the BASIC driver program. This means that some processing can be done between successive screen moves.

● **Modification 2** The usual way to move things about is to have the small ones move over big ones. Cars move over

TABLE 2

Variable Assembler	Name Basic	48K Hex	48K Dec	48K LSB	48K MSB	16K Hex	16K LSB	16K MSB
DELAY	DE	FF9A	65434	-102	-101	7F9A	32666	32667
START 1	T1	FFA0	65440	- 96	- 95	7FA0	32672	32673
START 2	T2	FFA2	65442	- 94	- 93	7FA2	32674	32675
START 3	T3	FFA4	65444	- 92	- 91	7FA4	32676	32677
LNEND1	L1	FFA6	65446	- 90	- 89	7FA6	32678	32679
LNEND2	L2	FFA8	65448	- 88	- 87	7FA8	32680	32681
BOTLN1	B1	FFAA	65450	- 86	- 85	7FAA	32682	32683
BOTLN3	B3	FFAC	65452	- 84	- 83	7FAC	32684	32685
LINLEN	LL	FFAE	65454	- 82	- 91	7FAE	32686	32687
SHLINE	SH	FFB0	65456	- 80	- 79	7FB0	32688	32689
SCROLL	SL	FFB2	65458	- 78	- 77	7FB2	32690	32691
DIFFR0	D0	FFB4	65460	- 76	- 75	7FB4	32692	32693
DIFFR1	D1	FFB6	65462	- 74	- 73	7FB6	32694	32695
DIFFR2	D2	FFB8	65464	- 72	- 71	7FB8	32696	32697
BUFFER	--	FFBA	65466	- 70	- 69	7FBA	32698	32699
POKEIT	--	FFFC	65532	- 4	- 3	7FFC	32764	32765
MEM SIZE		65150				32380		

TABLE 3

All have a DELAY of 500
Use ARROW keys to see scrolling

LINES DOWN (LD)	BYTES LEFT (SB)	LINE LENGTH (LL)	SCROLL LINE (SL)	EFFECT ON SCREEN PART SCROLLLING
1	0	64	16	Whole screen
1	0	32	16	Left half
9	0	64	8	Lower half
1	0	2	16	2 Left columns
1	0	64	2	2 Top lines
1	62	2	16	2 Right columns
15	0	64	2	2 Bottom lines
9	33	31	8	Bottom right
1	0	5	3	Top left
1	54	10	3	Top right
14	0	10	3	Bottom left
14	54	10	3	Bottom right
5	16	32	8	Screen centre
1	0	64	1	1 Line sideways
14	0	32	1	Line 14 left
8	0	64	1	Centre line
1	0	1	16	Left column
1	63	1	16	Right column
1	32	1	16	Centre column

roads, ships over oceans, pixels over screens. Sometimes this is reversed, paper moves over a typewriter head. Add the changes in Listing 5 and set the screen parameters for full scrolling and fastest speed. The key prompts will appear with a block — CHR$(191)-just below. Pressing the ARROWS now should cause broad horizontal and narrow vertical bands to be traced as the screen passes over the block. What's happening?

After each key press, the routine returns to BASIC fleetingly and executes line 825 which prints the block at the same position each time. But because the screen has moved on a byte since the last time, another block appears adjacent to the last one. Hence the traces. This is like having a permanently fixed pencil and mobile paper.

Incidentally, you may have noticed that the ARROW keys appear to have been reversed. This isn't really the case, remember that the ARROWS control the screen movement not the block which is stationary.

Other characters can be used instead of 191. A random character generator would be:

```
825 RR=RND(64)+127: PRINT@540,CHR$(RR);
```

Commands like SET and POKE can also be used. Finally, notice that despite the return to BASIC after each key press, the screen still moves at a very acceptable speed.

● **Modification 3** The next stage is to take screen control out of the user's hands and give it to the computer. When the ARROW keys are pressed, values of 8,16,32 and 64 are generated as mentioned earlier. Whenever the machine code program encounters one of the these in its periodic keyboard SCAN, it jumps to the appropriate subroutine and moves the screen as directed. However, the Accumulator cannot tell the difference between a value that comes from a key press and one that come from some other source.

Knowing this, it is possible to bypass the keyboard entirely and simply POKE the values 8,16,32 and 64 into the machine code program area. The address used as the dump is called POKEIT in the Assembler Listing. Once the value is in place, the program can go and collect it and relocate the screen accordingly.

Listing 6 shows the alterations needed to make the Accumulator scan POKEIT for data instead of the keyboard. 16K users should note lines 71 and 830.

It would be worth saving a copy of this version of the program because it is now significantly different from the original in its mode of operation.

The first three bytes of machine code mean LOAD the ACCUMULATOR with the contents of the POKEIT buffer. As only four values are recognized by the program, namely 8,16,32, and 64, you must be careful to ensure that only these are input into POKEIT otherwise nothing will happen. Line 830 loads the POKEIT buffer with the value P and then calls the scroll routine. As in the previous program, a return to BASIC follows each screen move. Line 835 does a quick scan of the SPACEBAR key. Press it each time you want to try another value for P. The program will remain in a loop until (BREAK) is pressed. If you RUN the program and specify full screen and a speed of 500, the prompt message should move in the direction specified by the value of P.

This mode was the original reason for developing the program. I wanted to display pages of revolving text that could move automatically at a pre-determined rate, first vertically then horizontally.

● **Modification 3A** Adding the slight changes in Listing 7 gives a simple demonstration of one use of automatic rotation. RUN the program and specify the following parameters:

PARAMETER:		
WINDOW NUMBER	1	(ENTER)
LINES DOWN	6	(ENTER)
BYTES FROM LEFT	33	(ENTER)
LINE LENGTH	2	(ENTER)
LINES TO SCROLL	5	(ENTER)
DELAY	1000	(ENTER)

Revolving words, sentences and pages may be of use to teachers of reading trying to increase a child's reading speed, word recognition and comprehension. For instance, a whole series of similar words could be made to revolve through the sentence, with the child stopping the scroll when a word seems to make sense.

● **Modification 4** The next two examples show that other things can be moved besides simple sentences and words. The first might be of use in physics or mathematics, the second is really just an interesting piece of graphics. Listing 8 has the new lines of BASIC. Line 1030 determines the rotation direction.

If the parameters below are entered, after the two waves have been drawn, the top half of the screen will rotate while the lower half remains stationary. A partial move like this, manually or automatically controlled, could be of use in teaching the ideas of phase differences. Of course, the number and form of the curves can easily be altered by changing the equation in line 960.

PARAMETER:		
WINDOW NUMBER	1	(ENTER)
LINES DOWN	1	(ENTER)
BYTES FROM LEFT	0	(ENTER)
LINE LENGTH	64	(ENTER)
LINES TO SCROLL	7	(ENTER)
DELAY	100	(ENTER)

● **Modification 4A** To produce a wave of the form shown in Fig 4, enter the BASIC lines given in Listing 9. Use the parameters

```
10 REM LISTING 2 MODIFICATION 1
15 REM LINES 70,80,90,140,190,200,220,230,250,780,840 CHANGED
!
!
!
!
70 DATA58,64,56,0,0,0,254,8,40,18,254,16,40,81
80 DATA254,32,202,44,255,254,64,202,90,255,201,0,0
90 DATA0,58,178,255,254,1,200,0
100 DATA42,160,255,17,186,255,237,75,174,255,237
110 DATA176,237,75,178,255,197,237,91,160,255,42,164,255,237,75
120 DATA174,255,237,176,193,205,138,255,40,9,197,205,142,255,205
130 DATA153,255,24,235,237,91,170,255,33,186,255,237,75,174,255
140 DATA237,176,201,0,58,178,255,254,1,200,0,42,170,255,17
150 DATA186,255,237,75,174,255,237
160 DATA176,237,75,178,255,197,237,91,170,255,42,172,255,237,75
170 DATA174,255,237,176,193,205,138,255,40,16,197,237,75,182,255
180 DATA237,66,235,237,66,235,205,153,255,24,228,237,91,160,255
190 DATA33,186,255,237,75,174,255,237,176,201,0,0,58,174
200 DATA255,254,1,200,0,0,237,75,178
210 DATA255,197,42,162,255,237,91,160,255,237,75,176,255,26,237
220 DATA176,18,205,153,255,193,205,138,255,200,0,0,197,205,142,255,19,35
230 DATA24,230,58,174,255,254,1,200,0,0,237,75
240 DATA178,255,197,42,168,255,237,91
250 DATA166,255,237,75,176,255,26,237,184,18,205,153,255,193,205,138,255,200
260 DATA0,0,197,237,75,184,255,205,146,255,24
270 DATA228,11,120,177,201,237,75,180,255,237,74,235,237,74,235
280 DATA201,1,0,0,205,96,0,201
!
!
!
!
780 PRINT@204,"*** TO RETURN-PRESS BREAK KEY ***";
!
!
!
840 GOTO830
```

Listing 4

```
10 REM LISTING 2 MODIFICATION 2
15 REM LINES 750,760 DELETED:INSERT 825:CHANGE 840
!
!
!
740 REM GENERATE A SCREENFUL OF GARBAGE
770 PRINT@140,"*** SCROLLING-PRESS UP/DOWN/RIGHT/LEFT ARROWS ***";
780 PRINT@204,"*** TO RETURN-PRESS BREAK KEY ***";
!
!
!
!
820 REM CALL MACHINE CODE
825 PRINT@540,CHR$(191);
830 POKE16526,130:POKE16527,254:Z=USR(0)
840 GOTO825
```

Listing 5

```
10 REM LISTING 2 MODIFICATION 3
15 REM LINES 740,770,790 DELETED:70,825,830,840 CHANGED:835 INSERTED
'
'
'
'
70 DATA58,252,255,0,0,0,254,8,40,18,254,16,40,81
71 REM 16K SYSTEM DATA 58,252,127.......
'
'
'
'
730 CLS:SW=1:REM SW=1 MEANS WINDOW 1
790 REM PUT THE PARAMETERS FOR WINDOW 1 INTO BUFFERS
800 ME=96:FORC=1TO26:POKE-ME,WI(SW,C,0):ME=ME-1:NEXT
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
820 REM CALL MACHINE CODE
825 CLS:PRINT@704,CHR$(15)::INPUT"VALUE OF P TO BE POKED INTO POKEIT 8,16,32,64ONLY-NO CHECK FOR INVALID
ENTRIES:SPACEBAR TO CHANGE P";P
830 POKE-4,P:POKE16526,130:POKE16527,254:Z=USR(0)/16K830 POKE32764,P:POKE16526,130:POKE16527,126:Z=USR(0)
835 Q=PEEK(14400):IFQ=128THEN825
840 GOTO830
```

Listing 6

```
10 REM LISTING 2 MODIFICATION 3A
15 REM 825 CHANGED:821 INSERTED
'
'
'
'
821 CLS:PRINT@462,"JACK AND JILL WENT UP THE HILL";
825 PRINT@740,CHR$(15):INPUT"VALUE OF P TO BE POKED INTO POKEIT 8,16,32,64 ONLY-NO CHECK FOR INVALID ENTR
IES:SPACEBAR TO CHANGE P";P
```

Listing 7

```
10 REM LISTING 2 MODIFICATION 4
15 REM 820-840 DELETED:900-1040 INSERTED
'
'
'
'
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
900 REM PHASE DIFFERENCE BETWEEN TWO WAVES
910 CLS
920 X=0
930 FORA=0TO24STEP24
940 FORB=1TO108
950 DG=B*0.174
960 Y=(SIN(DG)+1)*10+A
970 SET(X,Y)
980 X=X+1.184
990 IFX)127THENX=0
1000 NEXTB
1010 NEXTA
1020 POKE16526,130:POKE16527,254
1030 POKE-4,64:Z=USR(0)
1040 GOTO1030
```

```
10 REM LISTING 2 MODIFICATION 4A
15 REM 930,1010 DELETED:960,970 CHANGED:965-968,971 INSERTED
'
'
'
'
'
'
'
900 REM CRESTED WAVE GRAPHICS DISPLAY
910 CLS
920 X=0
940 FORB=1TO108
950 DG=B*0.174
960 Y=(SIN(DG)*5)+10
965 FORA=1TO30STEP2
966 X1=X-(A+1.2)
967 IFX1)127THENX1=X1-127
968 IFX1(0THENX1=X1+127
970 SET(X1,Y+(1.12*A))
971 NEXTA
980 X=X+1.184
990 IFX)127THENX=0
1000 NEXTB
1020 POKE16526,130:POKE16527,254
1030 POKE-4,64:Z=USR(0)
1040 GOTO1030
```

Listing 8 (Above left)

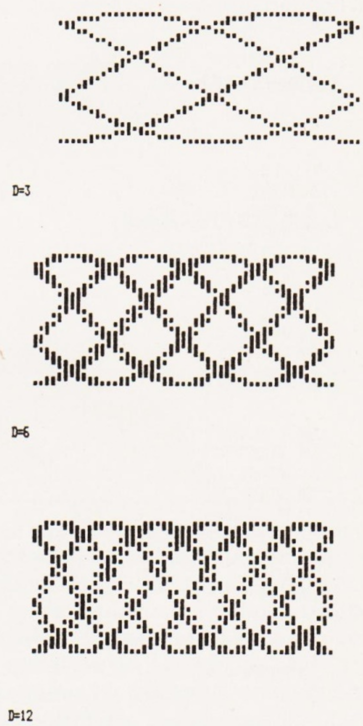
Listing 9 (Above)

Fig 4 (Left) Waveform produced by Listing 9

```
10 REM LISTING 2 MODIFICATION 5
15 REM INSERT NEW ROUTINE FROM 900-2000
'
'
'
'
'
'
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
900 REM LISSAJOUS FIGURES
910 ON ERROR GOTO 2000
920 CLS
930 P=64
940 C1=2
950 D=3
960 Y=Y+D
970 Y1=(COS(Y*0.0174)+1.5)*15
980 SET(63,Y1)
990 CO=CO+C1
1000 IFCO)96THENCO=CO-(2*C1):C1=-C1:P=32:GOTO1020
1010 IFCO(0THENCO=CO-(2*C1):C1=-C1:P=64
1020 POKE-4,P
1030 POKE16526,130:POKE16527,254:Z=USR(0)
1040 GOTO960
2000 RESUMENEXT
```

Listing 10 (Above)

Fig 5 (below) Some Lissajous figures



above except specify LINES TO SCROLL = 16.

● **Modification 5** So far, the screen has been moved and a stationary bright spot used to etch a trace on it. With the sine waves, the curves were drawn first and then the whole or part of the screen was moved afterwards in order to animate the display. The next possibility is to move both screen and spot in succession. Some interesting effects are obtained if this movement is at right angles.

● **Lissajous Figures** A harmonograph is a fascinating machine to watch. It often consists of a pencil and paper oscillating at right angles to each other on razor-edge or glimbal bearings. The resulting drawings can be very beautiful, especially when coloured.

If the screen (paper) is oscillated right-left and a spot (pencil) is oscillated up/down then depending on their relative rates and amplitudes, different figures can be produced which, within the limits of TRS80 low resolution graphics, give a reasonable approximation to Lissajous Figures. There are many hours of enjoyment to be gleaned from trying different equations and playing about (experimenting) with the other variables.

The code for this is given in Listing 10. Note the ON ERROR GOTO ... essential if the user is not to be incessantly plagued by FC ERRORS. Lines 950-980 control vertical movement, while lines 990-1010 look after the screen. Figure 5 gives some examples of the kind of thing produced. Specify full screen movement and fastest speed initially.

● **Modification 6** A chart recorder is used mainly for simulation purposes. The top half of the screen becomes a mobile recording sheet and the lower half is used for the demonstration program that generates the data to be recorded. The example given here, Listing 11, is a colorimeter being used to measure the increase in turbidity of a liquid growth medium as a population of bacteria develops.

A colorimeter is a device that measures the amount of light penetrating a liquid. The darker the colour of the liquid or the more material suspended in it, the more light is absorbed.

The light is simulated by an oscillating pixel moving from right

```
10 REM LISTING 2 MODIFICATION 6
15 REM INSERT NEW ROUTINE FROM 900-1180
!
!
!
!
!
!
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
900 REM CHART RECORDER AND COLORIMETER SIMULATION
910 CLEAR200
920 CLS
930 PO=1
940 K=2
950 A$=STRING$(64,191):B$=STRING$(5,191)
960 PRINT@0,A$;:PRINT@0+(8*64),A$;
970 FORA=0TO8:PRINT@0+(A*64),B$;:PRINT@59+(A*64),B$;:NEXTA
980 PRINT@0+(9*64),A$;
990 PRINT@0+(3*64)+8," CHART RECORDER AND COLORIMETER DEMONSTRATION ";
1000 PRINT@0+(10*64),A$;
1010 FORA=10TO14:PRINT@0+(A*64),B$;:NEXT
1020 PRINT@0+(14*64),A$;
1030 FORA=10TO14:PRINT@59+(A*64),B$;:NEXT
1040 FORA=1TO10:X1=RND(117)+6:Y1=RND(8)+33:SET(X1,Y1):NEXT
1050 FORX=117TO6STEP-1
1060 FORY=34TO41
1070 IFPOINT(X,Y)THEN1090ELSEN=N+1
1080 SET(X,Y):RESET(X,Y)
1090 NEXTY
1100 NEXTX
1110 FORB=1TON/50:SET(12,23-B):NEXT
1120 POKE-4,64:POKE16526,130:POKE16527,254:Z=USR(0)
1130 GE=GE+1
1140 PT=K+GE
1150 FORC=1TOPT-PO:X1=RND(117)+6:Y1=RND(8)+33:SET(X1,Y1):NEXT
1160 PO=PT
1170 N=0
1180 GOTO1050
```

Listing 11 (Above)

```
10 REM LISTING 2 MODIFICATION 7
15 REM INSERT NEW ROUTINE FROM 900-2070
!
!
!
!
!
!
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
900 REM PRIMITIVE TURTLE(!) GRAPHICS
!
!
920 CLS
930 A$=STRING$(3,128)+CHR$(160)+CHR$(176)+STRING$(2,188)+CHR$(176)
940 B$=CHR$(176)+CHR$(188)+STRING$(6,191)+CHR$(189)+CHR$(176)+CHR$(184)+CHR$(143)+CHR$(141)
950 C$=CHR$(128)+CHR$(136)+CHR$(142)+CHR$(129)+STRING$(2,128)+CHR$(136)+CHR$(142)+CHR$(129)
2000 PRINT@341,A$;
2010 PRINT@405,B$;
2020 PRINT@469,C$;
2030 FORA=1TO360STEP3:X=40+(COS(A*0.0174)+1)*15:Y=12+(SIN(A*0.0174)+1)*7.5:SET(X,Y):NEXT
2040 POKE16526,130:POKE16527,254
2050 P=PEEK(14400)
2060 POKE-4,P:Z=USR(0)
2070 GOTO2050
```

Listing 12 (Top right)

```
10 REM LISTING 2 MODIFICATION 7A
15 REM 2050,2060 CHANGED:910,970-1020,2065 INSERTED
!
!
!
!
!
!
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
900 REM PRIMITIVE TURTLE(!) GRAPHICS FROM NUMERICAL ARRAY
910 DIMP(48)
920 CLS
930 A$=STRING$(3,128)+CHR$(160)+CHR$(176)+STRING$(2,188)+CHR$(176)
940 B$=CHR$(176)+CHR$(188)+STRING$(6,191)+CHR$(189)+CHR$(176)+CHR$(184)+CHR$(143)+CHR$(141)
950 C$=CHR$(128)+CHR$(136)+CHR$(142)+CHR$(129)+STRING$(2,128)+CHR$(136)+CHR$(142)+CHR$(129)
970 FOR MO=1TO48:READMV:P(MO)=MV:NEXT
980 DATA 32,32,32,32,32,32,32,32,32,32
990 DATA 8,8,8,8
1000 DATA 64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64
1010 DATA 16,16,16,16
1020 DATA 32,32,32,32,32,32,32,32,32,32
2000 PRINT@341,A$;
2010 PRINT@405,B$;
2020 PRINT@469,C$;
2030 FORA=1TO360STEP3:X=40+(COS(A*0.0174)+1)*15:Y=12+(SIN(A*0.0174)+1)*7.5:SET(X,Y):NEXT
2040 POKE16526,130:POKE16527,254
2050 FOR B=1TO48
2060 POKE-4,P(B):Z=USR(0)
2065 NEXTB
2070 GOTO2050
```

Listing 13 (Right)

to left. The bacteria are pixels in the pathway. They are increasing exponentially. The program sends the oscillating pixel across the solution. POINT is used to detect the presence of a lighted pixel (=1 bacterium). After a complete pass the total count is scaled and plotted on the chart recorder as a vertical line. The recorder is then moved to the right, reproduction occurs and the next beam of light sweeps through the solution. It is worth trying different values of K in line 940. K=2 means that the population doubles each generation.

PARAMETER		
WINDOW NUMBER	1	(ENTER)
LINES DOWN	2	(ENTER)
BYTES FROM LEFT	6	(ENTER)
LINE LENGTH	52	(ENTER)
LINES TO SCROLL	7	(ENTER)
DELAY	1	(ENTER)

Although many other experiments can be simulated, the point is that partial screen scrolling gives the would-be animator an extra useful facility.

● **Modifications 7-7C** From one and two directional movement it is a short step to multidirectional options. The program in Listing 12 draws a turtle on the screen and then waits for the user to press the ARROW keys. This demonstrates that large graphics units can be moved smoothly all over the screen. Whole screen parameters and fastest speed are needed for all these "Turtle Graphics" examples.

The simplest way to obtain continuous multidirectional movement under program control is to use an array to store the moves. This data can be passed into the POKEIT buffer sequentially. Listing 13 shows how this can be implemented to move the turtle along a rectangular pathway. The version in Listing 14 is much more user-friendly. Data is input as U,D,L and R. The program then translates these into the corresponding numbers.

Finally, in the interests of space saving, Listing 15 line 960, has the same data in compressed form.

9L1L4U.......means 9 moves Left, 1 move Left, 4 moves Up....

To reduce things to their simplest, any number of moves greater than nine must be stated indirectly. Thus:

20 moves Left=9L9L2L = 3 sets of moves 9L+9L+2L

The two-dimensional array in Line 910 will hold 48 separate sets of moves. The string DT$ represents nine sets. There is no reason why the array should not be enlarged to cope with more complex movements.

