1/9

PERSONAL COMPUTER WORLD

50p U.S.$2.00

JANUARY 1979

USING THE MICRO TO INCREASE HOME AND OFFICE EFFICIENCY

Europe's first magazine for personal computers for home and business use

The Applesoft Touch

Play On....

Super Scamps VDU

This Year, Use Your Own Computer

PERSONAL COMPUTER WORLD

UK 50p
US $2.00

Vol 1, No 9
January 1979

Europe's first magazine for personal computers for home and business use

ISSN: 0142 0232

CONTENTS

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01 - 229 5599 (Publisher)
01 - 727 8758 (Editorial and Advertising)

Publisher:
A Zgorelec
Editor:
Meyer N. Solomon
Editorial Assistant:
Roger C. Wilkins
Policy Advisor:
Peter Crofton-Sleigh, FRAS

Consultants:
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R. W. Davy, David Hebditch, Sheridan Williams, Stephen Castell
Art: Sauveur Laurent Sant, Kathryn Hamme
Secretarial:
Vanessa Blackburn Kiddle

Layout Consultants:
D. Norris, T. Gabos
PCW Photography: Yoshi Imamura, Peter McGee
Typesetting & Artwork: Gilfillan
Cover: Portrait of Phreakstein, Welcoming in the New Year, by Sant.

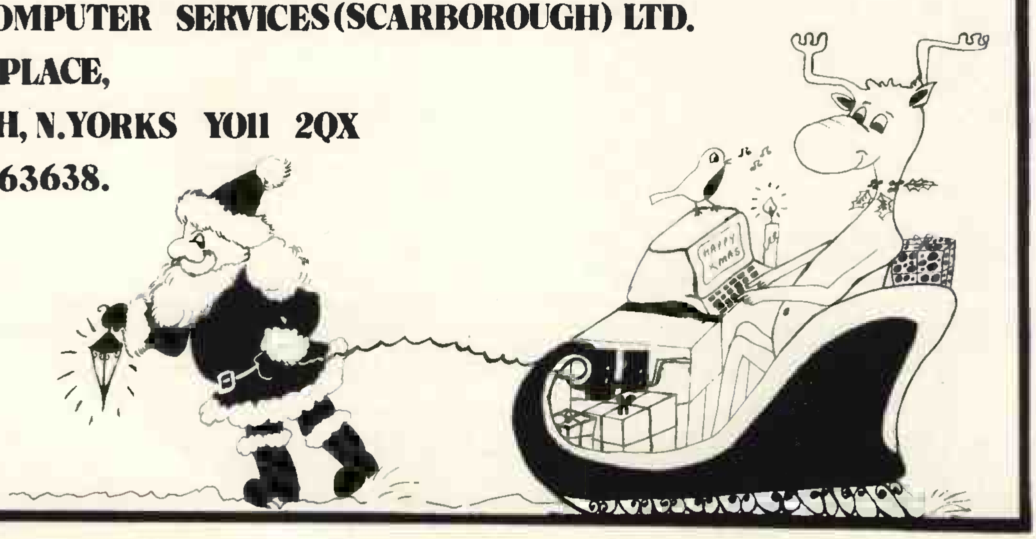
CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

We may make arrangements to offer our readers products at special prices, for a limited period, in line with the policy outlined above.

Published monthly by Intra Press, 62A Westbourne Grove, London W2. Phone: 01-229 5599. Subscription rates: Britain £8 for 12 issues. Prices include postage. USA – $10 for six issues, $20 for 12 issues. Continent and elsewhere: £9.80 for twelve issues. Prices include postage.
Printed by Carlisle Web Offset, 55 Condult Street, Newtown Trading Estate, Carlisle CA2 7NR. Sole UK Distributors: Seymour Press Ltd., 334 Brixton Road, London S.W.9., England. Distribution to specialist shops by Intra Press.



Editorial

Readers have at various times written in to us complaining about inordinate delays in receiving goods they've paid for. In our turn, we have photo-copies taken of their letters and forward them with a request for action. Usually, but not always, prompt action is taken. But in the case of one company, we were on the verge of naming it. The product it sells is very good value for money but because demand is greater than it can handle service is slow. To compound this, the company does not keep people informed, and sometimes ignores letters. Altogether, not a happy situation for the customers. Recently, however, it seems to have pulled up its socks. We sent a batch of letters to it and received a reply detailing what action had been taken – the only kind of acceptable action, giving customers their goods.

So, readers can be sure that we do try on their behalf.

We do apply pressure. PCW is now a power in the computer scene; but with power comes responsibility. To rush and name a company, without giving it a chance to explain or act, is not our policy. In this respect, we err on the side of caution. Had the company been named, the censure might have rubbed off unfairly on its product.

We ourselves have been sometimes lax in replying to letters. I take the responsibility for that and can only say that we here are working under tremendous pressure. We are fortunate to have readers who are, before anything, fond of PCW.

Publisher's Letter

Dear Reader,

It's well known that the sincerest form of flattery is imitation. We have a new game at PCW: spot the borrowed idea or phrase in other publications. On a scale of one to ten, we've scored maximum points twice.

Of course, requests for reprints and quoting from PCW are different, and we have a policy of giving permission gladly, so that PCW's presence is felt from schools to the United Nations.

PCW is the magician among magazines.

SUBSCRIPTIONS

When **PCW** started publication, we had a special six-issue offer. When these subscriptions expired, we sent out reminders.

The renewal rate was 70%!

PCW reader loyalty is becoming a byword in publishing. If you're having difficulty in obtaining **PCW** at your newsagent, take our subscription. You can find the details at the foot of P.3.

Editorial Assistant

Your Editor is in dire need. Pressure of work and plans for expansion have created a position for an *assistant to the editor.*

If you are good at small computer hardware, building kits, and expressing yourself clearly, write to the Editor with details about yourself.

I BYTE

BUT COMPUTERS DON'T

Come and see for yourself at

the BYTE SHOP Ltd

Stockists of the largest range of micro computers in the U.K.

Take the opportunity to experiment with and get to know any of the vast range of micro computers always in stock at The Byte Shop.

Whether you want a micro computer for your home, your business, for industry, for education - or if you'd just like to find out which model you get on with best - you'll find a visit to The Byte Shop a new and invaluable experience.

Call in at The Byte Shop any time from Monday to Saturday. It's right by Gants Hill tube station.

The Byte Shop 426/428 Cranbrook Rd., Gants Hill, Ilford, Essex. Telex 897311
Telephone 01-554 2177

Letters

GET ZEROED – OR ELSE

In reply to the 'Puzzle Dazzle' competition sent in by Mr. Broughton and featured on the 'letters' page of the November 1978 issue of P.C.W.

The answer to the problem of how to clear all the memories of an 8080 computer as described, is simply to switch the computer off and then on again; a read/write memory being volatile.

If a program is required it is merely in the form of a reminder to the operator:

1. Switch computer off.
2. Switch computer on.

End of program, continue as normal.

Elisabeth Else, (Age 16)
33 Albany Gardens,
Hampton Lane,
Solihull,
West Midlands B91 2PT

Elisabeth, that's not quite it. Find out why, write to me, and I'll send you £2 for the interest you've shown. – Editor.

ANOTHER AEROPLANE

With regard to John Coll's article (PCW, October) titled "Pilot flies across the Atlantic", he states that in England Pilot is only available on the SWTPC 6800 machine.

However, there is a version of Pilot running on the HP2000F computer at Brighton Polytechnic, which is written in Basic. I think you will also probably find versions of this on other Hewlett Packard machines.

On a final note, I would like to congratulate you on an excellent magazine, which I will continue to buy.

Q. J. North,
92 Hanover Street,
Brighton,
Sussex

THE NUMBERS RACKET

I should like to enquire through your columns whether any of your readers are interested in (simple) number theory, i.e. playing with numbers, and also possess a computer and, most important, have spare capacity and the time to assist me with some research.

The field I am working in is one that may appeal to schools as well as to individuals because it has the same sort of simplicity coupled with a wide range of interesting problems as has the study of prime numbers. The work to be done would involve writing programs, from flowcharts which I would supply, as I am unable to do this myself.

Anyone who is interested is invited to write to me at this address for further details.

P. A. Newton Webb,
31 Cleviston Park,
Llangennech,
Llanelli,
Dyfed

ELLIOTT 903 USERS

In answer to Sheridan Williams' letter in your August issue, "Not so small but just as beautiful": Yes, there is an active Elliott 900 Users Group. For many years the group consisted mainly of industrial & research users, but the most active members nowadays are schools or individuals who have bought second-hand machines.

Prices vary: the precious metal scrap value of a 903 is around £100. One 903 with a 16K x 18bit store, high-speed paper punch and reader, and Teletype, changed hands for £400; the usual price for a Teletype alone! We do our best to circulate information within the group about hardware for sale: and try to keep computer brokers and scrap dealers at bay.

On the software side, the 903 is far easier to program than any 8bit micro; it has ALGOL & FORTRAN, plus the usual assemblers and editors, any of which can be run in 8K using paper tape. In addition the users group has recently implemented a quite respectable BASIC interpreter, and a compiler may follow.