● **Modifications 8 - 9**
So far, all the examples have involved single windows. It is also possible to have many windows scrolling and rotating sequentially under both manual and program control. The final

```
10 REM LISTING 2 MODIFICATION 7B
15 REM 970-1080 CHANGED
:
:
:
970 FOR MO=1TO48
980 READMV$
990 IFMV$="U"THENP(MO)=8
1000 IFMV$="D"THENP(MO)=16
1010 IFMV$="L"THENP(MO)=32
1020 IFMV$="R"THENP(MO)=64
1030 NEXT
1040 DATA L,L,L,L,L,L,L,L,L,L
1050 DATA U,U,U,U
1060 DATA R,R,R,R,R,R,R,R,R,R,R,R,R,R,R,R,R,R,R,R
1070 DATA D,D,D,D
1080 DATA L,L,L,L,L,L,L,L,L,L
```

Listing 14

```
10 REM LISTING 2 MODIFICATION 7C
15 REM 1040-1080 DELETED:910,960-1020,2050-2065 CHANGED
:
:
900 REM PRIMITIVE TURTLE(!) GRAPHICS FROM STRINGS->2D NUMERICAL ARRAY
910 DIMP(48,2)
:
:
960 DT$="9L1L4U9R9R2R4D9L1L"
965 I=0
970 FORMO=1TOLEN(DT$)STEP2
975 I=I+1
980 P(I,1)=VAL(MID$(DT$,MO,1))
985 MV$=MID$(DT$,MO+1,1)
990 IFMV$="U"THENP(I,2)=8
1000 IFMV$="D"THENP(I,2)=16
1010 IFMV$="L"THENP(I,2)=32
1020 IFMV$="R"THENP(I,2)=64
1030 NEXT
:
:
:
2050 FOR B=1TOI
2055 FORB1=1TOP(B,1)
2060 POKE-4,P(B,2):Z=USR(0)
2062 NEXTB1
2065 NEXTB
```

Listing 15

```
10 REM LISTING 2 MODIFICATION 8
15 REM INSERT NEW ROUTINE FROM LINES 720-870
:
:
710 WI(R,C,0)=L:WI(R,C+1,0)=M:C=C+2:RETURN
720 REM DEMONSTRATION MULTIPLE WINDOW SCROLLING-MANUAL VERSION
730 CLS
740 FORA=15488TO16383:RR=RND(159)+32:POKEA,RR:NEXT
750 PRINT@0,CHR$(30)::INPUT"STATE WINDOW TO SCROLL";SW
790 REM PUT THE PARAMETERS FOR SELECTED WINDOW INTO BUFFERS
800 ME=96:FORC=1TO26:POKE-ME,WI(SW,C,0):ME=ME-1:NEXT
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
820 PRINT@0,CHR$(30);"ARROW KEYS MOVE WINDOW:SPACEBAR TO GET NEXT WINDOW";
830 POKE16526,130:POKE16527,254
840 P=PEEK(14400)
850 IFP=128THEN750
860 POKE-4,P:Z=USR(0)
870 GOTO840
```

Listing 16

```
10 REM LISTING 2 MODIFICATION 8A
15 REM 820,850 DELETED:740,870 CHANGED:725,750-760,865 INSERTED
:
:
:
720 REM DEMONSTRATION MULTIPLE WINDOW SCROLLING-MANUAL VERSION
725 REM CAN ACCESS WINDOWS BY SCANNING THE RAM BUFFER 16438
730 CLS
740 FORA=15360TO16383:RR=RND(159)+32:POKEA,RR:NEXT
750 FORBU=0TO3:SQ=PEEK(16438+BU)
755 IFSQ=0THEN830
760 SW=INT(BU*8+(LOG(SQ)/0.69))
790 REM PUT THE PARAMETERS FOR SELECTED WINDOW INTO BUFFERS
800 ME=96:FORC=1TO26:POKE-ME,WI(SW,C,0):ME=ME-1:NEXT
810 POKE-102,WI(SW,27,0):POKE-101,WI(SW,28,0)
830 POKE16526,130:POKE16527,254
840 P=PEEK(14400)
860 POKE-4,P:Z=USR(0)
865 NEXT
870 GOTO750
```

Listing 17

set of examples show how this might be done.

The modification given in Listing 16 allows the user to INPUT the window number to be scrolled. Answer the "NUMBER OF WINDOWS REQUIRED" prompt with a number > 1 and then input their parameters as usual. Use the ARROW keys to move the current window and the spacebar to enter another window choice.

To change the window manually, with the minimum of program interruption, the method shown in Listing 17 (lines 750-760) can be employed. These lines need some further explanation.

When a key is pressed, the bit pattern is copied into one of seven addresses in RAM according to which row the key is in. This RAM BUFFER occupies addresses 16438 to 16444. The table below shows how they are mapped to the keyboard rows.

TABLE 4

Address	Keyboard	Row
16438	@	G
16439	H	O
16440	P	W
16441	X	Z
16442	0	7
16443	8	/
16444	Enter	Spacebar

For example, pressing the letter A will cause a 2 to appear in 16438. As all these addresses have a "resting state" of zero, any non-zero value means a key in the corresponding row has been pressed.

If letters instead of numbers are used to request windows, any one of 26 windows can be called at the press of a single alphabet key. Scanning the four addresses 16438-16441 for a non-zero value will reveal the row containing the pressed key. Line 760 decodes the value PEEKed into a number between 1 and 26 reflecting the position in the alphabet of the key pressed. This value (SW) is then used to index into the WI array in line 800, causing the parameters for the requested window to be accessed. Pressing A gives window 1, B gives window 2 and so on. If you want more than 26 windows you're on your own!

Finally, the program in Listing 18 permits the number of moves and the associated directions to be entered by means of DATA strings (line 820). The values are stored in the hitherto unused array elements WI(R,C,1) and WI(R,C,2). By entering the parameters given in Table 5 for 12 windows, a graphics block (it's a Dalek) manouevres around an obstacle course.
The general idea is very simple. By specifying several windows which have overlapping parameters, the contents of one window can be passed to another rather like a conveyor belt. Figure 6 shows the overlapping areas in this example. To reverse the movement, complementary sets of instructions are used (line 820). At the same time, note that the parameters for windows 1-6 are the same as those for 12-7. Any amount

TABLE 5

WINDOW NUMBER NW	LINES DOWN LD	BYTES LEFT SB	LINE LENGTH LL	SCROLL LINES SL	DELAY DE	INSTRUCTIONS
1	1	0	32	5	1	9R9R1R
2	1	17	15	10	1	5D
3	6	17	34	5	1	9R9R1R
4	6	39	13	11	1	6D
5	12	5	48	5	1	9L9L9L7L
6	9	5	13	8	1	3U
7	9	5	13	8	1	3D
8	12	5	48	5	1	9R9R9R7R
9	6	39	13	11	1	6U
10	6	17	34	5	1	9L9L1L
11	1	17	15	10	1	5U
12	1	0	32	5	1	9L9L1L

```
10 REM LISTING 2 MODIFICATION 9
15 REM INSERT NEW ROUTINE FROM 690-1130
'
'
'
'
680 WI(R,28,0)=DM
685 REM USING MULTIPLE WINDOWS FOR ANIMATION
690 CN=0
700 READDT$
710 WI(R,0,1)=LEN(DT$)/2
720 FORMO=1TOLEN(DT$)STEP2
730 CN=CN+1
740 WI(R,CN,1)=VAL(MID$(DT$,MO,1))
750 MV$=MID$(DT$,MO+1,1)
760 IFMV$="U"THENWI(R,CN,2)=8
770 IFMV$="D"THENWI(R,CN,2)=16
780 IFMV$="L"THENWI(R,CN,2)=32
790 IFMV$="R"THENWI(R,CN,2)=64
800 NEXTMO
810 NEXTR
820 DATA 9R9R1R,5D,9R9R1R,6D,9L9L9L7L,3U,3D,9R9R9R7R,6U,9L9L1L,5U,9L9L1L
830 GOTO850
840 WI(R,C,0)=L:WI(R,C+1,0)=M:C=C+2:RETURN
850 CLS
860 A$=STRING$(2,128)+CHR$(184)+CHR$(188)+STRING$(3,191)+CHR$(188)+CHR$(180)
```

Listing 18

```
870 B$=STRING$(2,128)+CHR$(187)+STRING$(5,191)+CHR$(183)
880 C$=CHR$(128)+CHR$(184)+STRING$(8,191)+STRING$(2,140)+CHR$(183)
890 D$=CHR$(128)+CHR$(187)+" DALEK "+CHR$(183)
900 E$=CHR$(128)+CHR$(187)+STRING$(7,191)+CHR$(183)
910 Z$=STRING$(31,153)
920 Z1$=STRING$(11,143)
930 Z2$=STRING$(18,159)
940 Z3$=STRING$(17,138)
950 Z4$=STRING$(5,175)
960 FORA=0TO3:PRINT@33+(A*64),Z$;:NEXT
970 PRINT@384,Z3$;
980 FORA=0TO8:PRINT@384+(A*64),Z4$;:NEXT
990 PRINT@660,Z2$;
1000 FORA=0TO10:PRINT@309+(A*64),Z1$;:NEXT
1010 PRINT@0,A$;:PRINT@64,B$;:PRINT@128,C$;:PRINT@192,D$;:PRINT@256,E$;
1020 POKE16526,130:POKE16527,254
1030 FORR=1TONW
1040 ME=96:FORC=1TO26:POKE-ME,WI(R,C,0):ME=ME-1:NEXT
1050 POKE-102,WI(R,27,0):POKE-101,WI(R,28,0)
1060 FORC=1TOWI(R,0,1)
1070 FORC1=1TOWI(R,C,1)
1080 POKE-4,WI(R,C,2):Z=USR(0)
1090 NEXTC1
1100 NEXTC
1110 NEXTR
1120 FORT=1TO1000:NEXT
1130 GOTO1030
```

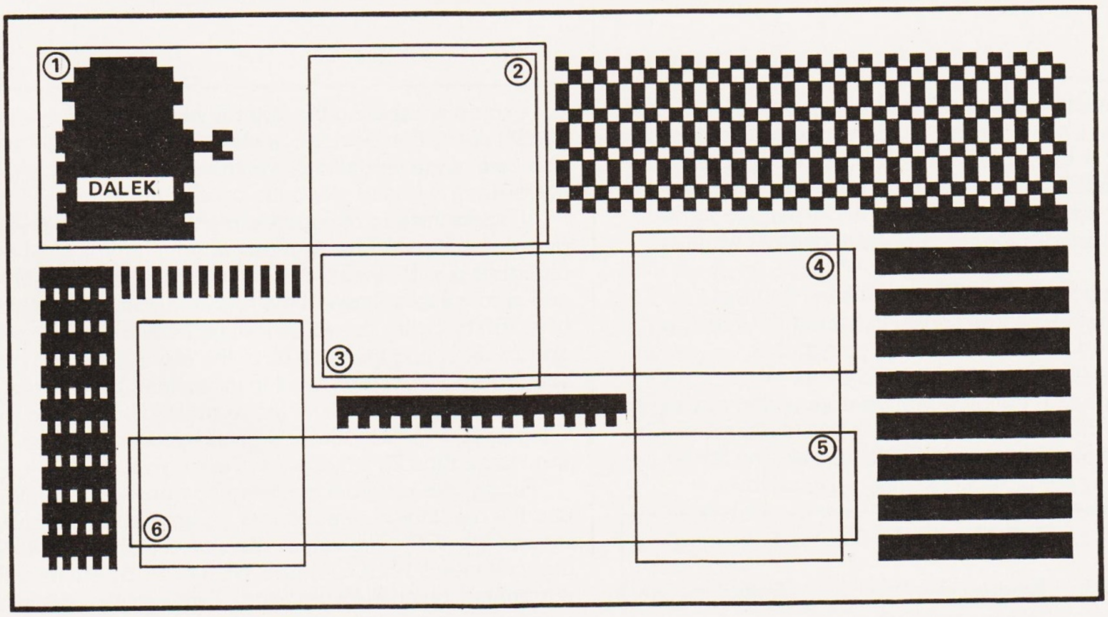


Fig. 6 Overlapping areas for Dalek obstacle course

of overlap is possible between adjacent windows.

The illustrations given here by no means exhaust the options for screen scrolling. It is appreciated that in one or two cases, the same effect can be created in other ways, but the intention has been to try to show that screen scrolling and rotation can have its uses, especially in the field of simple animation.

BOOK PAGE

Garry Marshall

The use and application of computers — a wide ranging topic. Our reviewer assesses four very different books.

Microman by Gordon Pask and Susan Curran is a beautifully produced book which shows that attractively designed and well structured books can be produced at a reasonable price. There are credits for an editor, design and art director, picture researchers and others and their joint effort is impressive.

Gordon Pask has contributed considerably to developments in cybernetics and in knowledge-based systems and this book reflects his work, ideas and experience in these fields. The Microman of the title is man enhanced by the creative, non-threatening uses of the micro that have become possible as a result of his work and the work of others.

The introduction claims that the book is not about computers but about the developing relationship between people and computers. About how we have shaped computers and, subsequently, how computers are shaping us and our environment. I remember vividly an electronics engineer saying to me, not too long ago, that the only use for computers was to design other computers. As the designs for computers have become increasingly complex, this has indeed become one essential function. Now, we also have factories full of robots building other robots but we have begun to learn how to take advantage of the power of computers in many other ways. This book describes many of those uses and the ideas behind them.

The book can be read as a layman's guide to how computers are used and how they are changing and enriching our lives: but it goes deeper than this. **Microman** has several strands running through it; the applications of computers, the changes they bring, the representation of knowledge, the nature of artificial intelligence and the likely future impact of computers. These strands interact with each other, so that themes recur in different chapters, rather like overlapping sliding panels, giving an apt analogy to the dynamic computational forms that are dealt with.

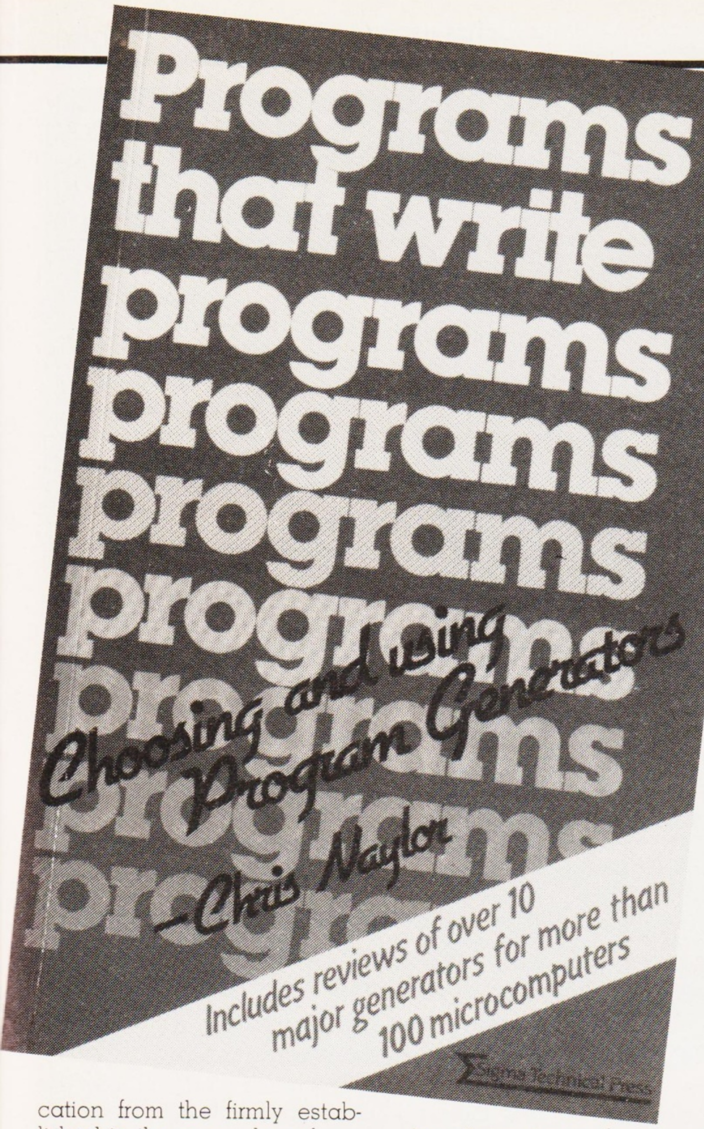
One chapter deals with computer networks and parallel processing, explaining how a number of computers linked together by a network, each performing computations at the same time can act in concert as one large computer. It then discusses how an expert system can exist in such a network with different computers carrying out different tasks, so that a problem is dealt with by subdividing it and passing each sub-problem to the appropriate specialist machine. The idea of a conflict, such as that which might arise when two parallel computations converge and interact in a way that has no clear resolution, is then introduced. This idea of conflict and the attempt to resolve it is developed as a basis of intelligent behaviour, and illustrated with reference to the game of 'Life' in an impressionistic way that requires no real knowledge of the technical issues involved in order to appreciate it.

An example of the way that computers have affected us at quite a deep level is provided by the way in which we think about artificial intelligence. Prior to the advent of computers, intelligence was thought of as something that was displayed by humans only and this naturally led to definitions of intelligence that involved humans and human characteristics. But in trying to make computers intelligent, discussions of machine intelligence cannot sensibly refer to definitions that involve humans and their characteristics. We must try to think more broadly than this. Not to do so could leave the unconstructive point of view that computers cannot be intelligent because they are not human.

Chapters on 'Language and knowledge' and 'Data structures and knowledge structures' discuss the ideas behind the representation and manipulation of knowledge in a computer and provide an explanation of the concepts of knowledge-based systems and some indication of what has been realised to date.

In a way, one of the middle chapters, 'The microprocessor in action', encapsulates the strengths of the book. The title of the chapter is not especially apt, for after describing altogether familiar applications of the microprocessor, ranging from the digital watch through uses in cars to robots, it suddenly launches into developments that are, in terms of the individual, much more important.

Xanadu is Ted Nelson's structured data bank that contains a vast amount of material, all with links and cross references to other items in the bank. This makes it a sort of dynamic reference library/encyclopaedia/word processor and as such it provides a new tool for accessing information, consultation and creating new material that is superior to all its conventional predecessors. Similarly, Nick Negroponte's Data Space coordinates computers, quadrophonic sound and video to create artificial experiences in a completely new way. Finally, the ways that the computer's ability to process knowledge can be harnessed in expert systems, computer-aided design and decision-making are described, so that the chapter has followed one strand of computer appli-



cation from the firmly established to the very edge of current developments.

There is a great deal to stimulate and inform the reader in this book, and it deserves careful reading. I would just note for the record that the material on micros is a little out of date, with photos of the ZX81 and Apple II (the book first appeared in hardback). Also the four picture researchers might have noticed that what purports to be a floppy disk in the illustration on page 26 is really a (rather rigid) video disk.

Chris Naylor's previous book, on expert systems, was about how to write one of your own to run on your micro. By way of contrast, **Programs That Write Programs** (subtitled Choosing and Using Program Generators) is not about how to write one, rather it is about what they are, what they are for and how to choose one from those that are available.

It is written in the same readable and racey style as the earlier book, in fact, the style is rather more appropriate this time. This book is basically aimed at the businessman who is about to buy, or has just bought, a micro and who needs to write programs for it to meet his individual needs. The book provides an entertaining and informative treatment of its topic, assuming no particular knowledge of its readers.

Aiming firmly at the businessman with no background in computing, the author begins his task of explaining about program generators by taking the line that unless you know something about computers, you are unlikely to be in a position to appreciate program generators and their potential uses. So the first part of the book is a rapid tour of how a computer works, what you can do with a computer and how to program it. This takes 50 pages so that it can offer only a broad sweep but it does this very effectively, extracting the principles involved quite effortlessly.

A program generator is a program that can write other programs. The first insight that we get is that any high-level language translator is a program generator, for it is a program that when given a high-level language program, writes the equivalent machine code program. This explains the presence of succinct treatments of BASIC and COBOL in the final section of the book which is devoted to reviews of program generators.

The concept of the application width of a program generator (ie the extent of the range of applications in which it is useful) is introduced. With any language system or program generator, the more closely it is tailored to a particular type of application the smaller its application width becomes. At one extreme is machine code, with the largest possible application width and at the other, any highly specialised applications package with only very specific uses.

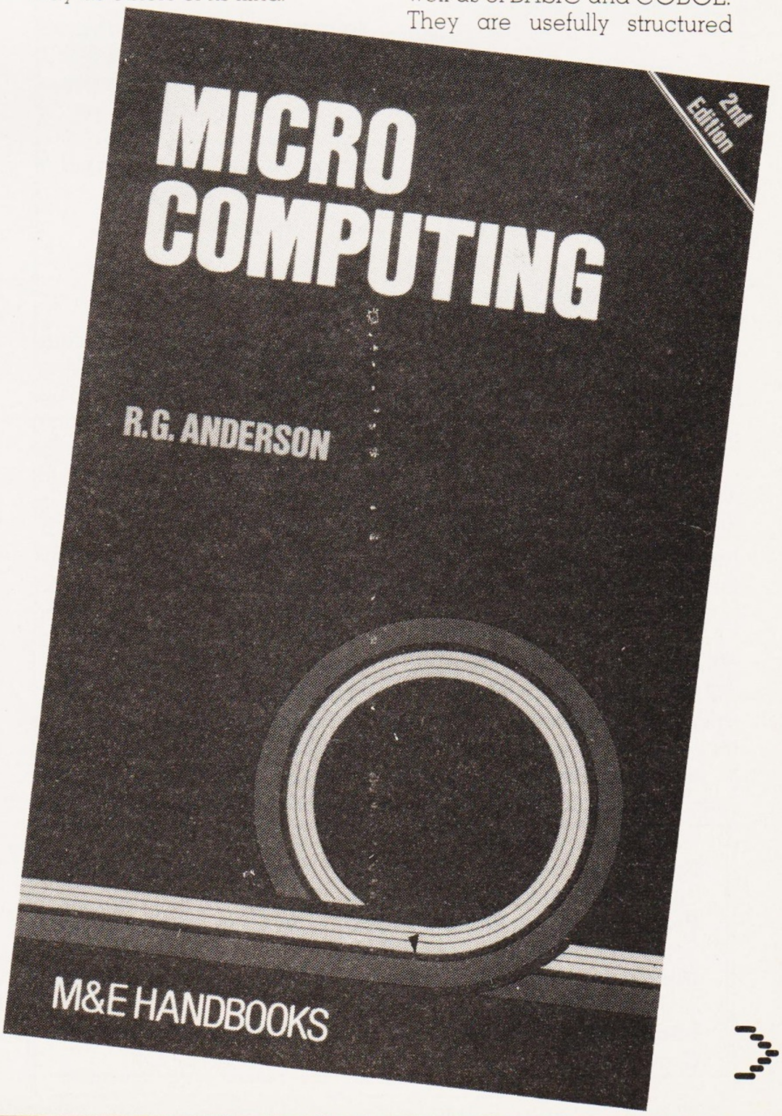
There is also a trade-off between the application width of a package and the ease with which one can learn to use it because the more features it has, the longer it will take to master them. Naylor points out that even though a narrowly applicable program will be comparatively easy to learn, even this learning will not be transferable unless the program is operated in much the same way as others of its kind.

What can be done with a computer is reduced to the four basic elements of input, output, manipulation and storage. At this level it is not at all difficult to acquire an appreciation of what can be done. Under the heading of storage, the treatment of files is very good. The general discussion of programming is also accurately aimed at its target audience, conveying certainly enough information, but not too much.

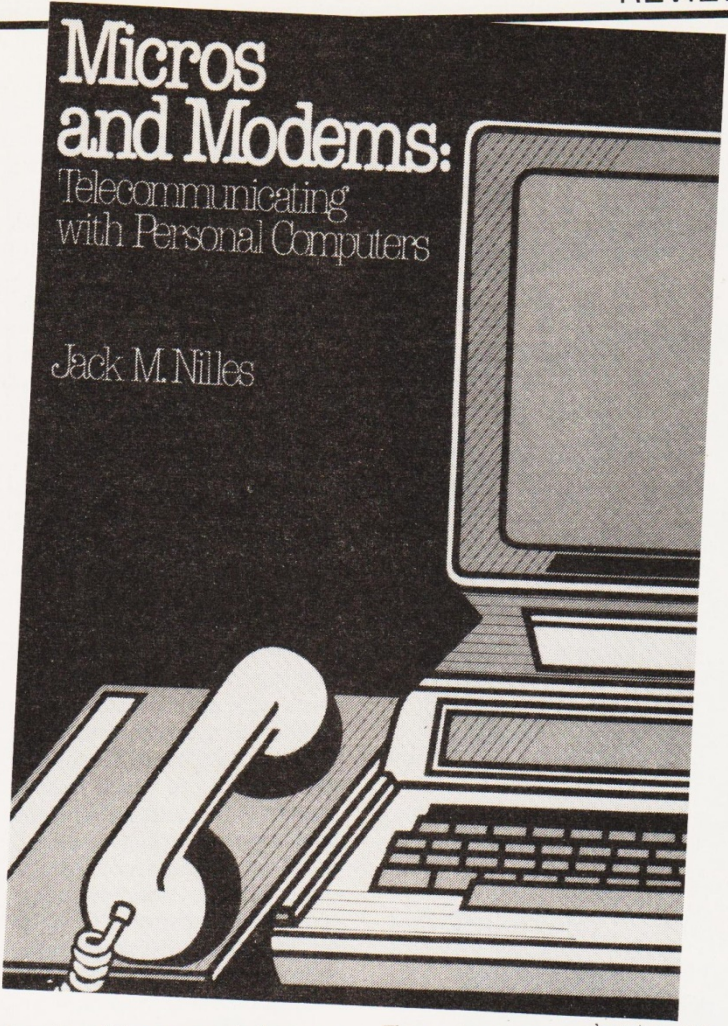
I think that this outline introduction to computing is more enlightening than many provided by entire books with this as their aim. It could be read with benefit by any newcomer to computers and computing.

The middle section of the book is a very brief summary of the first part, for those who think that they know enough to skip to it. I would suggest that everyone read the first part for its insights and skip the second, except that it does provide a useful glossary.

The final part of the book contains uniform reviews of nine program generators as well as of BASIC and COBOL. They are usefully structured



and succinct. The author does not attempt to select a 'best buy', for the generators are so varied in their capabilities and the position of the potential user on the learning curve so much a factor, that this is an impossibility. To give a brief flavour of the reviews, 'The last one' is described as 'a menu-driven method for writing COBOL programs in BASIC'. If that phrase arouses your curiosity, stimulates your imagination or just amazes you, then I suggest you buy the book.



I chose to review **Microcomputing** by R G Anderson because I thought it would provide a solid introduction to microcomputing from a slightly unusual angle. The book is written by an accountant, so that it could be expected to provide a treatment relevant to business computing: the author claims that it should be suitable for students studying for a wide range of professional institutional exams. That the book is from a respected series and in its second edition also suggests that it is should be good. I was sorely disappointed.

The core of the book covers elements of microcomputing and operating a microcomputer, then the concepts of programming and an outline of BASIC. This is a fairly conventional prospect but the treatment is disappointing and, in places, dubious. Statements such as the following do not really inspire confidence in the author:

'Some micros have an ASCII, ie QWERTY type standard keyboard' (p36).

'The Sharp MZ-80K (has) BASIC stored on cassette or disc which has to be loaded into ROM before processing is possible' (p45).

'Programs can . . . include READ statements, which requires the inclusion of DATA statements containing the variables (numbers) to be processed.' (p49-50).

In a list of the general characteristics and structure of BASIC, item(a) is 'Each instruction or statement is written on a separate line.' and item (z) is 'A string variable ends with a $ symbol.' (p67 and 69).

The cumulative effect of items like these is really depressing. An inability to distinguish between a variable and the name of a variable, not to mention the value associated with a variable, makes it difficult to progress far with BASIC.

The original edition of the book was based fairly closely on the PET. This influence persists so that a new chapter on local area networks fits rather uncomfortably with the older material and, to add to the tale of woe, unaccountably tails off into an account of Lisa, other similar systems and icons. There is a new chapter on graphics but to deal mainly with PET-style memory-mapped graphics in 1984 is a severe constraint.

There are some quite useful accountancy programs, but they do not extend to the use of files, so the reader is not taken very far into BASIC or its business applications. If the author has heard of program generators he gives no sign of it, but an awareness of program generators would be far more valuable to an accountant than the progams presented by the author. Definitely one to avoid.

Micros and modems by Jack Nilles is the most satisfactory book that I have seen on micros and communications. Using a micro to access Prestel and Micronet, for example, is becoming increasingly common. Nilles describes many similar uses that a communications capability makes possible and indicates a number that are likely to come in the future.

These include network information services, office automation, community bulletin boards, enhanced personal communication and entertainment facilities and telecommuting (that is, *not* commuting) to work and to school. All his predictions are firmly based on the idea that what is available in industry and business today is likely to be available and at a much reduced price, to the individual in the near future.

The book provides original ideas and new ways of looking at many topics because it deals with micros and communications in the broad context of the information environment and society in general and not just in their technical framework.

The first part of the book explains reasons why we should be interested in communicating with our micros (in case we are not). It deals in a non-technical way with micros and how to turn them into communications terminals and telecommunications systems (particularly the telephone network and broadcast systems but not neglecting cable systems and even satellite systems). It then goes on to discuss the interaction of people with the computer/communication system and the importance of making systems easy and natural to use. Ways in which this can be done are illustrated.

The book's second part is concerned with how to make micros communicate. How to make them communicate with each other, with mainframe computers, with other machines and even with people. A chapter each on hardware and software stresses strongly the interactions between the two. Communications software written in BASIC and in assembler is presented. The dialect of the BASIC is Cromemco Structured BASIC and although it is likely to differ from the versions available to you and me, it has the great merit of being easy to read and to understand.

The book concludes with chapters on communication networks with good treatment of local area networks and the developments that can result from linking micros with a network. The new jobs, the new styles of life and the new forms of entertainment that can result are explored and illustrated. The importance of these developments is not to be underestimated.

I can recommend **Micros and Modems** as an interesting and intelligent account of the developments themselves, the technology behind them and their consequences.

This month's books are:
Microman by Gordon Pask and Susan Curran (Century Publishing Co) 222 pages, £4.95
Programs That Write Programs by Chris Naylor (Sigma Technical press) 220 pages, £7.95
Microcomputing by R G Anderson (Macdonald and Evans) 210 pages, £3.50
Micros and Modems: Telecommunicating with Personal Computers by Jack Nilles (Reston) 170 pages, £14.40

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

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

Cut out and send to:

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

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

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

Please use BLOCK CAPITALS and include post codes.

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

. . . .

. . . .

Signature . . . .

Date . . . .

BACKNUMBERS

MARCH 1983
Colour Genie reviewed, Epson HX-20 review, PEEKing the Spectrum, Into Atari's BASIC, Terminology translated.

APRIL 1983
Froglet on the BBC Micro, PC-1251 hand-held review, Valley Variations, Galaxy reviewed, Micro Database, Lower case UK101.

MAY 1983
Spectrum Book Survey, Oric-1 Review, Going FORTH Again, Jupiter Ace review.

JUNE 1983
Interrupt handling, Rubic simulation on the Spectrum, Beating the RS232 Blues, Lynx review, Indexer.

JULY 1983
Atari renumber, 16-bit micros, Bomb-proof Tandy, Olivetti Praxis 30 review, Ikon Hobbit tape drive review.

AUGUST 1983
Speeding up the Sharp, Premier Dragon disc drive, Sord M5 review, BBC String Store, Planetfall.

SEPTEMBER 1983
FELIX knowledge shops, Software protection, Torch disc pack, ZX81 Backgammon, Dragon character generator, Three Tandy computers.

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

NOVEMBER 1983
BBC Word Processor, ZX LPRINT review, Laser 200 review, Writing Adventures, Learning FORTH Part 1, PET tape append.

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

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

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

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

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

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

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

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

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

A Computing today Reader Service

BACKNUMBERS

Please send me the following Backnumbers

ISSUE

At £1.40 each. I enclose £ ....

NAME ....

ADDRESS ....

....

....

.... POSTCODE ....

Signature ....

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

Access ☐ Barclaycard ☐

Insert Card No.

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

Please allow 21 days for delivery.

CT Aug '84

CHARTBUSTERS

BRITAIN'S SOFTWARE CHARTS

by the ASP Market Research Group

ARCADE

1	Jet Set Willy	Software Projects	Spectrum (2)
2	Sabre Wulf	Ultimate	Spectrum (-)
3	Fighter Pilot	Digital Integration	Spectrum (-)
4	Beach-Head	US Gold	CBM 64 (7)
5	Cosmic Cruiser	Imagine	CBM 64 (-)
6	Antics	Bug Byte	Spectgrum (-)
7	Les Flics	PSS	Spectrum (-)
8	Psytron	Beyond	Spectrum (3)
9	Cavelon	Ocean	CBM 64 (-)
10	Trashman	New Generation	Spectrum (-)

NON-ARCADE

1	Mugsy	Melbourne House	Spectrum (1)
2	Golf 64	Abrasco	Spectrum(-)
3	Flight Path 737	Anirog	CBM 64 (2)
4	Snooker	Visions	CBM 64 (4)
5	Fall of Rome	APS	Spectrum (3)
6	Fall of Rome	APS	CBM 64 (3)
7	Solo Flight	US Gold	CBM 64 (8)
8	Twin Kingdom Valley	Bug Byte	CBM 64 (6)
9	Classic Adventure	CDS	ZX 81 (-)
10	Blockbuster	Clever Clogs	Spectrum (-)

Compiled with the assistance of Britain's leading software distributors, including: Pinnacle, SDL, PCE, Websters, PCS and Software Centre.

SPECTRUM

1	Jet Set Willy	Software Projects (2)
2	Sabre Wulf	Ultimate (-)
3	Fighter Pilot	Digital (-)
4	Psytron	Beyond (3)
5	Jack and the Beanstock	Thor (-)
6	Antics	Bug Byte (-)
7	Les Flics	PSS (-)
8	Atic Atac	Ultimate (9)
9	Scrabble	Psion (-)
10	Trashman	New Generation (6)

COMMODORE 64

1	BMX Racers	Mastertronic (2)
2	Space Walk	Mastertronic (5)
3	Manic Miner	Software Projects (1)
4	Beach Head	Centresoft (-)
5	Snooker	Visions (-)
6	Black Hawk	Creative Sparks (3)
7	Colossus Chess	CDS (-)
8	Space Pilot	Anirog (4)
9	Chuckie Egg	A&F (-)
9	Revelation	Softek (-)

DRAGON 32

1	Buzzard Bait	Microdeal (-)
2	Cuthbert in Space	Microdeal (2)
3	Dragon Chessl	Oasis (3)
4	Hungry Horace	M. House (5)
5	Bug Diver	Mastertronic (-)
6	Eightball	Microdeal (4)
6	Spritemagic	Knight (-)
8	Dungeon Raid	Microdeal (-)
8	Skramble	Microdeal (8)
10	Mr Dig	Microdeal (-)

Compiled by W. H. Smith and Websters. Figures in brackets are last week's positions.

VIC-20

1	Duck Shoot	Mastertronic (1)
2	Tank Commander	Creative Sparks (3)
4	Snooker	Visions (-)
5	Computer War	Creative Sparks
6	Chariot Race	Micro Antics (4)
7	Vegas Jackpot	Mastertronic (-)
8	Phantom Attack	Mastertronic (-)
9	Sub Hunt	Mastertronic (-)
10	Games Designer	Galactic (-)

BBC

1	Aviator	Acornsoft (1)
2	Spitfire Command	Superior (9)
3	OverDrive	Superior (3)
4	Fortress	Pace (-)
5	JCB Digger	Acornsoft (2)
6	Snooker	Visions (-)
7	Battle Tank	Superior (4)
8	Chess	BBC (-)
9	Q Man	MRM (5)
10	Twin Kingdom Valley	Bug-Byte (-)

ZX81

1	Alien Reign	CRL (9)
2	Krypton Ordeal	Novus (2)
3	Planet Raider	Novus (4)
4	Walk the plank	Novus (3)
5	Black Crystal	Carnell (7)
6	Flight Simulation	Sinclair (8)
7	Mothership	Sinclair (-)
8	Reversi	Sinclair (-)
9	Sabotage	Sinclair (-)
10	City Patrol	Sinclair (-)

SINCLAIR, ATARI
COMMODORE

Walkers
COMPUTER SERVICE AND REPAIRS
Telephone: (021) 643 5474
Telegrams: Shiplog Birmingham
Telex 337124 TWSLOG G

COMPUTER REPAIRS
by the
SPECIALISTS

WALKERS will repair your MICRO quickly and efficiently.

Services available to suit the Customers requirements include "one-off" repairs and Quotations.

Most Repairs will be in the range
£15 to £30
subject to machine

FOR INFORMATION
PHONE 021-643 5474

CALL IN AT
58 OXFORD STREET, BIRMINGHAM, B5 5NX

OR SEND S.A.E. FOR DETAILS

DRAGON, BBC, SPECTRUM DEALERS

BBC B Computer 1.2 O.S.	**£399.00**
1.2 ROM	**£8.00**
Graphics Rom	**£33.00**
Microvitec Monitor	**£228.00**
Sanyo Green Monitors	**£97.00**
Disc Drives from:	**£185.00**
Joysticks (Pair)	**£17.90**
Wordwise Word Processor	**£39.00**
Acorn Electron	**£199.00**
All connectors, plugs and sockets for BBC, ribbon cable, discs C.20 C.15 C.12, cassettes etc. in stock.	
Centronics Printer Cable (BBC & Dragon)	**£12.90**
R.T.T.Y. Program for BBC B	**£7.50**
R.T.T.Y. Circuit Board including instructions	**£7.00**
R.T.T.Y. Eprom Version	**£20.00**
BBC/Slow Scan Program & Board	**£17.50**
Computer Dust Covers	**£3.00**
Star Gemini 10X Printer incl cable	**£275.00**
CP80 Printer (Inc. Cable)	**£230.00**
Printer Cable (BBC or Dragon 32)	**£12.90**
Epson RX80, FT, FX80	**(Phone availability)**
DRAGON SERVICE CENTRE	
Dragon 32	**£150.00**
Dragon 32 Disc Drive (Inc. Controller)	**£275.00**
Joysticks (pair)	**£14.90**
ZX Spectrum 48K	**£129.00**
Memotech 512	**£325.00**

Wide range of software for BBC, Dragon 32, ZX Spectrum etc. Please send SAE for full list. Post and package on small items £1. All available mail order. Access and Visa. 24 hour phone. All prices include VAT at 15%.

S P ELECTRONICS
48 Linby Road, Hucknall, Notts NG15 7TS.
TEL: Notts (0602) 640377

68000 SYSTEM KITS

We have a range of high quality, double sided, 8 by 8 PCB's which form the core of a 68000 system. Parts currently available are:

1) 68000 FOX CPU2 contains 68000 L8 micro with 8 mHz crystal clock, on board dynamics refresh sequencer, full bus buffering, full power up/manual reset (won't disturb RAM's) and 78 way gold edge connector bussing all CPU signals.

2) 68000 FOX BIO provides 8/16K boot EPROM, 4K RAM, 20 I/O lines (PIA), RS232 interface with 14 baud rates, 3 channel 16 bit counter-timer, full interrupt handling etc.

3) A boot loader/monitor is available in EPROMS.

Prices:

CPU2 PCB only	**£60**
BIO PCB only	**£60**
Component set for CPU2	**£110**
Component set for BIO	**£98**
Boot loader/monitor	**£25**

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

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

Superbase 64

TRANSFORMS THE COMMODORE 64 INTO A FULL-FEATURED AND PROFESSIONAL DATABASE SYSTEM! WITH UP TO 1000 CHARACTERS PER RECORD ON UP TO 4 SCREENS... AND UP TO 128 ITEMS PER RECORD, DEFINABLE AS KEY, TEXT, NUMERIC, CONSTANT, RESULT OR DATE... IN FILES OF UP TO 16M CHARACTERS! SUPERBASE 64 EVEN HAS SPREADSHEET AND CALCULATOR CAPABILITY, CALENDAR FUNCTIONS, EASY INPUT FROM WORDPROCESSOR/DATA FILES, BOTH MENU-DRIVEN AND PROGRAM OPTIONS, SORTING/SEARCHING, FULLY DEFINABLE OUTPUTS... SUPERBASE 64 IS ESSENTIAL IF YOU WANT THE MOST FROM YOUR 64! SUPPLIED ON CBM 1541 DISK WITH EXCELLENT TUTORIAL/REFERENCE MANUAL. OUR PRICE ~~£99.95~~ £88.00!

VIZAWRITE 64

VIZAWRITE 64 IS A HIGH-PERFORMANCE, LOW-COST WORD PROCESSOR, WITH ON-SCREEN FORMATTING, THAT TAKES FULL ADVANTAGE OF THE 64'S COLOUR, GRAPHICS AND MEMORY FEATURES... AND SUPPORTS VIRTUALLY ANY PRINTER! WITH A COMPREHENSIVE AND EASY-TO-FOLLOW USER MANUAL, VIZAWRITE IS THE ULTIMATE PERSONAL COMPUTER WORD PROCESSOR! AVAILABLE ON CARTRIDGE (~~£89.95~~ £75!), DISK (~~£79.95~~ £68!) OR WITH VIZASPELL (~~£99.95~~ £85!)

Master 64

MASTER 64 IS A TOTALLY NEW CONCEPT... A COMPLETE PROGRAM DEVELOPMENT PACKAGE, THAT'S AVAILABLE NOW FOR THE CBM 64. MASTER HAS 85 NEW COMMANDS... AND BASIC IV TOO! PLUS PROGRAMMER'S TOOLKIT, MACHINE CODE MONITOR, BUSINESS BASIC, KEYED DISK ACCESS, MULTIPLE SCREENS, USER-DEFINABLE INPUT ZONES, REPORT GENERATOR, 22-PLACE ARITHMETIC, DISK DATA COMPRESSION, DATE CONTROL, SCREEN PLOT, SCREEN DUMP, AND MORE... EVERYTHING YOU NEED TO PROGRAM YOUR 64 TO TOP PROFESSIONAL STANDARDS! (~~£143.75~~ £115!)

*** WANT IT TOMORROW? *** CALL US TODAY! *** ON 01-546-7256

VIZASPELL (DISK)	~~£59.95~~	£49.95	DTL COMPILER 64 (DISK)	~~£114.95~~	£109.00
EASYSCRIPT (DISK)	~~£75.00~~	£69.00	SUPERBASE (CBM 700)	~~£517.50~~	£425.00
EASYSPELL (DISK)	~~£50.00~~	£48.00	SUPERSCRIPT (CBM 700)	~~£517.50~~	£425.00
SIMON'S BASIC (CARTRIDGE)	~~£50.00~~	£48.00	SUPEROFFICE (8096/8296)	~~£799.25~~	£650.00
ASSEMBLER 64 (DISK)	~~£24.95~~	£23.00	SUPERBASE (8096/8296)	~~£454.25~~	£375.00
TOOL 64 (CARTRIDGE)	~~£49.95~~	£46.95	SUPERSCRIPT (8096/8296)	~~£454.25~~	£375.00
PRACTICALC (DISK/TAPE)	~~£44.50~~	£41.50	SUPERSCRIPT (3000-8000)	~~£286.35~~	£225.00
MULTIPLAN (DISK)	~~£79.95~~	£75.00	MASTER (4000-8296)	~~£339.25~~	£225.00

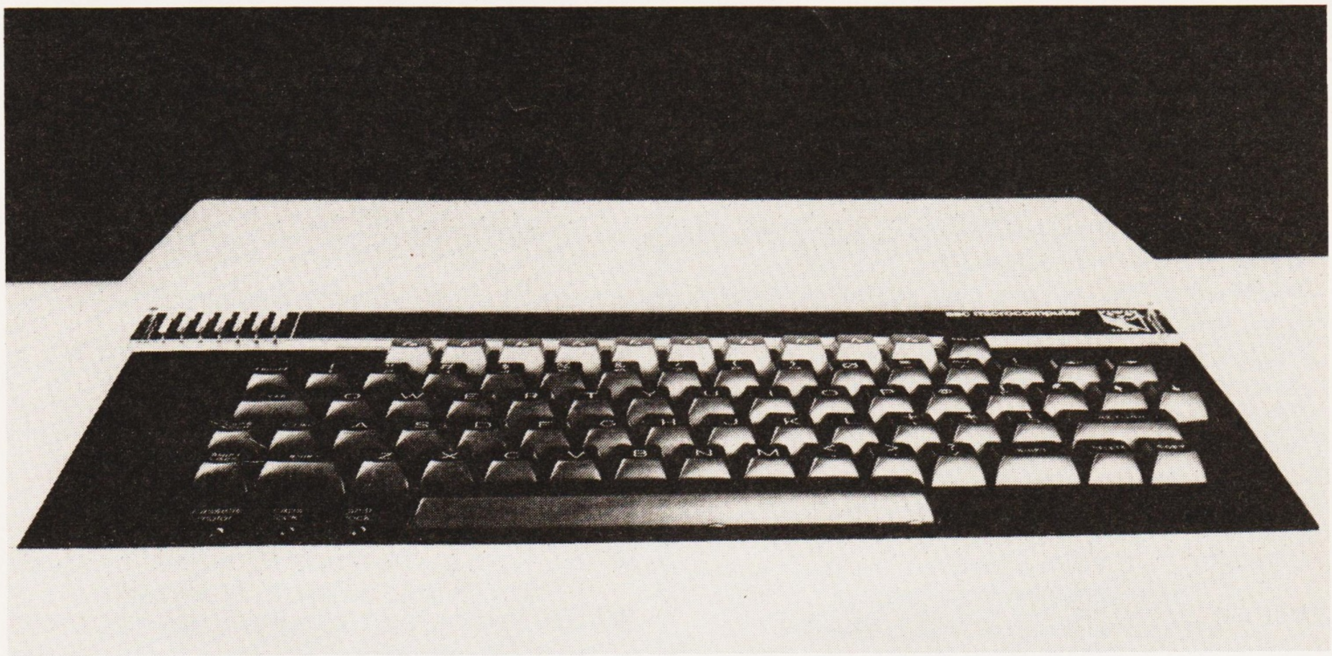
PRICES INCLUDE 15% VAT AND ARE CORRECT ON GOING TO PRESS. ORDER BY POST OR PHONE, USING CHEQUE, ACCESS/BARCLAY CARD OR OFFICIAL ORDER. DESPATCH IS BY SAME-DAY 1ST CLASS POST, PRE-PAID ORDERS FREE. PRODUCT DATA AVAILABLE ON REQUEST. REF A28

VISA

Calco Software

LAKESIDE HOUSE, KINGSTON HILL, SURREY, KT2 7QT. TEL 01-546-7256

FINDING AND KEEPING



G. W. Gallagher

Do you collect stamps, paintings, books, beermats or parking tickets? If you need to keep track of a multiplicity of objects, here's the program for you. Written for the BBC micro, it's easily converted for other machines.

One of the reasons given for buying a computer for use at home is that it will provide an excellent way of keeping records and accounts. Accounts have been dealt with on many occasions but there are other home uses which become apparent as one becomes addicted to using the computer.

In my own case, I could see three subjects for filing immediately:

- A means of cross-reference for composer, musicians, etc for a record collection.
- A record of useful articles seen in various computing magazines. (I could cut out the articles but am reluctant to mutilate magazines unless necessary.)
- A collection of thimbles which has grown rapidly enough to need a filing system for individual details but it could just as easily be a collection of stamps, coins or any other collectable item.

The program devised worked equally well for each requirement and can be used for any number of purposes. The example shown here is of the 'thimble' program.

DATA STORAGE

The information about each item is stored as one string, built up from as many sections as are required to contain the various pieces of information about each item. The use of a single string, instead of several separate pieces of data, is simply to save memory space. This is particularly important if the file is to be saved on cassette. In such a case, the strings will have to be taken into an array in memory before they can be added to or searched for information. The probable limit of the number of items which may be taken into the array is about 250 (which should be remembered when planning a cassette file for this program). There will be no such limit for a disc file, as the array is only used to hold temporarily new items to be added to the file. If the file is to be very large, it is probably safer to keep it on a separate disc to avoid space problems.

As the computer used was a BBC B, the working sections of the program are given as PROCs. For any other type of computer, these should be used as subroutines.

In this particular example (ie a collection of thimbles) four sectons are used for each item as follows:

- 16 spaces (for a maker's name)
- 20 spaces (for a set reference)
- 18 spaces (for the artist's name)
- 22 spaces (for a description)

Further sections can be added if required, as long as the length of each section is kept constant. For example, if in the first section, the name was less than 16 bytes long, blank spaces are added. This is necessary if the process of extracting information is to work.

It is convenient to make a note of the lengths of the sections in REM statements at the beginning of the program (see lines 30-80). The total length of the specified sections is 76 bytes and two bytes are used by the filing system to note in each record the type and length of the data enclosed. I have actually used 80 bytes as the length of the completed entry on the file. This means that since we are using a random access file, the 'pointer' which indicates the position in the file at any particular time, will be moved on 80 bytes from one record to the next.

Listing 1

```
   10 REM..THIMBRR
   20 DIMD$(300)
   30 REM...1...MAKER..(16)
   40 REM...2...SET..(20)
   50 REM...3...ARTIST(18)
   60 REM...4...DESCRIPTION..(22)
   70 REM...RECORD LENGTH 80
   80 PROCchoice
   90 ON C GOTO 100,190,430,490,500
  100 PROCfromfile:N=I-1:ADD=0:REM;CHANGE ON FIRST
 RUN TO N=0
  110 CLS:PRINT''"Type = when you have finished ad
ding    items."
  120 PRINT'"Press the return key after each entry
."
  130 CLS:ADD=ADD+1:D$(ADD)="":PROCfirst:IF N$="="
 THEN 160 ELSE 140
  140 PROCsecond:PROCthird:PROCfourth:PROCcheck
  150  GOTO 110
  160 PRINT''"These items will now be filed."
  170 IF N=0 THEN PROCnewfile ELSE PROCaddfile
  180 GOTO 80
  190 CLS:PRINT''"Type 1.  to look for a maker"
  200 PRINT'"      2.  to look for a set"
  210 PRINT'"      3.  to look for an artist"
  220 PRINT'"      4.  to look for a description"
  230  INPUTCS:IF (CS-1)*(CS-2)*(CS-3)*(CS-4)<>0 T
HEN 230
  240  ON CS GOTO 250,300,350,390
  250 PRINT''"The maker is?":INPUT C1$
  260  IF LEN(C1$)>16 THEN 270 ELSE 280
  270 C1$=LEFT$(C1$,16)
  280 VDU2:PRINT';C1$:PROClooking
  290 PROCWAIT:VDU3:GOTO 80
  300 PRINT''"The name of the set is?":INPUT C2$
  310 IF LEN(C2$)>22 THEN 320 ELSE 330
  320 C2$=LEFT$(C2$,22):CLS:PRINT'C2$
  330 VDU2:PRINT'C2$:PROClooking
  340 PROCWAIT:VDU3:GOTO 80
  350   PRINT''"The name of the artist is?":INPUT
C3$
  360  IFLEN(C3$)>18 THEN C3$=LEFT$(C3$,18)
  370 VDU2:PRINT'C3$:PROClooking
  380 PROCWAIT:VDU3:GOTO 80
  390  PRINT''"The desription you wish to find is?
":INPUT C4$
  400  IFLEN(C4$)>22 THEN C4$=LEFT$(C4$,22)
  410 VDU2:PRINT'C4$:PROClooking
  420 PROCWAIT:VDU3:GOTO 80
  430 CLS:PRINT'"What is the number of the thimble
?"
  440 INPUT number :ADD=1:N=0
  450 PROCchanging:PROCamending:IF R=3 THEN 480
  460 IF C$="Y" OR C$="y"THEN 480 ELSE 470
  470 PROCamending:GOTO 460
  480 PROCaddfilesingle:GOTO 80
  490 VDU2:PROClist:VDU3:PROCWAIT:GOTO 80
  500  END
  510 DEFPROCfirst
  520 PRINT'"The name of the maker or manufacturer
?"
  530  INPUT N$:IF N$="=" THEN 580
  540 D$(ADD)=D$(ADD)+N$
  550 IF LEN(D$(ADD))>15 THEN 570 ELSE 560
  560 D$(ADD)=D$(ADD)+" ":GOTO 550
  570 D$(ADD)=LEFT$(D$(ADD),16)
  580 ENDPROC
  590 DEFPROCsecond
  600 PRINT'"The name of the set"
  610 INPUT N$:D$(ADD)=D$(ADD)+N$
  620 IF LEN(D$(ADD))>35 THEN 640 ELSE 630
  630 D$(ADD)=D$(ADD)+" ":GOTO 620
  640 D$(ADD)=LEFT$(D$(ADD),36)
  650 ENDPROC
  660 DEFPROCfourth
  670 PRINT'"The description of the thimble?":INPU
T N$:D$(ADD)=D$(ADD)+N$
  680 IF LEN(D$(ADD))>75 THEN 700 ELSE 690
  690 D$(ADD)=D$(ADD)+" ":GOTO 680
  700 D$(ADD)=LEFT$(D$(ADD),76)
  710 ENDPROC
  720 DEFPROCcheck
  730 PRINT; N+ADD,D$(ADD)
  740 PRINT'"Is this correct?(Y/N)"
  750 INPUT C$:IF C$="Y" OR C$="y" THEN 780
  760 IF C$="N" OR C$="n" THEN 770 ELSE 750
  770 D$(ADD)="":ADD=ADD-1
  780 ENDPROC
  790 DEFPROCfromfile
  800 X=OPENIN("thimble")
  810 I=0:PX=PTR#X
  820 REPEAT
  830 PTR#X=PX
  840 I=I+1
  850  INPUT#X, D$
  860 PX=PX+80
  870 UNTIL EOF#X
  880 CLOSE#X
  890 ENDPROC
  900 DEFPROCnewfile
  910 X=OPENOUT("thimble")
  920 PX=PTR#X
  930 FOR I=1 TO N+ADD
  940 PTR#X=PX
  950 PRINT#X,D$(I)
  960 PX=PX+80
  970 NEXT
  980 CLOSE#X
  990 ENDPROC
 1000 DEFPROCchoice
 1010 CLS:PRINT''"Type 1  to add items"
 1020 PRINT'"      2  to extract items"
 1030 PRINT'"      3  to correct or remove items"
 1040 PRINT'"      4  for a complete list"
 1050 PRINT'"      5  to end"
 1060 INPUT C:IF(C-1)*(C-2)*(C-3)*(C-4)*(C-5)<>0 T
HEN 1060
 1070 ENDPROC
 1080 DEFPROClooking
 1090 X=OPENIN("thimble")
 1100 I=0:PX=PTR#X
 1110 REPEAT
 1120 PTR#X=PX
 1130 I=I+1
 1140 INPUT#X, D$
 1150 IF CS>1 THEN 1170 ELSE PROClookfirst
 1160 GOTO 1220
 1170 IF CS>2 THEN 1190 ELSE PROClooksecond
 1180 GOTO 1220
 1190 IF CS>3 THEN 1210 ELSE PROClookthird
 1200 GOTO 1220
 1210 PROClookfourth
 1220 PX=PX+80
 1230 UNTIL EOF#X
 1240 CLOSE#X
 1250 ENDPROC
 1260 DEFPROClookfirst
 1270 L=LEN(C1$):IF L=16 THEN1310
 1280 J=1
 1290 IF MID$(D$,J,L)<>C1$ THEN 1300 ELSE 1320
 1300 J=J+1:IF J<17-L THEN 1290 ELSE 1330
 1310  IF LEFT$(D$,16)<>C1$ THEN 1330 ELSE 1320

 1320 PRINT;I;TAB(6);MID$(D$,17,20);TAB(30);MID$(D
$,37,18);TAB(50);RIGHT$(D$,22)
 1330 ENDPROC
 1340 DEFPROClooksecond
 1350 L=LEN(C2$):IF L=20 THEN1310
 1360 J=1
 1370 IF MID$(D$,J+16,L)<>C2$ THEN 1380 ELSE 1400
 1380 J=J+1:IF J<21-L THEN 1370 ELSE 1410
 1390 IF MID$(D$,17,20)<>C2$ THEN 1410 ELSE 1400

 1400 PRINT;I;TAB(6);LEFT$(D$,16);TAB(30);MID$(D$,
37,18);TAB(50);RIGHT$(D$,22)
 1410 ENDPROC
 1420 DEFPROClookthird
 1430 L=LEN(C3$):IF L=18 THEN 1470
 1440 J=1
 1450 IF MID$(D$,36+J,L)<>C3$ THEN 1460 ELSE 1480

 1460 J=J+1:IF J<19-L THEN 1460 ELSE 1490
 1470 IF MID$(D$,36,L)<> C3$ THEN 1490 ELSE 1480
 1480 PRINT;I;TAB(6)LEFT$(D$,16);TAB(30);MID$(D$,1
7,20);TAB(52)RIGHT$(D$,22)
 1490 ENDPROC
 1500 DEFPROCWAIT
 1510 X=GET:IFX<>32 THEN 1510
 1520 ENDPROC
 1530 DEFPROCthird
 1540 PRINT'"The name of the artist"
 1550 INPUT N$:D$(ADD)=D$(ADD)+N$
 1560 IF LEN(D$(ADD))>53 THEN 1580 ELSE 1570
 1570 D$(ADD)=D$(ADD)+" ":GOTO 1560
 1580 D$(ADD)=LEFT$(D$(ADD),54)
 1590 ENDPROC
 1600 DEFPROCaddfile
```

```
1610 X=OPENUP("thimble")
1620 PX=PTR#X+N*80
1630 FOR I=1 TO ADD
1640 PTR#X=PX
1650 PRINT#X,D$(I)
1660 PRINTD$(I)
1670 PX=PX+80
1680 NEXT
1690 CLOSE#X
1700 ENDPROC
1710 DEFPROClookfourth
1720 L=LEN(C4$):IF L=22 THEN 1760
1730  J=1
1740 IF MID$(D$,56+J,L)<>C4$ THEN 1750 ELSE 1770

1750 J=J+1:IFJ<23-L THEN 1740 ELSE 1780
1760 IF RIGHT$(D$,22)<>C4$ THEN 1780 ELSE 1770
1770 PRINT;I;TAB(6)LEFT$(D$,16);TAB(30);MID$(D$,1
7,20);TAB(52);RIGHT$(D$,22)
1780 ENDPROC
1790 DEFPROCamending
1800 PRINTD$
1810 PRINT''"Do you wish to:"
1820 PRINT'"1.  change this record"
1830 PRINT'"2.  cancel this record"
1840 PRINT'"3.  leave it unchanged"
1850 INPUT R:IF (R-1)*(R-2)*(R-3)<>0 THEN 1850
1860  ON R GOTO 1880,1870,1890
1870 D$="ZZZZ"+STRING$(72," "):ENDPROC
1880 D$(ADD)="":PROCfirst:PROCsecond:PROCthird:PR
OCfourth:PROCcheck1
1890 ENDPROC
1900 DEFPROCchanging
1910 X=OPENIN("thimble")
1920 PX=(number-1)*80
1930 PTR#X=PX
1940 INPUT#X,D$
1950 CLOSE#X
1960 ENDPROC
1970 DEFPROCaddfilesingle
1980 X=OPENUP("thimble")
1990 PX=(number-1)*80
2000 PTR#X=PX
2010 PRINT#X,D$(ADD)
2020 CLOSE#X
2030 ENDPROC
2040 DEFPROCcheck1
2050 PRINT;number,D$(ADD)
2060 PRINT'"Is this correct?(Y/N)"
2070 INPUT C$:IF C$="Y" OR C$="y" THEN 2100
2080 IF C$="N" OR C$="n" THEN 2090 ELSE 2070
2090 D$(ADD)=""
2100 ENDPROC
2110 DEFPROClist
2120 X=OPENIN("thimble")
2130 I=0:PX=PTR#X
2140 REPEAT
2150 PTR#X=PX
2160 I=I+1
2170 INPUT#X,D$
2180 PRINT;I;D$
2190 PX=PX+80
2200 UNTIL EOF#X
2210 CLOSE#X
2220 ENDPROC
```

FORMING THE SECTIONS

The PROCs used are as follows:

- **PROCfirst** (510-580) takes in the information for the first section and makes sure that it is the correct length. It is in this section that there is the chance to stop adding items by typing in '=' instead of any other data.
- **PROCsecond** (590-650) takes in the data for the second section and corrects it for length if necessary.
- **PROCthird** (1530-1590) repeats the process for the third section.
- **PROCfourth** (660-710) repeats the process for the fourth section.
- **PROCcheck** (720-780) gives the opportunity to check the string once the sections have been added together. If the string is correct, it is stored in the array as D$(ADD). If it is not correct, the string is scrapped and the process repeated.

The variables used at this stage are:

D$ The array which stores the new records before they are added to the file.
N The number of records which are already on the file.
ADD The number of items added at any one time.

EXTRACTING INFORMATION

It is possible to search for information included in any section of the string without necessarily giving the complete section for checking. For example, if I wished to obtain a list of thimbles, all of which contained the word 'rose' as part of the description, I cold do so by using the MID$ string function. Taking L to be the length of 'rose', ie four bytes, the program will check through each of the four sections in turn, moving along one byte at a time, until all the possible consecutive groups of four bytes have been checked. Thus the word 'rosette' and the phrase 'yellow rose' would each be picked out. The PROCs used are:

- **PROClookfirst** (1260-1330) checks the first section.
- **PROClooksecond** (1340-1410) checks the second section.
- **PROClookthird** (1420-1490) checks the third section.
- **PROClookfourth** (1710-1780) checks the fourth section.

If you have included more than four sections then this list of PROCs must be extended to cover the extra ones used.

ALTERATIONS TO RECORDS

The order of the records on the file is important to my filing system as each thimble or LP record sleeve has on it a label with the same number as that given to it in the file. Thus, if a thimble or LP is broken, exchanged or perhaps sold, that particular position on the file must not be lost but kept open until the number is used again.

The program makes it possible to call up a particular record by number and to alter, cancel, or return it to the file unaltered. Cancelling a record means replacing it by a set phrase, in this case, "ZZZZ", which is unlikely to be used otherwise in section 1.

If you wish to amend or cancel a particular record but do not known its number, then calling up the records which fit the description (or artist or make that is known) will result in a list of all possible items, including their numbers. The chosen number can then be called.

The following PROCs deal with the alteration:

- **PROCamending** (1790-1890) which offers the possibilities of changing, cancelling or leaving the record unaltered. If the string is to be altered, it is redone completely and checked on completion, using:
- **PROCcheck1** (2040-2100) which is similar to the check used plus new items. The difference is in the numbering of the item which must be kept the same as the original brought from the file, else it will be put back in the *wrong* place.

FILING PROCEDURES

The example is based on the filing system for BBC Basic II which includes the command OPENUP, in addition to the commands in Basic I. Only two of the PROCs will need changing for the earlier models (they will be pointed out when the situation arises). The name of the file in this case is "thimble" and this should be replaced wherever it appears by the appropriate name.

- **PROCnewfile** (900-990) opens up a new file when the first collection of data is ready to be filed. Separating the first filing from later ones avoids the misuse of the command OPENOUT which has the unfortunate characteristic of destroying any file of that name already in existence. This command should be used with care!

Listing 2

```
  10  REM..LISTING 2
1600 DEFPROCaddfile
1604 *RENAME thimble TEMP
1608 Y=OPENOUT("thimble")
1612 X=OPENIN("TEMP")
1616 PY=PTR#Y
1620 PX=PTR#X
1624 REPEAT
1628 PTR#X=PX
1632 INPUT#X,D$
1636 PTR#Y=PY
1640 PRINT#Y,D$
1644 PY=PY+80
1648 PX=PX+80
1652 UNTIL EOF#X
1656 CLOSE#X
1660 FOR I=1 TO ADD
1664 PTR#Y=PY
1668 PRINT#Y,D$(I)
1672 PY=PY+80
1676 NEXT I
1680 CLOSE#Y
1684 *DELETE TEMP
1700  ENDPROC
1970 DEFPROCaddfilesingle
1972 *RENAME thimble TEMP
1974 Y=OPENOUT("thimble")
1976 X=OPENIN("TEMP")
1978 PY=PTR#Y
1980 PX=PTR#X
1982 REPEAT
1984 PTR#X=PX
1986 INPUT#X,D$
1988 PTR#Y=PY
1990 PRINT#Y,D$
1992 PY=PY+80
1994 PX=PX+80
1996 UNTIL PX=(N-1)*80
1998 PTR#Y=PY
2000 PRINT#Y,D$(ADD)
2002 REPEAT
2004 PY=PY+80
2006 PX=PX+80
2008 PTR#X=PX
2010 INPUT#X,D$
2012 PTR#Y=PY
2014 PRINT#Y,D$
2016 UNTIL EOF#X
2018 CLOSE#X
2020 CLOSE#Y
2022 *DELETE TEMP
```

Listing 3

```
   10 REM..THIMBRR
   20 DIMD$(300)
   30 REM...1...MAKER..(16)
   40 REM...2...SET..(20)
   50 REM...3...ARTIST(18)
   60 REM...4...DESCRIPTION..(22)
   70  CLS:PRINT''"Please make sure that your cass
ette is  correctly positioned to load the file.  P
ress the SPACE BAR when ready"
   80 PROCWAIT:PROCfromfile:N=I-1
   85  PROCchoice
   90 ON C GOTO 100,190,430,490,500
  100  ADD=N
  110 CLS:PRINT''"Type = when you have finished ad
ding    items."
  120 PRINT'"Press the return key after each entry
."
  130 CLS:ADD=ADD+1:D$(ADD)="":PROCfirst:IF N$="="
 THEN 160 ELSE 140
  140 PROCsecond:PROCthird:PROCfourth:PROCcheck
  150  GOTO 110
  160  GOTO 85
  190 CLS:PRINT''"Type 1.  to look for a maker"
  200 PRINT'"      2.  to look for a set"
  210 PRINT'"      3.  to look for an artist"
  220 PRINT'"      4.  to look for a description"
  230  INPUTCS:IF (CS-1)*(CS-2)*(CS-3)*(CS-4)<>0 T
HEN 230
  240  ON CS GOTO 250,300,350,390
  250 PRINT''"The maker is?":INPUT C1$
  260  IF LEN(C1$)>16 THEN 270 ELSE 280
  270 C1$=LEFT$(C1$,16)
  280 VDU2:PRINT';C1$:PROClooking
  290 PROCWAIT:VDU3:GOTO 80
  300 PRINT''"The name of the set is?":INPUT C2$
  310 IF LEN(C2$)>22 THEN 320 ELSE 330
  320 C2$=LEFT$(C2$,22):CLS:PRINT'C2$
  330 VDU2:PRINT'C2$:PROClooking
  340 PROCWAIT:VDU3:GOTO 80
  350  PRINT''"The name os the artist is?":INPUT C
3$
  360  IFLEN(C3$)>18 THEN C3$=LEFT$(C3$,18)
  370 VDU2:PRINT'C3$:PROClooking
  380 PROCWAIT:VDU3:GOTO 80
  390  PRINT''"The desription you wish to find is?
":INPUT C4$
  400  IFLEN(C4$)>22 THEN C4$=LEFT$(C4$,22)
  410 VDU2:PRINT'C4$:PROClooking
  420 PROCWAIT:VDU3:GOTO 80
  430 CLS:PRINT'"What is the number of the thimble
?"
  440 INPUT number  :ADD=number
  450 PROCamending:IF R=3 THEN 480
  460 IF C$="Y" OR C$="y"THEN 480 ELSE 470
  470 PROCamending:GOTO 460
  480 GOTO 80
  490 VDU2:PROClist:VDU3:PROCWAIT:GOTO 80
  500  CLS:PRINT''"Please make sure that your cass
ette is  correctcly positioned to save the file. P
ress the SPACE BAR when ready"
  510 PROCWAIT:PROCnewfile:END
  515 DEFPROCfirst
  520 PRINT''"The name of the maker or manufacturer
?"
  530  INPUT N$:IF N$="=" THEN 580
  540 D$(ADD)=D$(ADD)+N$
  550 IF LEN(D$(ADD))>15 THEN 570 ELSE 560
  560 D$(ADD)=D$(ADD)+" ":GOTO 550
  570 D$(ADD)=LEFT$(D$(ADD),16)
  580 ENDPROC
  590 DEFPROCsecond
  600 PRINT'"The name of the set"
  610 INPUT N$:D$(ADD)=D$(ADD)+N$
  620 IF LEN(D$(ADD))>35 THEN 640 ELSE 630
  630 D$(ADD)=D$(ADD)+" ":GOTO 620
  640 D$(ADD)=LEFT$(D$(ADD),36)
  650 ENDPROC
  660 DEFPROCfourth
  670 PRINT'"The description of the thimble?":INPU
T N$:D$(ADD)=D$(ADD)+N$
  680 IF LEN(D$(ADD))>75 THEN 700 ELSE 690
  690 D$(ADD)=D$(ADD)+" ":GOTO 680
  700 D$(ADD)=LEFT$(D$(ADD),76)
  710 ENDPROC   '
  720 DEFPROCcheck
  730 PRINT; N+ADD,D$(ADD)
  740 PRINT'"Is this correct?(Y/N)"
  750 INPUT C$:IF C$="Y" OR C$="y" THEN 780
  760 IF C$="N" OR C$="n" THEN 770 ELSE 750
  770 D$(ADD)="":ADD=ADD-1
  780 ENDPROC
  790 DEFPROCfromfile
  800 X=OPENIN("thimble")
  810 I=0
  820 REPEAT
  840 I=I+1
  850  INPUT#X, D$(I)
  870 UNTIL EOF#X
  880 CLOSE#X
  890 ENDPROC
  900 DEFPROCnewfile
  910 X=OPENOUT("thimble")
  930 FOR I=1 TO ADD
  950 PRINT#X,D$(I)
  970 NEXT
  980 CLOSE#X
  990 ENDPROC
 1000 DEFPROCchoice
 1010 CLS:PRINT''"Type 1  to add items"
 1020 PRINT'"      2  to extract items"
 1030 PRINT'"      3  to correct or remove items"
 1040 PRINT'"      4  for a complete list"
 1050 PRINT'"      5  to end"
 1060 INPUT C:IF(C-1)*(C-2)*(C-3)*(C-4)*(C-5)<>0 T
HEN 1060
 1070 ENDPROC
 1080 DEFPROClooking
 1100 I=0
 1110 REPEAT
```

```
1130 I=I+1
1150 IF CS>1 THEN 1170 ELSE PROClookfirst
1160 GOTO 1220
1170 IF CS>2 THEN 1190 ELSE PROClooksecond
1180 GOTO 1220
1190 IF CS>3 THEN 1210 ELSE PROClookthird
1200 GOTO 1220
1210 PROClookfourth
1220 UNTIL I=ADD
1250 ENDPROC
1260 DEFPROClookfirst
1270 L=LEN(C1$):IF L=16 THEN1310
1280 J=1
1290 IF MID$(D$(I),J,L)<>C1$ THEN 1300 ELSE 1320

1300 J=J+1:IF J<17-L THEN 1290 ELSE 1330
1310  IF LEFT$(D$(I),16)<>C1$ THEN 1330 ELSE 1320

1320 PRINT;I;TAB(6);MID$(D$(I),17,20);TAB(30);MID
$(D$(I),37,18);TAB(50);RIGHT$(D$(I),22)
1330 ENDPROC
1340 DEFPROClooksecond
1350 L=LEN(C2$):IF L=20 THEN1310
1360 J=1
1370 IF MID$(D$(I),J+16,L)<>C2$ THEN 1380 ELSE 14
00
1380 J=J+1:IF J<21-L THEN 1370 ELSE 1410
1390 IF MID$(D$(I),17,20)<>C2$ THEN 1410 ELSE 140
0
1400 PRINT;I;TAB(6);LEFT$(D$(I),16);TAB(30);MID$(
D$(I),37,18);TAB(50);RIGHT$(D$(I),22)
1410 ENDPROC
1420 DEFPROClookthird
1430 L=LEN(C3$):IF L=18 THEN 1470
1440 J=1
1450 IF MID$(D$(I),36+J,l)<>C3$ THEN 1460 ELSE 14
80
1460 J=J+1:IF J<19-L THEN 1460 ELSE 1490
1470 IF MID$(D$(I),36,L)<> C3$ THEN 1490 ELSE 148
0
1480 PRINT;I;TAB(6)LEFT$(D$(I),16);TAB(30);MID$(D
$(I),17,20);TAB(52)RIGHT$(D$(I),22)
1490 ENDPROC
1500 DEFPROCWAIT
1510 X=GET:IFX<>32 THEN 1510
1520 ENDPROC
1530 DEFPROCthird
1540 PRINT'"The name of the artist"
1550 INPUT N$:D$(ADD)=D$(ADD)+N$
1560 IF LEN(D$(ADD))>53 THEN 1580 ELSE 1570
1570 D$(ADD)=D$(ADD)+" ":GOTO 1560
1580 D$(ADD)=LEFT$(D$(ADD),54)
1590 ENDPROC
1710 DEFPROClookfourth
1720 L=LEN(C4$):IF L=22 THEN 1760
1730  J=1
1740 IF MID$(D$(I),56+J,L)<>C4$ THEN 1750 ELSE 17
70
1750 J=J+1:IFJ<23-L THEN 1740 ELSE 1780
1760 IF RIGHT$(D$(I),22)<>C4$ THEN 1780 ELSE 1770

1770 PRINT;I;TAB(6)LEFT$(D$(I),16);TAB(30);MID$(D
$(I),17,20);TAB(52);RIGHT$(D$(I),22)
1780 ENDPROC
1790 DEFPROCamending
1800 PRINTD$
1810 PRINT''"Do you wish to:"
1820 PRINT'"1.  change this record"
1830 PRINT'"2.  cancel this record"
1840 PRINT'"3.  leave it unchanged"
1850 INPUT R:IF (R-1)*(R-2)*(R-3)<>0 THEN 1850
1860  ON R GOTO 1880,1870,1890
1870 D$="ZZZZ"+STRING$(72," "):ENDPROC
1880 D$(ADD)="":PROCfirst:PROCsecond:PROCthird:PR
OCfourth:PROCcheck1
1890 ENDPROC
2040 DEFPROCcheck1
2050 PRINT;number,D$(ADD)
2060 PRINT'"Is this correct?(Y/N)"
2070 INPUT C$:IF C$="Y" OR C$="y" THEN 2100
2080 IF C$="N" OR C$="n" THEN 2090 ELSE 2070
2090 D$(ADD)=""
2100 ENDPROC
2110 DEFPROClist
2130 I=0
2140 REPEAT
2160 I=I+1
2180 PRINT;I;D$(I)
2200 UNTIL I=ADD
2220 ENDPROC
```

- **PROCfromfile** (790-890) is only used here as a counting device so that the numbering of new items will be correct. It will be needed in a slightly amended form for the cassette file but it could be omitted for the disc file as long as some other way of keeping note of the number of records on the file is adopted. One way is to use the first record on the file to hold the number which tells you how many records there are. This means that the data would then begin with the second record.
- **PROCaddfile** (1600-1700) It is here that knowing the number N of records already on the file becomes important. The pointer (PTR#X) is moved to the beginning of the (N+1)th record by moving it N*(length of 1 record) bytes. This can only be done using OPENUP and the alternative PROC for other versions will be found later in the article. Once the pointer is in the correct place, the ADD new items will be added on to the file.
- **PROClooking** (1080-1250) is the means of looking through the items of the file for items which satisfy given conditions, such as a particular artist or description.
- **PROCchanging** (1900-1960) finds the record asked for by number and offers it for amendment.
- **PROCaddfilesingle** (1970-2030) If a single record has been amended after PROCchanging, this is the means of putting the revised version back into the correct position. It also uses the command OPENUP and will need alteration for other systems.

THE MAIN PROGRAM

This is situated between lines 10-500. It uses two PROCs which have not already been described.
- **PROCchoice** (1000-1070) contains the menu available.
- **PROCWAIT** (1500-1520) waits for the space bar to be pressed before moving on.

Other variables are:

C	The variable carried forward from the menu.
CS	The variable indicating which section is to be tested when extracting information
C1$, C2$, C3$, C4$	The variables representing the strings to be searched for in sections 1, 2, 3 or 4 respectively.

Line 100 contains two statements. When the file is being used for the first time it should read:

100 N=0:ADD=0

After the first use, the line should be:

100 PROCfromfile:N=I−1:ADD=0

This change could be avoided by putting in an extra question, eg Is this a new file? (Y/N) but the question would then have to be answered every time the program was used. It is a matter of personal choice as to which method is preferred.

USING A PRINTER

When lists of items are extracted from the file, it is useful to have the list printed out (if a printer is available). The VDU2 statement which enables the printer will be found on lines 280, 330, 370, 410 and 490. The printer is then switched off on the following lines when the lists are complete and is only on when actually needed.

The complete program will be found in **Listing 1.** Option 4 on the menu (a complete list) will be found to be useful when the program is first run. It will give an immediate check as to whether the items are being filed as you expect. **Listing 2** gives the alterations necessary when the command OPENUP is not available. **Listing 3** gives suggested alterations when a cassette file is used. To use the file for the first time, ie when no previous file exists, an extra line can be added but must be deleted before the second and subsequent runs.

65 N=0: GOTO 100

Subscriptions

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

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

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

COMPUTING TODAY Subscriptions,

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

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

Looking for a magazine with a professional approach with material written by micro users for micro users? Why not do yourself a favour and make 1984 the year you subscribe to Computing Today and we'll give you a truly personal approach to microcomputing.

SUBSCRIPTION ORDER FORM

Cut out and SEND TO :
COMPUTING TODAY Subscriptions
INFONET LTD,
TIMES HOUSE,
179 THE MARLOWES,
HEMEL HEMPSTEAD,
HERTS HP1 1BB.

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

SUBSCRIPTION RATES (tick ☐ as appropriate)

- £13.90 for 12 issues UK ☐
- £17.55 for 12 issues Overseas Surface ☐
- £37.20 for 12 issues Overseas Airmail ☐

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

BARCLAYCARD VISA

Please use BLOCK CAPITALS and include postcodes.

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

ADDRESS ....

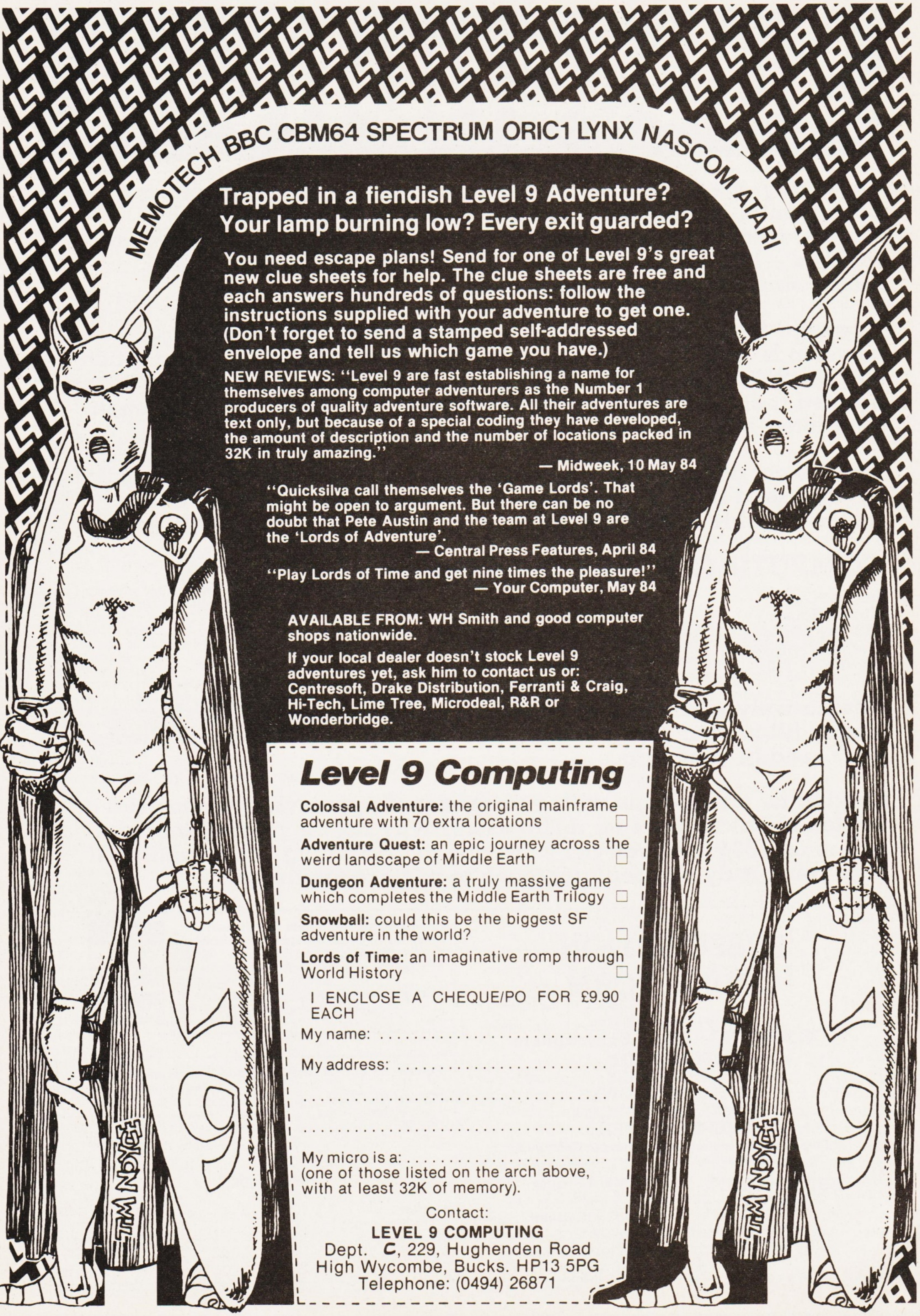
....

.... POSTCODE ....

Signature ....

Date ....

CT Aug '84


MEMOTECH BBC CBM64 SPECTRUM ORIC1 LYNX NASCOM ATARI
Trapped in a fiendish Level 9 Adventure? Your lamp burning low? Every exit guarded?
You need escape plans! Send for one of Level 9's great new clue sheets for help. The clue sheets are free and each answers hundreds of questions: follow the instructions supplied with your adventure to get one. (Don't forget to send a stamped self-addressed envelope and tell us which game you have.)
NEW REVIEWS: "Level 9 are fast establishing a name for themselves among computer adventurers as the Number 1 producers of quality adventure software. All their adventures are text only, but because of a special coding they have developed, the amount of description and the number of locations packed in 32K in truly amazing."
— Midweek, 10 May 84
"Quicksilva call themselves the 'Game Lords'. That might be open to argument. But there can be no doubt that Pete Austin and the team at Level 9 are the 'Lords of Adventure'.
— Central Press Features, April 84
"Play Lords of Time and get nine times the pleasure!"
— Your Computer, May 84
AVAILABLE FROM: WH Smith and good computer shops nationwide.
If your local dealer doesn't stock Level 9 adventures yet, ask him to contact us or: Centresoft, Drake Distribution, Ferranti & Craig, Hi-Tech, Lime Tree, Microdeal, R&R or Wonderbridge.
Level 9 Computing
Colossal Adventure: the original mainframe adventure with 70 extra locations ☐
Adventure Quest: an epic journey across the weird landscape of Middle Earth ☐
Dungeon Adventure: a truly massive game which completes the Middle Earth Trilogy ☐
Snowball: could this be the biggest SF adventure in the world? ☐
Lords of Time: an imaginative romp through World History ☐
I ENCLOSE A CHEQUE/PO FOR £9.90 EACH
My name: ....
My address: ....
My micro is a: ....
(one of those listed on the arch above, with at least 32K of memory).
Contact:
LEVEL 9 COMPUTING
Dept. C, 229, Hughenden Road
High Wycombe, Bucks. HP13 5PG
Telephone: (0494) 26871

EXTENDING THE 64'S BASIC PART4

Tony Cross

How long is a string? Or come to that, any other type of variable? In this month's article we look at string operations (with two new keywords) and variable storage (another new keyword).

So far in this series I have only looked at the use of parameters in keywords. It is quite possible, of course, to use strings and string expressions as keyboard parameters, it just requires some different techniques. This month, therefore, I am going to concentrate on evaluation and manipulating strings and string expressions and then taking a look at BASIC variables, both string and numeric.

STORING STRINGS

Strings are stored and manipulated in a way that is quite different from the way numbers are handled. This difference arises because of the physical size of the strings themselves. You may remember from last month, that floating point numbers are always five bytes long and integers are one or two bytes long at the most. Strings,on the other hand, can be anything from 0 to 255 bytes long.

In addition, when we need to manipulate floating point and integer numbers, it is the numbers themselves that are moved around. Now a 255 byte long strip is a very large beast indeed — if it became necessary to move several strings of this length around then the Interpreter would slow down considerably.

The Commodore 64s BASIC (like many other BASICs) overcomes these problems by using a two part storage method for its strings:

- The string itself
- A string descriptor (or header)

The first part (the string itself) is stored at some convenient location in memory. (More on this later.) The second part (the string descriptor) is a three byte 'block' that completely describes the string. It consists of two pieces of information:

- The string length (the first byte of the descriptor)
- The string address (the last two bytes of the descriptor)

Because the string descriptor contains all the information we need to know about the string, we can do all the manipulation on the descriptor, instead of on the (much longer) string. This makes working with strings much faster (and simpler).

Now let's have a closer look at how the string itself is stored. There are actually three main types of string:

1 String variables I'll be looking at variables in detail later, for now you can ignore them completely.

2 Constant strings These are strings which are contained within the program text itself. For example, in the following program:

```
10 PRINT "FRED"
```

The string "FRED" is a constant, ie it is part of the program text and cannot be changed at RUN time. The descriptor for this type of string actually points to the text of the program itself.

3 Temporary strings These are strings which are created as the program is running. For example, in the following program:

```
10 LET A$="FRED"
20 LET B$="BLOGS"
30 PRINT A$+B$
```

Line 30 has to create the temporary string "FRED BLOGGS" so that it can be printed. The string is only temporary because when the PRINT routine has finished, it is no longer needed and can be 'forgotten about'. Strings like this have to be stored somewhere whilst they are being manipulated, and an area of memory called the 'string storage space' is used. This area (which I will be describing in detail later) is accessed via a pointer called FRETOP. FRETOP points to the bottom of the string storage area, ie where the next temporary string will be stored. (String storage space 'grows' downwards from high memory — usually address $9FFF).

When a temporary string is created, it is copied into string space starting at the current FRETOP location. FRETOP is then moved down just below it so that another string can be stored. When the routine which created the temporary string no longer needs it, FRETOP is simply moved back above it again making the space available for another string. (This is called de-allocating a temporary string.)

In this way the most efficient use is made of the string storage space — a temporary strings are only stored there for as long as they are needed.

THE TEMPORARY STRING STACK

There is another feature of temporary strings that you need to know about, that is the 'temporary string stack'. To see why a string stack is needed (and how it is used) have a look at the following program:

```
10 LET A$="FRED"
20 LET B$="JOHN"
30 LET C$="SID"
40 PRINT MID$(A$+B$+LEFT$(C$,2),3,7)
```

There's a lot of string processing to be done in line 40 — far too much to be tackled all in one go. Complex expressions like this have to be evaluated in simple stages and 'built up' a bit at a time.

First the temporary string A$+B$ is evaluated and stored in string space. The descriptor for this string must be saved for use later and so it is 'pushed' onto the temporary string stack.

The second temporary string, LEFT$(C$,2), is then evaluated and stored in string space (just below the A$+B$ string as it happens). The descriptor for this string is also 'pushed' onto the temporary string stack. (There are now two descriptors on the stack.)

These two descriptors are then 'popped' off the stack and used in evaluating the third temporary string, MID$(A$+B$+LEFT$(C$,2),3,7). The first two temporary strings, which are no longer needed, are then de-allocated (FRETOP is moved back above them) and the third temporary string is copied into string space (over the top of the other two). The descriptor for this string is then 'pushed' onto the string stack. (It is now the only string on the stack). The expression has now been completely evaluated (because there are no more 'operators' to be dealt with), so this string is returned as the evaluated result: the PRINT routine can now print this string, de-allocate it and then return control to the Interpreter.

You can see that a temporary string stack is essential when evaluating this type of expression. It is used to store the descriptors to the various temporary strings which are produced. On the Commodore 64 this stack is three 'slots' long (nine bytes) and it is located at addresses $0019 to $0021 ($0019 is the 'top' of the stack). If a string expression generates more than three temporary strings (which must all be kept at the same time) then a FORMULA TOO COMPLEX error will be given.

DEALING WITH STRING EXPRESSIONS

Evaluating string expressions is no more difficult than evaluating numeric expressions — you simply use a different set of ROM routines. To completely extract a string parameter, however, you need to call two separate ROM routines. The first of these, called EXPR, evaluates the string parameter and returns a descriptor to the evaluated string in FPA1. The second routine, called GETSTG, recovers the descriptor from FPA1 and checks to see if it is a temporary string. If it is, then GETSTG de-allocates it (by moving FRETOP above it).

The EXPR routine (EXPRession evaluations) at address $AD9E, is actually the main expression evaluation subroutine. (Strings don't have a separate 'entry point' like INCBYT, INCINT and NUMEXP.)

EXPR evaluates the expression (numeric or string) pointed to by TXTPTR (TXTPTR is NOT incremented first). If the expression found is a string then EXPR returns a descriptor to the string in FPA1. (Length in $61, and address in $62/$63, with the low byte in $62). If the expression found is numeric, EXPR simply returns a floating point number in FPA1.

EXPR leaves TXTPTR pointing to the delimiter character at the end of the expression and modifies all the registers. Although it looks as though EXPR has completely extracted the string parameter, there are two important points which need clarifying. Firstly, we must check to ensure that a string expression was found (rather than a numeric one). And secondly, we need to find out whether the string is a temporary one, in which case it will have to be de-allocated when we have finished with it.

GETSTG (GET a STrinG), at address $B6A3, performs both of these functions for us. Firstly, it checks to ensure that the expression just evaluated produced a string result (a TYPE MISMATCH error is given if not). Next, the string length is loaded into the X/Y registers (high byte in Y, low byte in X). This address is also loaded into locations $22/$23 (high byte in $23, low byte in $22). More importantly, GETSTG checks the string address to see if it is a temporary string. If it is then GETSTG de-allocates it by moving FRETOP above it.

Now it may seem strange to you that we should de-allocate the string as soon as it has been evaluated. However, the string still exists and the descriptor still points to it — all we have done is to 'remove' it from string space. Of course, if another temporary string were to be created it would overwrite the current one. The only way this can happen though is if you call EXPR before you have finished dealing with the current string!

OTHER STRING OPERATIONS

If the string parameter came from a statement keyboard then the string descriptor in A and X/Y is all you need to carry out whatever action you had in mind. If the keyboard is a function keyword however, a value will have to be returned. Some string functions, like LEN(string$), return numeric results. Others, like LEFT$(strings$, number), return string results. Those functions that return numeric results usually return a floating point number in FPA1, and those functions that return string results return a string descriptor in FPA1. (Length in $61 and address in $62/$63, with the low byte in $62).

```
 20 033C            !###################################
 30 033C            !#                                 #
 40 033C            !#        INSTRING  KEYWORD        #
 50 033C            !#                                 #
 60 033C            !#    VERSION 1.0 -- 13/01/84      #
 70 033C            !#                                 #
 80 033C            !# COPYRIGHT (C)  A.L.CROSS 1984 #
 90 033C            !#                                 #
100 033C            !###################################
110 033C            !
120 033C            !
130 C4B8            *=$C4B8
140 C4B8            !
150 C4B8            !
160 C4B8            !VARIABLES AND EQUATES
170 C4B8            !
180 C4B8            TSTOPB       =  $AEFA
190 C4B8            TSTCLB       =  $AEF7
200 C4B8            TSTCOM       =  $AEFD
210 C4B8            GETSTG       =  $B6A3
220 C4B8            CVTFPN       =  $B391
230 C4B8            CHKSTK       =  $A3FB
240 C4B8            EXPR         =  $AD9E
250 C4B8            ADDS1        =  $FB
260 C4B8            ADDS2        =  $22
270 C4B8            LEN1         =  $FD
280 C4B8            LEN2         =  $FE
290 C4B8 00         COUNT       BYT $00
300 C4B9            !
310 C4B9            !
320 C4B9 E0FF       INSTR       CPX #$FF            !CHECK FUNCTION FLAG
330 C4BB F003                   BEQ INSTOK
340 C4BD 4C08AF                 JMP $AF08
350 C4C0 A901       INSTOK      LDA #$01            !CHECK STACK SPACE
360 C4C2 20FBA3                 JSR CHKSTK
370 C4C5 20FAAE                 JSR TSTOPB          !CHECK BRACKET
380 C4C8 209EAD                 JSR EXPR            !GET STRING1$
390 C4CB 20A3B6                 JSR GETSTG
400 C4CE 85FD                   STA LEN1            !SAVE LENGTH BYTE
410 C4D0 A533                   LDA $33
420 C4D2 48                     PHA                 !SAVE CURRENT END OF
430 C4D3 A534                   LDA $34
440 C4D5 48                     PHA                 !STRING SPACE POINTER
450 C4D6 86FB                   STX ADDS1           !SAVE LOW ADDS BYTE
460 C4D8 8633                   STX $33             !UPDATE END OF
470 C4DA 84FC                   STY ADDS1+1         !SAVE HIGH ADDS BYTE
480 C4DC 8434                   STY $34             !STRING SPACE POINTER
490 C4DE 20FDAE                 JSR TSTCOM          !CHECK COMMA
500 C4E1 209EAD                 JSR EXPR            !GET STRING2$
510 C4E4 20A3B6                 JSR GETSTG
520 C4E7 85FE                   STA LEN2            !SAVE LENGTH BYTE
530 C4E9 68                     PLA                 !RESTORE OLD END OF
540 C4EA 8534                   STA $34
550 C4EC 68                     PLA                 !STRING SPACE POINTER
560 C4ED 8533                   STA $33
570 C4EF 20F7AE                 JSR TSTCLB          !CHECK BRACKET
580 C4F2 A5FD                   LDA LEN1            !GET LEN(STRING1$)
590 C4F4 38                     SEC
600 C4F5 E5FE                   SBC LEN2            !SUB LEN(STRING2$)
610 C4F7 902E                   BCC NOTIN           !IF STRING2$>STRING1$
620 C4F9 8DB8C4                 STA COUNT
630 C4FC EEB8C4                 INC COUNT           !INIT LENGTH COUNT
640 C4FF A201                   LDX #$01            !INIT POSITION COUNT
650 C501 A000       OUTRLP      LDY #$0             !INIT POSITION INDEX
660 C503 B1FB       MAINLP      LDA (ADDS1),Y       !STRING1$ CHAR
670 C505 D122                   CMP (ADDS2),Y       !EQUAL TO STRING2$
680 C507 D008                   BNE NOTEQL
690 C509 C8                     INY                 !NEXT CHARACTER
700 C50A C4FE                   CPY LEN2            !END OF STRING2$?
710 C50C B012                   BCS STGFND
720 C50E 38                     SEC
730 C50F B0F2                   BCS MAINLP          !REPEAT
740 C511 E8         NOTEQL      INX                 !INC POSITION COUNT
750 C512 CEB8C4                 DEC COUNT           !DEC CHARACTER COUNT
760 C515 F010                   BEQ NOTIN           !IF AT END OF STRING1$
770 C517 E6FB                   INC ADDS1           !INC STRING1$ ADDS
780 C519 D0E6                   BNE OUTRLP          !POINTER
790 C51B E6FC                   INC ADDS1+1
800 C51D 38                     SEC
810 C51E B0E1                   BCS OUTRLP
820 C520 8A         STGFND      TXA                 !GET CHARACTER POS
830 C521 A8                     TAY                 !INTO A/Y PAIR
840 C522 A900       RTNEND      LDA #$0
850 C524 4C91B3                 JMP CVTFPN          !CONVERT TO FPN
860 C527 A000       NOTIN       LDY #$0             !SET A/Y TO ZERO
870 C529 38                     SEC
880 C52A B0F6                   BCS RTNEND
```

Listing 1. The INSTR keyword.

Returning a numeric result is fairly straightforward and I don't expect you will have any problems. However, there are a couple of ROM routines that might be of use in this area.

ASCII (return an ASCII value), at address $78B, first calls GETSTG to 'sort out' the evaluated string. It then loads FPA1 with the ASCII value of the first character in the string (in floating point format). If the string was null (a length of zero) then an ILLEGAL QUANTITY error will be given. All the registers are modified by a call to this routine.

VALUE (returns the string VALUE), at address $B7AD, also calls GETSTG first, to 'sort out' the evaluated string. It then loads FPA1 with the value of the string (in floating point format), up to the first non-numeric character. All the registers are modified by a call to this routine.

Returning a string result is a little bit more difficult however, because the string itself has to be stored somewhere in memory. The obvious place to use is the string storage area used for temporary strings — there are a number of ROM routines to help you do this.

STRSPC (allocate STRing SPaCe) at address $B47D, allocates space in the string storage area. The number of bytes to be allocated is specified in the A register on entry. STRSPC returns with the address of the first byte in the X/Y registers (high byte in Y, low byte in X) and the length in the A register. These values are also in FPA1 (length in $61 and address in $62/$63, with the low byte in $62).

MOVSTG (MOVe a STrinG) at address $B688, moves a string into the last allocated area. The address of the string to be moved is specified in the X/Y registers (high byte in Y, low byte in X) and the string length is specified in the A register. The A and Y registers are modified by MOVSTG but the X register is not.

STGPTR (set up the STrinG PoinTeRs) at address $B4CA, sets up the temporary string stack and assembles the string descriptor in FPA1 for the string described in locations $61/$62/$63. (Length in $61 and address in $62/$63, with the low byte in $62). All the registers are modified by STGPTR.

CVTSTG (ConVerT to STrinG), at address $BDDD, converts a floating point number in FPA1 to string. On return, the A/Y registers contain the address of the string (high byte in Y, low byte in A). The string will be terminated by a null byte and all the registers are modified.

NEW KEYBOARD ROUTINES

There are two new keyboard routines in this section, INSTR and MULT$. Both are function keywords but INSTR returns a numeric result, whilst MULT$ returns a string result.

INSTR(string1$,string2$)

INSTR searches 'string1$' to see if it contains 'string2$'. If 'string2$' is contained within 'string1$' then the position of the first character of 'string2$' is returned. If 'string2$' is not contained within 'string1$', a zero result is returned. For example:

PRINT INSTR("FRED","RED") will print the value 2.
PRINT INSTR("FRED","ED") will print the value 3.
PRINT INSTR("FRED","FED") will print the value 0.
PRINT INSTR("FRED","FREDA") will also print the value 0.

MULT$(length,string$)

MULT$ returns a string of length 'length' characters which contains only the first character of 'strings$'. For example:

```
0330          !####################################
033C          !#                                  #
033C          !#        MULT$ KEYWORD    •        #
033C          !#                                  #
033C          !#     VERSION 1.0 -- 16/01/84      #
033C          !#                                  #
033C          !# COPYRIGHT (C)  A.L.CROSS 1984    #
033C          !#                                  #
033C          !####################################
033C          !
033C          !
C52C          *=$C52C
C52C          !
C52C          !
C52C          !VARIABLES AND EQUATES
C52C          !
C52C          TSTOPB       =  $AEFA
C52C          TSTCLB       =  $AEF7
C52C          TSTCOM       =  $AEFD
C52C          GETBYT       =  $B79E
C52C          GETSTG       =  $B6A3
C52C          STRSPC       =  $B47D
C52C          STGPTR       =  $B4CA
C52C          CHKSTK       =  $A3FB
C52C          EXPR         =  $AD9E
C52C          STGLEN       =  $FB
C52C          !
C52C          !
C52C E0FF     MULT         CPX #$FF          !TEST FUNCTION FLAG
C52E F003                  BEQ MULTOK
C530 4C08AF                JMP $AF08         !SYNTAX ERROR
C533 A901     MULTOK       LDA #$01          !CHECK STACK SPACE
C535 20FBA3                JSR CHKSTK
C538 20FAAE                JSR TSTOPB        !CHECK BRACKET
C53B 209EB7                JSR GETBYT        !GET LENGTH
C53E 8A                    TXA
C53F 48                    PHA               !SAVE LENGTH
C540 20FDAE                JSR TSTCOM        !CHECK COMMA
C543 209EAD                JSR EXPR
C546 20A3B6                JSR GETSTG        !GET CHARACTER
C549 85FB                  STA STGLEN        !SAVE CHAR LENGTH
C54B 20F7AE                JSR TSTCLB        !CHECK BRACKET
C54E 68                    PLA               !RESTORE LENGTH
C54F 207DB4                JSR STRSPC        !ALLOCATE STRING SPACE
C552 AA                    TAX               !STRING LENGTH
C553 A000                  LDY #$0           !INITIALISE INDEX
C555 A5FB                  LDA STGLEN        !TEST CHAR LENGTH
C557 F002                  BEQ COPYLP
C559 B122     CHAROK       LDA ($22),Y       !GET CHARACTER
C55B 9162     COPYLP       STA ($62),Y       !COPY STRING
C55D C8                    INY
C55E CA                    DEX
C55F D0FA                  BNE COPYLP
C561 4CCAB4                JMP STGPTR        !SET STRING POINTERS
```

Listing 2. The MULT$ keyword.

PRINT MULT$(3,"FRED") will Print 'FFF'.
PRINT MULT$(5,"$#") will Print '$$$$$'.
PRINT MULT$(4,"*") will Print '****'.
PRINT MULT$(0,"DICK") will print the null string.

THE INSTR KEYWORD

The full listing for INSTR is given in Listing 1. Since you've seen most of the 'standard' techniques before I'll just stick to describing the new ones.

The 'string1$' parameter is extracted first, by calling EXPR and GETSTG. The length and address bytes (in A and X/Y) are saved in the variables LEN1 and ADDS1.

As I mentioned earlier, if 'string1$' was a temporary string, GETSTG will have de-allocated it (by moving FRETOP back above it). If we were to now call EXPR again to extract 'string2$', it would overwrite 'string1$' in the string storage area.

It's fairly easy to get round this problem by a 'crafty' manipulation of the FRETOP pointer. First of all, the current value of FRETOP is saved on the stack (so that we can put it back later). The string address for 'string1$ (in the X/Y registers) is then loaded into FRETOP (this was the FRETOP address before de-allocation of 'string1$'). It is now safe to extract 'string2$' without it overwriting 'string1$' (because we have effectively re-allocated 'string1$').

After extracting 'string2$' (by calling EXPR and GETSTG) the length byte is saved in the variable LEN2. (The string address can be left in locations $22/$23). Because GETSTG has de-allocated 'string2$', FRETOP is currently pointing to the bottom of 'string1$'. By restoring the original value of FRETOP from the values on the stack we can 'manually' de-allocate 'string1$', leaving the string storage area area clear.

Having extracted the two string parameters, checking to see if 'string1$' contains 'string2$' is fairly straight forward. It's simply a case of successively checking each character of 'string1$' against each character of 'string2$'. If all the characters of 'string2$' haved been checked and 'passed', then "string2$' *is* contained within 'string1$'. On the other hand, if all the characters of 'string1$' have been checked and a perfect match has not been found, then 'string2$ is NOT contained within 'string1$'.

During the comparison process, the X register is used as a pointer to the current start character in 'string1$'. If a match is found, then the value in the X register is returned in floating point format in FPA1. (By copying it into the Y register, loading the A register with 0, and calling CVTFPN.) However, if a match cannot be found, or if 'string2$' is longer than 'string1$, then a value of 0 is returned in FPA1 — by loading both the A and Y registers with 0 and calling CVTFPN.

THE MULT$ KEYWORD

The full listing for MULT$ is given in Listing 2 and, as with INSTR, I'll stick to describing the important points.

The two parameters are extracted using routines you have seen before — GETBYT for the 'length' and EXPR/GETSTG for the 'string'. The 'length' parameter is then used to allocate space in the string storage area (where the string result will be 'assembled'). This is done by loading the A register with the 'length' parameter and calling STRSPC.

The result string is then 'assembled' by writing the first character of the 'string' parameter (pointed to by location $22) throughout the allocated area (pointed to by location $62). The only exception to this occurs when the 'length' is zero, in this case a null byte is written throughout the allocated area (to ensure that a null string is returned).

All that remains is to set up a descriptor for the result string in FPA1 (and on the temporary string stack). This is done by simply calling the STRPTR routine (because locations $61/$62/$63 still hold the result string pointers written there by the STRSPC routine).

VARIABLES — THE BASIC TRUTH

You might have noticed that up till now I have deliberately avoided mentioning BASIC's variables (except in passing). This has not been because there is anything difficult about using variables but rather because they are so easy to use! The expression evaluation subroutine (that we have been using to extract all the parameters) deals with variables for us. If it comes across a variable name in an expression then it 'automatically' gets the value of the variable and uses it in the expression. In addition, it checks that the variable type is valid for the current expression and gives a TYPE MISMATCH error if not.

As you can see, using variables in keyword parameters is very simple indeed — so simple in fact, that they are completely transparent! However, there may be occasions when you want to access the value in a variable directly, or perform some 'block process' on a particular group of variables. For these reasons and to complete the picture of how BASIC works, I am going to spend the rest of this month looking at BASIC's variables.

STORING VARIABLES

There are three types of variable used by BASIC;

- Numeric variables
- String variables
- Array variables

Each of these variable types are stored in a separate area in memory. Figure 1 shows the general layout of these areas.

The main variable storage area begins immediately after the end of the program and, since the program length can vary, this location is pointed to by a two byte pointer called VARTAB. VARTAB is located at address \$2D/\$2E and it points to the first byte of the main variable storage area.

The array variable storage begins immediately after the main variable area. The beginning of this area (and the end of the main variable storage area) is pointed to by a two byte pointer called ARYTAB. ARYTAB is located at address \$2F/\$30 and it points to the first byte of the array storage area.

To indicate the end of the array storage area there is a second pointer called STREND, located at address \$31/\$32. STREND points to the end of the array storage area +1.

The string variable storage area is also the temporary string storage space that I mentioned earlier. This area 'begins' at the highest address available to BASIC (usually \$9FFF) and it is pointed to by a two byte pointer called MEMSIZ which is located at address \$37/\$38. String space 'grows' downward and the lowest address of string space is pointed to by a pointer called FRETOP (which we met earlier). FRETOP, located at address \$33/\$34, points to the last byte of the string area −1.

NUMERIC VARIABLES

There are two types of numeric variable, integers (two bytes long) and floating point (five bytes long). Let's begin by looking at floating point variables.

Each floating point variable is stored in a seven byte 'slot' — two bytes for the variable name and five bytes for the floating point value. Figure 2 shows the general layout of a typical floating point variable.

The variable name for floating point variables is stored in straight ASCII format. For example, the variable name AB will be stored as \$41 \$42 and the variable name A will be stored as \$41 \$00.

Integer variables are stored in the same seven byte 'slot' - they just don't use some of the bytes. Each integer variable uses two bytes for the variable name and two bytes for the integer value. The remaining three bytes are not used and are set to zero. Figure 3 shows the layout of a typical integer variable.

The variable name for integer variables is stored in ASCII format with the high bit of both bytes set (1). (This is done to distinguish between floating point and integer variables.) For example, the variable name AB% will be stored as \$C1 \$C2 and the variable name A% will be stored as \$C1 \$80.

STRING VARIABLES

I mentioned earlier that BASIC uses a two part storage method for strings (the string itself and a string descriptor). Not surprisingly, string variables are stored in exactly the same way. The text of string variables is stored in the string storage area and the descriptors for these strings are stored in the main variable storage area. The descriptors are stored using the same seven byte 'slot' used by numeric variables. String variables use two bytes for the variable name and three bytes for the string descriptor (the last two bytes are not used and are set to zero). Figure 4 shows the layout of a typical string variable.

The variable name for string variables is stored in ASCII format with the high bit of the second byte set (1). For example, the variable name AB\$ will be stored as \$41 \$C2, and the variable name A\$ will be stored as \$41 \$80.

STRING GARBAGE COLLECTION

Unfortunately there is a major problem with this method of string variable storage. This is best illustrated by running the following short program:

```
10 DIM A$(200)
20 FOR C=1 TO 189
30 FOR S=1 TO 200
40 A$(S)=A$(S)+"A"
50 NEXT S
60 PRINT "LOOP NUMBER";C
70 NEXT C
```

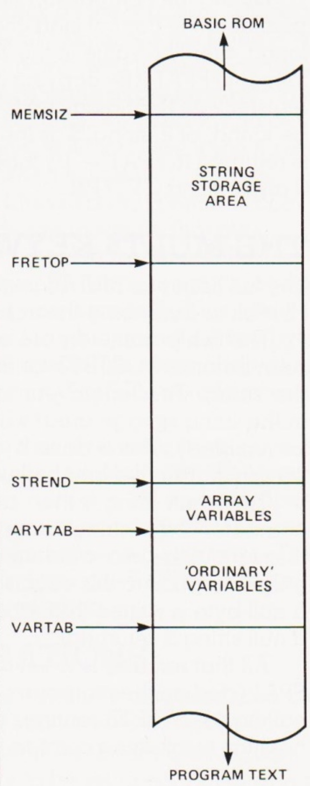


Fig. 1 Variable storage areas.

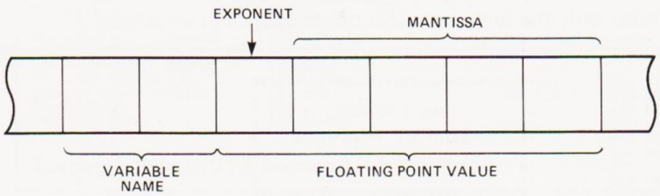


Fig. 2 Floating point variable storage.

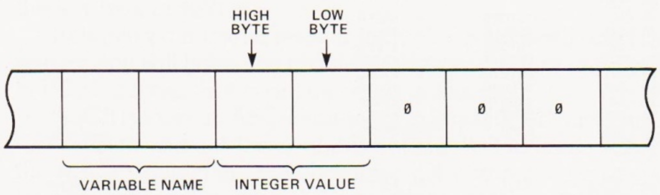


Fig. 3 Integer variable storage.

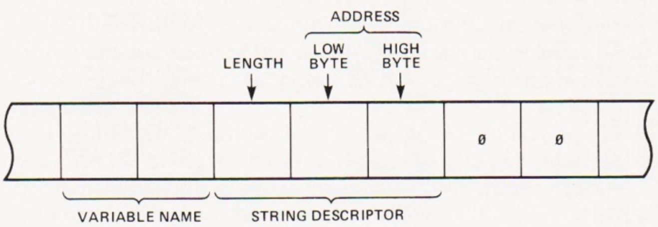
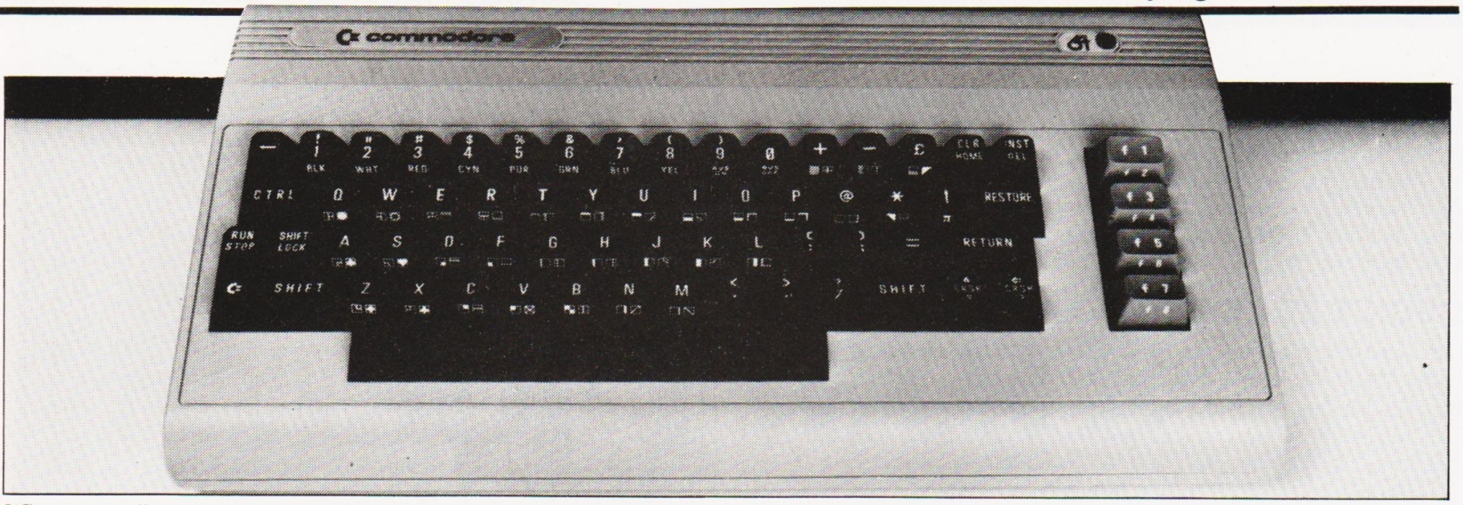


Fig. 4 String variable storage.



What you will see is a program that executes slower and slower each time round the outer loop (the FOR C=1 to 189 loop). The reason for this increasing delay is to do with the way BASIC stores its string variables.

When the contents of an existing string variable are changed, BASIC doesn't overwrite the old string in the string space area because the length may have changed. Instead it adds the new string to the bottom of the string space area and changes the variable descriptor to point to this new string. The old string is now redundant *but it is still stored in the string space area.* If many variables are changed, then the string space area will eventually grow so large that it will fill all the available memory.

When this happens, the Interpreter calls a special routine which runs through the string space area and removes all redundant strings. All the 'good' strings are then re-distributed to pack them tightly together. This routine, which is located at address $B526, is called GARBAG (string GARBAGe collection). GARBAG is a fairly long and slow routine and it can introduce some long delays.

For example, if there is a large amount of free memory or if the program modifies a lot of string variables, then GARBAG will introduce noticable delays into the execution time.

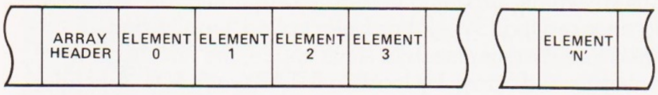


Fig. 5 Array element storage.

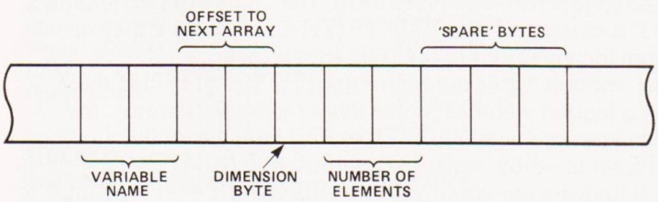


Fig. 6 header for one-dimensional array.

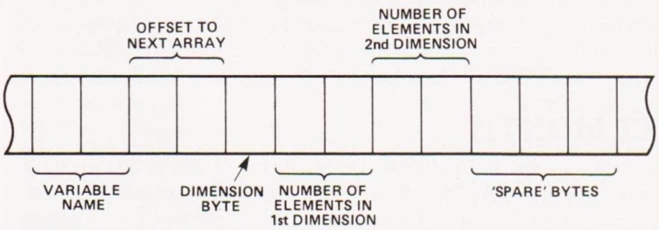


Fig. 7 Header for two-dimensional array.

The short program I showed you earlier is designed to modify lots of variables (each time round the FOR S=1 to 200 loop, 200 new strings are added) and to use most of the available memory (200 strings * 189 bytes per string requires over 36K of memory). This creates the 'worst of both worlds', eventually reaching the situation where garbage collection is being done every time a new string is added. This is why the program takes nearly an hour to run!!

Unfortunately, there isn't a lot you can do about garbage collection, although reducing the amount of memory available for string storage (by lowering MEMSIZ) will make GARBAG run faster (but more often!).

The FRE(0) keyword in BASIC 'forces' garbage collection to take place, so it can be included in programs to make garbage collection occur in places where you can afford the time taken. This can help to prevent it occurring during time critical periods.

ARRAY VARIABLES

The 'ordinary' variables that we have just looked at shared a common seven byte storage 'slot'. Although this makes everything nice and neat, it does waste a lot of memory (three bytes per variable and two bytes per string variable). If array elements were to be stored using this common 'slot' size then the 'wastage' would be even greater (especially with large arrays). For this reason array elements are 'packed' much tighter so that no space is wasted.

Arrays are stored using a two part method:

- An array header (which describes the array)
- The array elements (which contain the data)

The elements of an array are stored sequentially after the header. Figure 5 shows this arrangement, where 'N' corresponds to the dimension of the array.

The array elements use the smallest amount of space required for each variable type — ie integers use two bytes per element, strings use three bytes per element and floating point numbers use five bytes per element.

The length of the header depends on the array being defined, so let's have a look at how the array header is organised. Figure 6 shows a typical one dimension array header.

The 'array variable' name is stored in ASCII format with the high bits set or reset according to the variable type. The 'offset to the next array' is used by the Interpreter when searching for array variables.

The 'dimension byte' specifies the number of dimensions in the array. For example, if a DIM A (23) statement had been executed then the dimension byte would be $01. Similarly, if a DIM A(23,14) statement had been executed then the dimension byte would be $02.

The 'number of elements bytes' specify the number of elements in the array. This value is always one greater than the number specified in the DIM statement as the array elements count from 0. For example, a DIM A(12) statement will create 13 elements (numbered 0-12). The 'number of elements bytes' for this array will be $00 $0D (ie 13).

The area marked 'spare bytes' is used for the additional information needed in a multi-dimensional array. Figure 7 shows a typical two dimension array header.

As you can see, this header has two additional bytes specifying the number of elements in the second dimension. A three dimension array header would have two more bytes specifying the number of elements in the third byte and so on.

DIRECT ACCESS TO VARIABLES

Fortunately, it is not necessary to know exactly how the different types of variable are stored in order to access them directly. The ROM routine which 'reads' the variables can cope with all types 'automatically'. The routine in question is called FNDVAR (FiND a VARiable) and it is located at address $B08B.

FNDVAR reads the variable name being pointed to by TXTPTR and returns a pointer to the contents of the variable found. In the case of numeric variables this is the number itself and in the case of string variables it is the string descriptor.

As I am sure you know, BASIC variable names can be any length you like but only the first two characters are significant. FNDVAR actually reads all characters from the TXTPTR location up to the first 'non-numeric' and 'non-alphabetic' character but only the first two characters are 'saved' as a variable name. If the 'non-numeric' and 'non-alphabetic' character that FNDVAR stopped at is either '$' or '%', then TXTPTR will be left pointing to the first non-space character after the '$' or '%'. However, if this character is not '$' or '%' then TXTPTR will be left pointing at the character. All the registers are modified by FNDVAR.

On entry to FNDVAR the variable INTARY (at address $10) must contain either $00 or $FF. If integers and arrays are allowed then INTARY must contain $00, and if integers and arrays are not allowed it must contain $FF.

If integers are not allowed (INTARY = $FF) and an integer variable is found, then a SYNTAX error will be given. If arrays are not allowed (INTARY = $FF) and an array variable is found, then a pointer to the 'ordinary' variable of the same name will be returned. For example, if the variable B(13) is found and arrays are not allowed, then a pointer to the variable B will be returned. In this case, TXTPTR will be left pointing to the '(' character after the variable name causing a SYNTAX error by the next character checking or end of statement routine.

On leaving FNDVAR, the variable TYPE (at address $0D) will contain either $00 or $FF. If a numeric variable was found then TYPE will contain $00; if a string variable was found then TYPE will contain $FF. If type is numeric ($00) then the variable NUMTYP (at address $0E) will contain either $00 or $80. If an integer variable was found then NUMTYP will contain $80; if a floating point variable was found then NUMTYP will contain $00. If TYPE is string ($FF) then the contents of NUMTYP are undefined.

Also on leaving FNDVAR, the A/Y registers will contain the address of the variable found (high byte in Y, low byte in A). This address is also returned in locations $47/$48 (high byte in $48, low byte in $47). This address points to the exponent byte of a floating point number, the high byte of an integer number or the length byte of a string descriptor.

In other words, on leaving FNDVAR you only need to interrogate TYPE (and perhaps NUMTYP) to find out what type of variable was found, and then read the value of the variable from the location pointed to by the A/Y registers (or location $47/$48).

ANOTHER NEW KEYWORD

This month's third keyword, SWAP, is particularly useful for sorting operations — especially when sorting strings.

SWAP var1,var2

SWAP simply exchanges the contents of two variables of the same type. For example, if A=23 and B=16, then after a SWAP A,B statement has been executed, A will contain 16 and B will contain 23.

The two variables (var1 and var2) can be of valid type — numeric, string or array — but they must be of the same type or a TYPE MISMATCH error will be given.

```
033C          !###################################
033C          !#                                 #
033C          !#          SWAP  KEYWORD          #
033C          !#                                 #
033C          !#    VERSION 1.0 -- 16/01/84      #
033C          !#                                 #
033C          !# COPYRIGHT (C)  A.L.CROSS 1984   #
033C          !#                                 #
033C          !###################################
033C          !
033C          !
C564          *=$C564
C564          !
C564          !
C564          !VARIABLES AND EQUATES
C564          !
C564          FNDVAR       =  $B08B
C564          TSTCOM       =  $AEFD
C564          ERRORS       =  $A437
C564          CHKSTK       =  $A3FB
C564          V2ADDS       =  $47
C564          INTARY       =  $10
C564          NMTYPE       =  $0E
C564          TYPE         =  $0D
C564          V1ADDS       =  $FB
C564 00       NUMTYP      BYT $00
C565 00       VARTYP      BYT $00
C566          !
C566          !
C566 E000     SWAP        CPX #$0             !CHECK STATEMENT FLAG
C568 F003                 BEQ SWAPOK
C56A 4C08AF               JMP $AF08           !SYNTAX ERROR
C56D A901     SWAPOK      LDA #$01            !CHECK STACK SPACE
C56F 20FBA3               JSR CHKSTK
C572 A900                 LDA #$0
C574 8510                 STA INTARY          !ALLOW INT/ARRAYS
C576 208BB0               JSR FNDVAR          !GET 1ST VARIABLE
C579 85FB                 STA V1ADDS          !SAVE VARIABLE ADDRESS
C57B 84FC                 STY V1ADDS+1
C57D A50D                 LDA TYPE            !SAVE VARIABLE TYPE
C57F 8D65C5               STA VARTYP
C582 A50E                 LDA NMTYPE          !AND NUMBER TYPE
C584 8D64C5               STA NUMTYP
C587 20FDAE               JSR TSTCOM          !CHECK FOR COMMA
C58A 208BB0               JSR FNDVAR          !GET 2ND VARIABLE
C58D A50D                 LDA TYPE            !COMPARE TYPES
C58F CD65C5               CMP VARTYP
C592 F005                 BEQ TSTNUM
C594 A216     TYPMIS      LDX #$16            !TYPE MISMATCH
C596 4C37A4               JMP ERRORS
C599 C900     TSTNUM      CMP #$0             !NUMERIC TYPE?
C59B D00F                 BNE STGVAR          !MOVE STRING
C59D A50E                 LDA NMTYPE          !COMPARE NUMBER TYPES
C59F CD64C5               CMP NUMTYP
C5A2 D0F0                 BNE TYPMIS
C5A4 C900                 CMP #$0             !FLOATING POINT?
C5A6 F008                 BEQ FPNVAR          !MOVE FP NUMBER
C5A8 A202                 LDX #$02            !MOVE TWO BYTES
C5AA D006                 BNE CPYVAR
C5AC A203     STGVAR      LDX #$03            !MOVE THREE BYTES
C5AE D002                 BNE CPYVAR
C5B0 A205     FPNVAR      LDX #$05            !MOVE FIVE BYTES
C5B2 A000     CPYVAR      LDY #$0             !INITIALISE INDEX
C5B4 B1FB     COPYLP      LDA (V1ADDS),Y      !FROM VAR 1
C5B6 48                   PHA                 !ONTO THE STACK
C5B7 B147                 LDA (V2ADDS),Y      !FROM VAR 2
C5B9 91FB                 STA (V1ADDS),Y      !TO VAR 1
C5BB 68                   PLA                 !FROM THE STACK
C5BC 9147                 STA (V2ADDS),Y      !TO VAR 2
C5BE C8                   INY
C5BF CA                   DEX
C5C0 D0F2                 BNE COPYLP
C5C2 60                   RTS                 !FINISHED
```

Listing 3. The SWAP command.

The full listing for SWAP is given in Listing 3 and as with previous keywords, I'm going to stick to describing the new parts. Having checked that SWAP has been called 'legally' and confirmed that there is sufficient stack space, the first task is to allow integers and arrays by loading INTARY with $00. A pointer to 'var1' can then be obtained by calling FNDVAR. The variable address (in the A/Y registers) is saved in V1ADDS and the contents of TYPE ($0D) and NMTYPE ($0E) are saved in VARTYP and NUMTYP respectively.

After checking for the comma separator the 'var2' pointer is obtained (again by calling FNDVAR). The value of TYPE returned by 'var2' is compared with VARTYP (TYPE for 'var1'): if they are not equal then a TYPE MISMATCH error is given.

If the variable types are both string (TYPE = $FF) then the X register is loaded with $03 (move three bytes). If, however, the variable types are numeric (TYPE = $00) then the value of NMTYPE returned by 'var2' is compared with NUMTYP for 'var1'). If they are not equal, a TYPE MISMATCH error is given.

If the number types are both integer (NMTYPE = $80) then the X register is loaded with $02 (move two bytes). If the number types are both floating point ($00) then the X register is loaded with $05 (move five bytes). The final part of the SWAP routine is the copying loop itself. This loop copies the number of bytes specified in the X register from the V1ADDS location ('var1' contents) to the V2ADDS location ('var2' contents) via the stack.

NEXT MONTH

Next month's installment will be the final part of this series. In it I'll be describing the more 'obscure' ROM routines - including some examples of their use. And as you've come to expect by now, I'll also be presenting some more new and useful keyword routines.

CP/M SOFTWARE DIRECTORY

ACCOUNTS

ACTPAK
Solicitor Systems and Services,
Systems House,
127 High Street,
Beckenham,
Kent BR3 1AG
Phone 01-658 3937 and 3938
Contact: Mr. W. Freeman
Price: £1500 (includes three days on-site training).

AUTOMATIC INVOICING
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £1000

BETPLAN
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £2000

BILLFLOW
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £395

BOSS-PAYPLAN
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £1000

BOSS-PAYROLL
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £500

BUSINESS DESK
Paxton Computers Ltd,
28 New Street,
St. Neots,
Huntingdon,
Cambridgeshire PE19 1AJ
Phone: 0480 213785
Telex: 32720
Contact: Jayson Brown
Price: From £595 to £2000

COSTING LEDGERS– BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

EXACT
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £750

GENERAL LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

INTERACC
Software Aids International Ltd,
16 Norval Road,
North Wembley,
Middlesex HA0 3TE
Phone: 01-904 8139
Contact: Mr. D.V. Bull
Price: From £200

INVENTORY MANAGEMENT – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

ISL
The Byte Shop (Birmingham) Ltd
94-96 Hurst Street,
Birmingham B5 4TD
Phone: 021-622 3165
Contact: Mr. Southern
Price: £1100

NOMINAL AND PURCHASE LEDGER
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £300

NOMINAL LEDGER
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £325

NOMINAL LEDGER – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PAYROLL
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £200

PAYROLL
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £425

PAYROLL
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising
Price: £500

PAYROLL – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PURCHASE LEDGER
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £500

PURCHASE LEDGER
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £325

PURCHASE LEDGER
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising
Price: £500

PURCHASE LEDGER – BUSINESS MANAGEMENT SYSYEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PURCHASE LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

SALES LEDGER – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

SALES LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

DATABASES

BT-80
This is a Digital Research product and should be available from most software suppliers.

CARDBOX
Caxton Software Publishing Company Ltd,
10-14 Bedford Street,
Covent Garden,
London WC2E 9HE
Phone: 01-379 6502
Contact: W.D. Barrow
Price: £195

CONDOR
MOM Systems Ltd,
40-41 Windmill Street,
Gravesend,
Kent DA12 1BA
Phone: 0474 57746
Telex: 965492
Price: From £195 to £650

DATAFLEX
Equinox Computers,
16 Anning Street,
New Inn Yard,
London EC2A 3HB
Phone: 01-739 2387
Telex: 27341
Contact: Mike Kusmirak
Price: Single-user £495, multi-user £595

DATAFLOW
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising

DATAFLOW
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £185

DATA MANAGEMENT SYSTEM
Compsoft Ltd,
Hallams Court,
Shamley Green,
Near Guidford,
Near Guildford,
Surrey
Phone: 0483 898545
Contact: Heather Kearsley,
Price: £195

MICROPEN
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £125

DATASTAR
This is a MicroPro International product and should be available from most software suppliers.

DATA STORE ONE SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

dBASE II
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

dBASE II DATABASE MANAGEMENT SYSTEM
Sussex Microsystems Ltd,
43 East Street,
Horsham,
West Sussex
Phone: 0403 68071
Contact: Keith Gale
Price: "under £500"

FILEFORCE
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £250

GUARDIAN
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILING ADDRESS
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILING SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

MAILMASS
Vision Associates Ltd,
57 Woodham Lane,
New Haw,
Weybridge,
Surrey,
Phone: 0932 55932
Contact: Graeme Sleeman
Price:

PRISM
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

TRENDISK/1c
Microtrend UK,
PO Box 51,
Pateley Bridge,
Harrogate,
North Yorkshire HG3 5DF
Phone: 0423 711878
Contact: Ian Cummings
Price: £150

GRAPHICS

DISPLAY MANAGER
This is a Digital Research product and should be available from most software suppliers.

DATAPLOT III
Grafox Ltd,
35 St. Clements,
Oxford OX4 1AB
Phone: 0865 242597
Telex: 83147
Contact: Geordie Herbert
Price: £495

DATAPLOT+
Grafox Ltd,
35 St. Clements,
Oxford OX4 1AB
Phone: 0865 242597
Telex: 83147
Contact: Geordie Herbert
Price: £195

LANGUAGES

ALGOL-60
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

APL V80
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

BAZIC
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £124

C-BASIC
This is a Microsoft product and should be available from most software suppliers.

CBASIC COMPILER
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £333

CIS COBOL
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £425

COBOL-80
This is a Microsoft product and should be available from most software suppliers.

COMAL
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £145

POLYFORTH
Computer Solutions Ltd,
Treway House,
Hanworth Lane,
Chertsey,
Surrey KT16 9LA
Phone: 093 28 65292
Contact: Chris Stephens
Price: From £750 to £2750

FORTRAN-80
This is a Microsoft product and should be available from most software suppliers.

JRT PASCAL
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

KBASIC
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

LISP-80
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £50

M-BASIC
This is a Microsoft product and should be available from most software suppliers.

PASCAL/M
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PASCAL/MT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PASCAL/MT+
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £233

PASCAL/Z
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PRO PASCAL
Prospero Software,
37 Gwendolen Avenue,
London SW15 6EP
Phone: 01-785 6848
Contact: Tony Hetherington
Price: £220

PRO FORTRAN
Prospero Software,
37 Gwendolen Avenue,
London SW15 6EP
Phone: 01-785 6848
Contact: Tony Hetherington
Price: £220

UCSD PASCAL
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MODELLING

CALCSTAR
This is a MicroPro International product and should be available from most software suppliers.

FASTPLAN
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £395

PEACHCALC
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £200

PLAN80
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

SUPERCALC
Tamsys Ltd,
12a Sheet Street,
Windsor,
Berkshire SL4 1BG
Phone: 075 35 56747
Contact: David Atkinson
Price: £129

SUPERCALC I
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £126

SUPERCALC II
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £195

PROGRAMMING

ACCESS MANAGER
This is a Digital Research product and should be available from most software suppliers.

ANIMATOR
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: From £225

BASIC UTILITY DISK
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

BASKAM
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £95

COBOL DEVELOPMENT SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: Full system £550

CLIP
Tamsys Ltd,
12a Sheet Street,
Windsor,
Berkshire SL4 1BG
Phone: 075 35 56747
Contact: David Atkinson
Price: £95

FORMS UTILITY GENERATOR
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £120

PSORT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

QSORT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

SPP
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £133

T/MAKER II
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £165

TRANSAM SCIENTIFIC SUBROUTINES
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £495

TYPESTAR
Sussex Microsystems Ltd,
43 East Street,
Horsham,
West Sussex
Phone: 0403 68071
Contact: Keith Gale
Price: £95

XLT86
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £100

WORD PROCESSING

BENCHMARK
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham

BRITISH SPELLGUARD
Vision Associates Ltd,
57 Woodham Lane,
New Haw,
Weybridge,
Surrey,
Phone: 0932 55932
Contact: Graeme Sleeman
Price: £179

LETTERIGHT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

LEXICOM/2
Microtrend UK,
PO Box 51,
Pateley Bridge,
Harrogate,
North Yorkshire HG3 5DF
Phone: 0423 711878
Contact: Ian Cummings
Price: £350

MAGIC WAND
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILMERGE
This is a MicroPro International product and should be available from most software suppliers.

PEACHTEXT
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £250

SPELL
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £55

SPELLBINDER
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £275

SPELLING PROOFREADER
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

SPELLSTAR
This is a MicroPro International product and should be available from most software suppliers.

TEXTWRITER III
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

WORD PROCESSING SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

WORDSTAR
This is a MicroPro International product and should be available from most software suppliers.

LUCAS MICRO

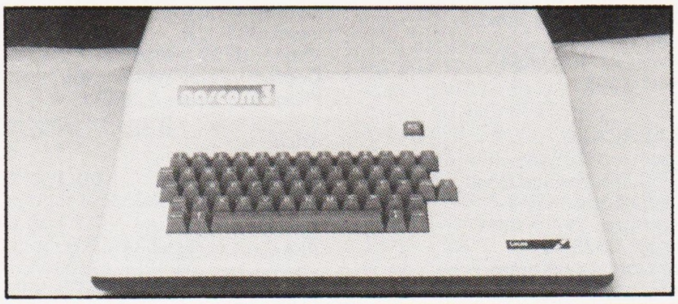
LUCAS LX

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 Z80A microcomputer aimed more at the professional and business user. Hence 5Mb 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 14K 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 784 by 256 (two colours) 392 by 256 (four colours) COLOUR 8 TEXT 25 by 80
SOUND	optional



DEALER CBM MICRODEALER

COMMODORE 720

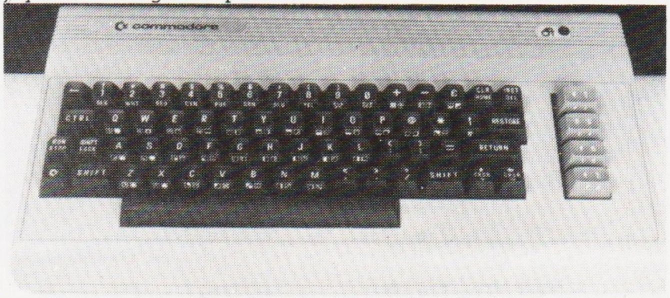
MEMORY	256K 20K ROM
LANGUAGE	Commodore BASIC
CASSETTE	300 baud
DISC	Twin in-built floppy drives
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☑ FUNCT ☑
DISPLAY	TV ☐ MONITOR SUPPLIED ☑
INTERFACE	PARA ☑ SERIAL ☑ BUS ☐
GRAPHICS	BLOCK ☑ USER ☐
	LINE ☐ RES 80 by 25
	COLOUR 16 TEXT 80 by 25
SOUND	**Three channels**

Notes. The Commodore 720 is the top model in the 700 range of business machines. It is built round the 6509 processor, but there is a dual processor (Z80 or 8088) option. The machine has been designed to meet the IEC specifications. The black-and-white monitor screen is integral and features tilt and swivel. The keyboard may be detached. The dual disc drives are built-in to the main housing and use DMA transfer, increasing speed.

COMMODORE 64

MEMORY	64K RAM 26K ROM
LANGUAGE	PET BASIC
CASSETTE	300 baud
DISC	extra DOS
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☐ FUNCT ☑
DISPLAY	TV ☑ MONITOR SUPPLIED ☐
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑
	LINE ☐ RES 80 by 25
	COLOUR 16 TEXT 40 by 25
SOUND	**Three channels**

Notes. The Commodore 64 is a 6510 based micro that can also use Pascal, COMAL, LOGO, FORTH and PILOT. Programs can be loaded from cassette recorder or disc drives, both extra, or cartridges. The various peripherals include printer, joysticks and games paddles.

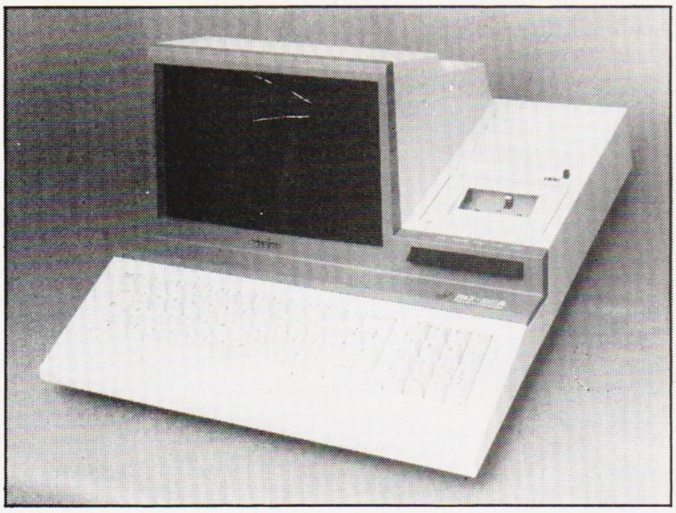


SHARP MICRODEALER

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.

BERKSHIRE

Kuma Computers Limited
Fully support with software and hardware. Sharp 700, Sharp MZ80A, and MZ80B.
11 York Road, Maidenhead, Berks. Phone for details — **0628 71778**

COMPUTER 100
181 Oxford House, Reading, Berks RG1 7UZ
TEL: 0734 591616
One of the UKs largest stockists

CAMBRIDGESHIRE

MS CONSULTANTS (CAMBRIDGE) LTD
Bridgefoot House, 159 High St., Huntingdon PE18 6TF. Tel: 0480 51721
Specialists in Sharp Hardware & Software

LANCASHIRE

Sumita **12-14 Avenham St. Preston (0772) 51686**
Sharp MZ-3541 Based Software, Bakehouse Orders System, Barstock System.
SALES — SERVICE SOFTWARE

LINCOLNSHIRE

SHARP CENTRE
16 Melville Street, Lincoln.
Tel: Lincoln 32379
All Sharp Microcomputers. Specialists in Industrial Process Control Systems. Customers include British Telecom, M.O.D. CRA & CTA Members

LONDON

SHARPSOFT LTD.
With the Sharp user in mind.
Sharp MZ 700 including cassette £199.99
Sharpsoft Ltd, Crisallen House, 86-90 Paul Street, London EC2.
Tel: 01 - 729 5588.

MICROS FOR MANAGERS
149 Gloucester Rd., London SW7 4TH
New program for PC1251: cost estimatic and stock additive. PC5000 stock £1639. Ask us for portable and pocket computer hardware and software.
Tel: 01-370 5125 for appointment

MIDLANDS

JAXREST LTD
Linton House, Catherine Street, Aston, Birmingham.
Tel: 021 328 0543
Sharp Service and Support

NORFOLK

GRASSROOT COMPUTERS
25 Wellington Rd., Dereham.
TEL: 0362 4640
Specialist in small business computing. Send for details of stockbox — the complete and thorough stock system with 2,500 items per disc. Other modules to follow.

SCOTLAND

micro change ltd
170 Eastwood Mains Road, Clarkston, Glasgow G76. Tel: 041- 554 7623
SHARP ALTOS SUPERBRAIN
Call us for your software and hardware requirements — We will tailor our data base to meet these.

LOWDATA SYSTEMS
A division of Lowdon Bros. & Co (Engineers Ltd). PO Box 53, Blackness Road, Dundee, DD1 9JG.
Tel: (0382) 22314
Full range of Sharp Equipment stocked

WALES

STEVE'S COMPUTER COMPANY LTD
CASTLE ARCADE, CARDIFF.
Tel: 0222 — 371578
OPEN: Mon — Sat. 9 - 5.30
For Components to Computers

WEST SUSSEX

MICROCENTRE LTD.
SYSTEMS, SOFTWARE SERVICE, SUPPORT
28-30 Station Road, Bognor Regis, Sussex.
TEL: 0243 827779

YORKSHIRE

PROGRAM 1
For Business Software
35 Albion Street, Hull.
TEL: 0482 20022

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.

CORNWALL/DEVON

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

HERTFORDSHIRE

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

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.

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

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

COMPUTING +
264 Earls Court Road, London SW5. Tel: 01 373 4508 or 01 373 5000 Ext 240. Suppliers of mini and micro computers for home and office use, together with supporting software. Leading manufacturers' products supplied.
Come and see the Mackintosh.

SOUTH LONDON

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

MIDDLESEX

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.

SCREENS MICROCOMPUTERS
6 Main Avenue, Moor Park, Northwood Middlesex.
Tel: Northwood (09274) 20664
Official Dealers for: Acorn, Atari, Commodore, Dragon, Memotech, Oric, Sinclair, Sirius, Apricot.

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
ZX81 REPAIR SERVICE

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

SUSSEX

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

TYNE AND WEAR

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

WALES

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

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

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

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)

ADVERTISERS INDEX

COMPUTING TODAY

Lineage: 40p per word.
Semi display: £9.00 per single column centimetre
Ring for information on series bookings/discounts.

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

01-437 0699

Send your requirements to:
IAN ATKINSON
ASP LTD, 1 GOLDEN SQUARE,
LONDON W1

SOFTWARE APPLICATIONS

Extender: A Machine Code Utility Program For The LII TRS80 & Video Centre comprising of:
BASYST: Converts Basic programs into System format with six letter file names. CLOSE: Closes program off prior to MERGEing. DATA COMPILER: Compiles data statements between two memory addresses. FIND & RESCUE: Reinstates Basic program, even when NEWed. HEX TO DEC: Converts 1 or 2 byte hexadecimal number to decimal equivalent. LOWCASE: Changes high case letters to lowcase. MERGE: Merges two programs together. RENUMBER: Renumbers whole or part of Basic program. ONESTROKE: Allows multiple entry of defined keywords. PUT: Puts a 1K block of memory to screen. SQUASH: Removes all unwanted spaces in program listing. RUN: Runs Basic program with single keystroke. SINGLE & PAGE: Allows single line or page scrolling. SYSTEM TAPE COPIER: Allows the copying of all system tapes. VALIDATE: Validates all line branches in listing.
£9.95 from:
OLYMPIC SOFTWARE,
15 Arnhem Close, Aldershot, Hants, GU11 1RJ.

SPECTRUM TAPE COPY. Fully assembled and commented machine code listing with Hex loader and full instructions. Cheques for £2.00 etc to D. Woodthorpe, 64 Abingdon Avenue, Lincoln.

M & J SOFTWARE
fig-Forth Assembly Source Listings .... £7 each
Available for the following processors: 6502,Z80, 6809, 8080, 1802, 9900, 6800, 8086/88 and PDP11.
fig-Forth Installation Manual £5
A complete 'how to do it' guide to the implementation of FORTH from the above listings.
Cheques and POs to:
M & J SOFTWARE, 34 Grays Close, Scholar Green, Stoke-on-Trent ST7 3LU.
Tel: (0782) 517876

COMMODORE 64 "SPRITER"
Enables single or multicolour sprites to be designed interactively, compiled and saved to tape. Tape **£3.50** to:
P. STRATFORD
43 Chanctonbury Drive, Shoreham, Sussex BN4 5FR

ACCESSORIES

FERRANTI
I.Te.C.
BBC MICRO
CASSETTE LEADS
(a) 7 pin din to 7 pin din
(b) 7 pin din to 2 x 3.5 mm & 1 x 2.5 mm Jacks
(c) 7 pin din to 3 pin din & 1 x 2.5 mm Jack
ONLY £2.25 ea inc p&p/VAT
Ferranti Oldham ITeC
Department 101, Orme Mill, Crimbles Street, Waterhead, Oldham OL4 3JA

MRL PRINTERS
Chinwa CP80 195 (224 inc VAT). Star Gemini 10X (120 cps 80 col) 215 (247 inc VAT). Star Gemini 15X (120 cps 132 col) 335 (385 inc VAT). Star Delta 10 (160 cps 80 col) 359 (413 inc VAT). Juki Daisywheel (18 cps) 369 (424 inc VAT). Plus paper, ribbons & refills etc. Free delivery by Securicor.
RING 0506 31605 FOR DETAILS
MRL Printer Interfaces
Available to fit most micros: Electron, Spectrum, Atari, Commodore 64 incl free cable, free delivery. From **£39.95** inc VAT.
RING 0506 31605 FOR DETAILS
Selection of Disc Drives at bargain prices from **£135** (inc VAT) to fit BBC, Tandy, Nascom etc.
MICRO RESEARCH (Freepost)
8 Napier Square, Houston Industrial Estate, Livingstone, West Lothian EH54 5DG

ALARMS

BURGLAR ALARM equipment. Please visit our 2,000 sq. ft. showrooms or write or phone for your free catalogue. C.W.A.S. Ltd., 100 Rooley Avenue, Bradford BD6 1DB. Telephone: 0274 731532.

SIT'NS VACANT

Are you
A MICRO ENTHUSIAST
Technically competent with a professional approach?
Have a current driving licence?
THEN you may be interested in working as an engineer on IBM, ACT, DEC and SANYO personal computers, as well as a variety of printers and other devices, based at new premises in the West End of London. Please send full details in writing to **SCC, Litchfield House, 85 Smallbrook, Queensway, Birmingham B5 4JF.**

ACCESSORIES

BLANK CASSETTES
Guaranteed top quality computer/audio cassettes at great budget prices.
Packed in boxes of 10 with labels, inlay cards and library case.
Prices include VAT, post and packing.

☐ (C5)	£4.35	☐ (C10)	£4.40
☐ (C12)	£4.45	☐ (C15)	£4.50
☐ (C30)	£4.70	☐ (C60)	£5.30
☐ (C90)	£7.00		

BASF FLOPPY DISCS
Prices of boxes of 10
☐ 5¼ Single side/ **Single** density £18
☐ 5¼ Double side/Double density £19
☐ 5¼ Double side/Quad density £23
DISC DRIVES
Include Manual, Leads, Utilities Disc
☐ TEAC 55A 40 tracks – £139 each
☐ TEAC 55F 40/80 switchable D.S. – £209 each **FREE DELIVERY UK ONLY**
Indicate quantity of each product required in boxes
Cheque/P.O. enclosed for £ ....
NAME ....
ADDRESS ....
PROFESSIONAL MAGNETICS LTD
Cassette House, 329 Hunslet Road, Leeds LS10 3YY
FREEPOST Tel. (0532) 706066 **CT**

COMPUTER CABINETS
A range of cabinets and housing units especially designed for your home computer system. Keep it dust free, tidy and secure.
Send SAE for colour brochure
MARCOL CABINETS
Solent Business Centre, Millbrook Road West, Southampton
Tel: 0703 731168

HARDWARE

NEWBRAIN & SANYO
Professional Micro Computers for the price of hobby machines.
NEWBRAIN ON SPECIAL OFFER
With over £200.00 free software. (Accounts, databases, etc, etc)
Limited Offer-ring now!
SANYO 550/555 COMPUTERS
Micropro Wordstar, Calcstar etc at no extra cost! **Printers:** Epson, KDC, Juki, Shinwa, Daisystep 2000 etc. **Sanyo Monitors & Recorders.**
Call STEVENAGE (0438) 812439 anytime for hardware/software lists
Mail Order and **Access** facilities.
ANGELA ENTERPRISES
4 Ninnings Lane, Rabley Heath, Welwyn, Herts AL6 9TD.

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.

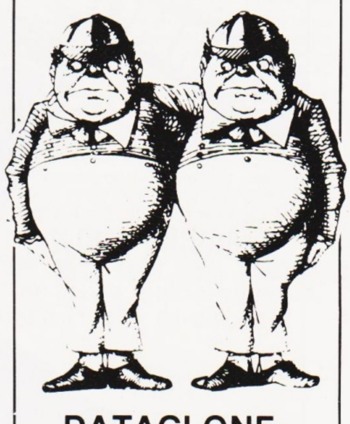
The result?

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: 01-993 2134 Telex: 21879

BOOKS AND PUBLICATIONS

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

HARDWARE AND SOFTWARE

COMPUTING +

264 Earls Court Rd, London SW5. Tel: 01-373 4508 or 01-373 5000 Ext 240.

Suppliers of mini and micro computers for home and office use, together with supporting software. Leading manufacturers' products supplied.

Just in from APPLE the MACKINTOSH and 2C bringing Hi-TECHNOLOGY to your fingers.

FOR SALE

SHARP MZ80A with Seikosha printer. Excellent condition still boxed, little used. Various software and manuals. Best offer over £600. Telephone Amersham 22130.

19Mb HARD DISC Drive with power supply. User manual and application guide. Good working condition £175 delivered. Tel: (02602) 71290.

ARCADE GAME P.C.B.S. original Hustler, Frogger, Kong, Defender etc. Power supplies, R.G.B monitors new or used. Phone N'pton 0604 716646.

ACORN ATOM: fully expanded, floating point power supply. As new £100.00. Tel: 01-346 7055.

OLIVETTI TE-300 teleprinter with stand. Very good condition. £70. Telephone 061-748-7160.

ATARI 800 64K computer complete with 2 joysticks, cassette dec, 3 cartridges, 4 tapes, literature all new. Selling because emigrating £300ono. Phone evenings 670 4625.

SOFTWARE HOME FINANCE

ORIC/ATMOS personal finance package for home, club or small business. Details from: Softbacks (Dept CT), FREEPOST, Watford WD1 8FR (no stamp needed).

WANT TO FILL THIS SPACE?

RING IAN NOW ON 01-437 0699

CT WOULD LIKE TO APOLOGISE FOR ANY INCONVENIENCE CAUSED TO CUSTOMERS OF G.C. BUNCE AND SONS BY AN INCORRECT TELEPHONE NO. IN THE MAY/ JUNE ISSUES. THE CORRECT TELEPHONE NO. IS 06286 61696

SOFTWARE GAMES

ATARI 400/800
TRS80 1/3 GENIE I/II

Arcade and adventure games, educational and utility programs. On tape. SAE for details. Please state micro.

T. Smith Software (Dept CT)
20 Wesley Road, Portsmouth, Hants PO3 5ER

MAIL ORDER ONLY PLEASE

747 PILOT for VIC20 with super expander cartridge or new 8K/16K expansion RAM version. £6.99. Victay (CT), 12 Leahill Close, Malvern, WR14 2UE.

COMPUTER SOFTWARE and accessories, send 2 x 16p stamps for free lists stating which computer to: Electronic Facilities, 3 High Street, Sandy, Beds SG19 1AG.

REPAIRS

COMMODORE REPAIRS

by CMB approved service engineers: for all out of guarantee units. Eg. Vic 20, CBM 64, C2N Datasette, printers, Vic 20 modulators, disc units etc. For more details tel or SAE:

G. C. Bunce & Son
36 Burlington Road, Burnham, Bucks SL1 7BQ.
TEL: (06286 61696)

COMMODORE 64

OXFORD PASCAL

YOUR FUTURE WITH COMPUTERS DEPENDS ON PASCAL

BECAUSE

- PASCAL is the world's most popular programming language after BASIC
- PASCAL is FAST... up to 20 times the speed of BASIC

WHAT IS OXFORD PASCAL?

OXFORD PASCAL is a COMPLETE implementation of the popular programming language PASCAL running on the Commodore 64

GRAPHICS & SOUND

With the OXFORD extensions to PASCAL you can programme dazzling graphics and astounding sound

WHAT DO I GET?

With the cassette version of OXFORD PASCAL you get:

- A powerful text editor for preparing PASCAL programs
- A resident compiler giving 14k of user RAM
- A powerful set of colour graphics and sound extensions to standard PASCAL

With the disk version of OXFORD PASCAL you get:

- All the above, PLUS
- A full disk to disk compiler exploiting the entire memory of the 64 for PASCAL programs.

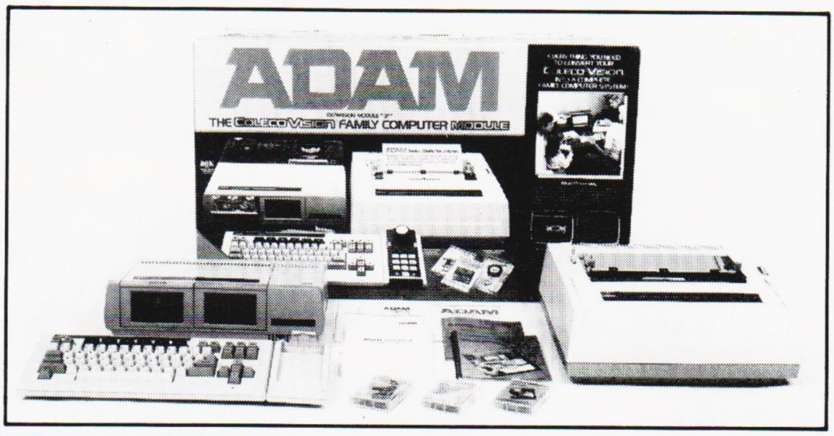
HOW MUCH DOES OXFORD PASCAL COST?

Cassette version (resident compiler only £14.95
Full disk compiler (1541, 4040, 8050, 8250 drives) £49.95
Prices are exclusive of VAT. There is also a small charge for post and packing.

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

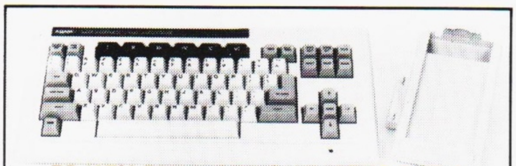
A COMPLETE PACKAGE - ALL THIS FOR £499!

* 80K* RAM (Exp to 144K)
* Full Stroke Keyboard
* 256K Data Storage Unit
* Daisywheel Printer
* Built-in Word Processing
* Buck Rogers Arcade Game
* Colecovision Compatible

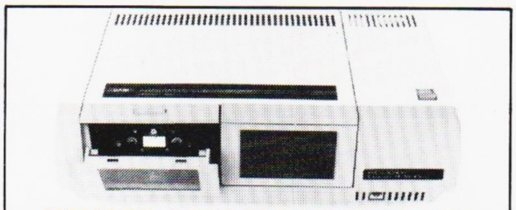


ADAM™ - £499

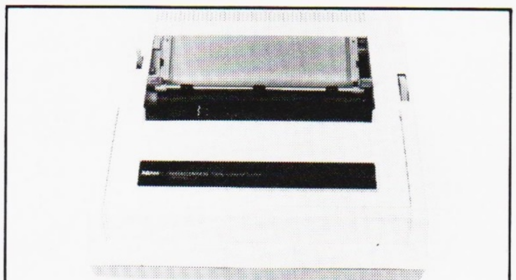
Inc VAT



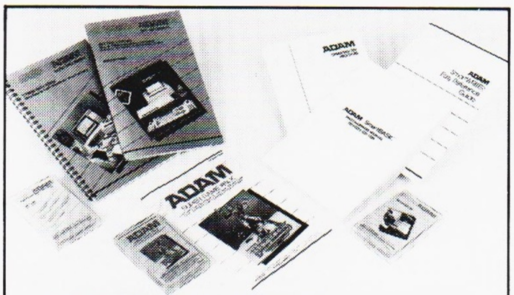
75 KEY FULL STROKE KEYBOARD



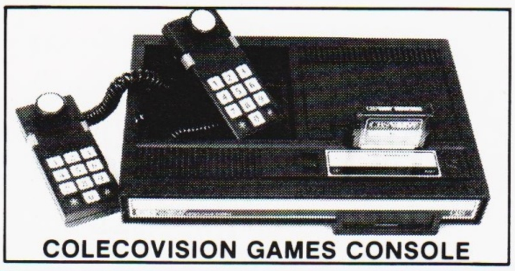
MEMORY CONSOLE & DATA DRIVE



DAISYWHEEL PRINTER



COMPREHENSIVE INSTRUCTIONS



COLECOVISION GAMES CONSOLE

QUITE SIMPLY - VALUE FOR MONEY!

If you're looking for real value in a computer system, one which can handle anything from serious Word Processing to enhanced Colecovision style video games such as Buck Rogers, look no further. The Coleco Adam is here with a package which will make you wonder if you're dreaming when we tell you about it. A price breakthrough in computer systems, Adam is comprised of an 80K RAM memory console with a built-in 256K digital data drive; a professional quality, stepped and sculptured 75 key full-stroke keyboard; a letter quality daisywheel printer and a full word processing program built into the Console. Two additional pieces of software, Smart BASIC and also 'Buck Rogers - Planet of Zoom' (the ultimate in advanced video games), are included as well as a blank digital data pack. Adam can be used with any domestic colour Television set.*

MEMORY CONSOLE/DATA DRIVE: *The heart of the Adam system is the 40K ROM and 64K RAM memory console which combines with the 32K ROM and 16K RAM in Colecovision to give you a total of 72K ROM (including 24K cartridge ROM) and 80K RAM (expandable to 144K). Built into the memory console is a digital data drive which accepts Adam's digital data packs, a fast and reliable mass storage medium that is capable of storing 256K of information, that's about 250 pages of double spaced text! The console is also designed to accomodate a second optional digital data drive.

FULL STROKE KEYBOARD: The Adam keyboard has been designed as a professional quality keyboard that combines ease of use with an impressive array of features. It is stepped and sculptured for maximum efficiency and has 75 full stroke keys which include 6 colour coded Smart Keys which are redefined for each new application; 10 command keys which are dedicated to the word processing function, and 5 cursor control keys for easy positioning of the cursor at any point on the screen. You can attach a Colecovision controller to the keyboard to function as a numeric keypad for easy data entry. It can also be held like a calculator, a feature which makes working with numbers particularly easy. The joystick part of the hand controller can be used in the same way as the cursor control keys, to move the cursor around the screen.

LETTER QUALITY PRINTER: The SmartWriter letter quality daisywheel printer is a bi-directional 80 column printer which prints at a rate of 120 words per minute. It uses standard interchangeable daisywheels, so a variety of typestyles are available. The printer has a 9.5 inch wide carriage for either single sheets or continuous fan fold paper and uses standard carbon ribbons. It is comparable to many printers which cost as much as the total Adam package. The printer can be used either with the Adam's SmartWriter word processing program or as a stand alone electronic typewriter.

BUILT-IN WORD PROCESSOR: Adam comes with SmartWriter word processing built-in. This program is so easy to use that you only have to turn the power on and the word processor is on line and ready to go. Detailed instruction books are not necessary as the Computer guides you step by step, working from a series of Menu commands. It enables you to type in text, then completely edit or revise it with the touch of a few keys. Changes are readily made and a series of queries from the computer confirm your intentions, so that you can continuously double check your work as you type.

COMPATIBILITY WITH COLECOVISION: By using high speed interactive microprocessors in each of the modules, the Coleco Adam is designed to take additional advantage of both the 32K ROM and 16K RAM memory capability in the Colecovision. If you do not already own a Colecovision Console (£99 inc VAT), then you will need to purchase this when you initially purchase your Adam Computer package (£499 inc VAT), making a total purchase price of (£598 inc VAT).

WHAT IS COLECOVISION: Colecovision is one of the worlds most powerful video game systems, capable of displaying arcade quality colour graphics of incredible quality on a standard Colour TV set. The console (see picture bottom left) accepts 24K ROM cartridges such as Turbo and Zaxxon and is supplied with the popular Donkey Kong cartridge and a pair of joystick controllers. Colecovision has a range of licenced arcade hits available such as: Gorf, Carnival, Cosmic Avenger, Mouse Trap, Ladybug, Venture, Smurf, Pepper II, Space Panic, Looping, Space Fury, Mr Do, Time Pilot, Wizard of Wor and many others. So there you have it, Adam plus Colecovision the unbeatable combination. Send the coupon below for your FREE copy of our 12 page Colour brochure giving details on the complete Adam system.

SILICA SHOP LTD., 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX Tel: 01-309 1111 or 01-301 1111

ORDER NOW - OR SEND FOR A FREE COLOUR BROCHURE

To: SILICA SHOP LTD, Dept CT 0884, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX Telephone: 01-309 1111 or 01-301 1111

LITERATURE REQUEST:

☐ Please send me your FREE 12 page colour brochure on Colecovision/Adam
☐ I own a .... Videogame ☐ I own a .... Computer

Mr/Mrs/Ms: .... Initials: .... Surname: ....

Address: ....

....

....

.... Postcode: ....

ORDER REQUEST:

Please send me: ☐ Adam (add-on package only) .... £499 inc VAT
☐ Adam & Colecovision (£499+£99) ..... £598 inc VAT

☐ I enclose Cheque/P.O. payable to Silica Shop Limited
☐ CREDIT CARD - Please debit my Access/Barclaycard/Visa/Am Ex/Diners Club

Card Number: ....

Rev 2