If any of your readers know of any Elliott/GEC 900-Series machine for sale; or they have one in their garage and would like some software for it; I suggest they write to me.

Terry Froggatt,
Secretary, 900 Group,
Elliott Computer Users Association,
c/o 2 Skelcher Road,
Shirley, SOLIHULL,
West Midlands

PET Groupies

As you may know, a group of PET owners have decided to form an organisation for the sharing of ideas and information about the PET. The group has now taken on a formal existence and already has over 50 members.

I have taken over the job of secretary from Norman Fox who has done such a splendid job of getting the group set up.

We are hoping that as many PET owners and users as possible will join the group and come along to our meetings to share their knowledge and to learn from the experiences of others. So far we have held two meetings and I for one have learnt a great deal. We have seen the PET being used to control a variety of equipment through its user port; we have heard it playing music and we have swapped programs and ideas.

The group is deliberately independent of Commodore and we are not in competition with their own User' Club. We feel that there will be times when we wish to be very critical of Commodore, the delay in producing the long awaited printer, and the problems of head alignment on the cassette decks, being two issues where external pressure may speed things up.

We will be circulating all dealers with information concerning the group in the hope that they may be willing to supply details of our group to all new buyers. If any dealer wants further information then please get in touch with me.

We will be holding regular meetings throughout the country and we will be producing a regular newsletter with ideas from our own members and with information gleaned from elsewhere, including some of the information produced by the PET groups in the USA.

If anyone wishes to join please send an SAE to me and I will send full details.

Mike Lake,
Independent PET Users' Group,
9 Littleover Lane,
Derby

THREE HOURS, SEVENTEEN MINUTES, THIRTY EIGHT SECONDS – for Lunch

As an avid reader of your magazine and one not possessing two chips to rub together I am always on the lookout for routines that can be applied to my TI 57 programmable calculator.

In the November issue my eye was caught by the long listing for evaluating a square root (by R. E. L. Ferguson). I felt sure I could program my calculator for this, and so I did. It required 30 steps.

As the article mentioned that the 6800 takes less than 1.5 secs for evaluating $\sqrt{99999}$ I thought you might like to know that for the given algorithm my calculator takes 3 hrs 17 mins 38 secs, which gives me ample time for a lunch break. Of course, if I'm in a hurry I use the square root key.

L. R. Carter,
Management Tutorials,
1 Strathmore Drive,
Charvil,
Reading RG10 9QT

THE CASE FOR AUTOMATIC RELOCATION

A. Clements (November) has produced a very interesting article that gives much food for thought. However, as with most authors, it shows a bias that presumably reflects his special interests and experience.

For example block moves are treated as a trivial requirement yet at least one large computer has a special single instruction for this purpose, giving very fast execution and a saving in the size of programs. This instruction or a suitable subroutine may also be used to set a block to a single value.

In my own special interest one program run may typically relocate around 200K bytes in total, hardly trivial.

On a more general basis how, without this facility, other than by pseudo subroutines or NOP's, does one insert program amendments that increase its size? NOP's are wasteful of bytes and one may still be caught.

It is also good practice to set unused memory, to zero or other suitable value, to minimise program corruption should a jump error be made. HALT is probably safest.

In many programs a block of variables needs an initial zero. This again is expensive if programmed in full.

One monitor facility usually omitted is *automatic relocation of branch addresses.* With the right, simple, programming technique it needs very few bytes to implement and may even be made completely transparent to the user. Time wasting amendments and catastrophic failures due to wrong branches are eliminated.

The cheapest implementation uses the National PACE. The routine needs a mere 20 bytes plus 2 or 4 bytes for each subroutine or label actually in the memory. The popular 8080 and 6800 need a slightly longer routine and 7, 8 or 9 bytes per item. Using this facility program lengths may be amended and subroutines inserted in any order without restriction. The simplest system permits a total of up to 256 items but little amendment is needed to permit unlimited numbers.

Extension of the routine will permit a program to automatically select and load only the sub routines it needs from a library of any size. For a large library a reel to reel tape machine may be switched on and left to load itself. Cassettes will do in most cases.

Machine code programming is often condemned as tedious and difficult. This routine permits programmes to be written in assembly language and then hand assembled by simple substitution of hex codes. Branch addresses will then be inserted automatically throughout the program. There is no need for tedious and error prone calculation.

Competent assemblers use vast chunks of expensive memory. High level languages do even worse either in software or less efficient object code. This last monitor routine permits easy operation with minimum wasted memory and the largest possible programs. Much of the monitor itself may be subroutines in the programs giving even greater efficiency.

This relocation routine wastes memory but only one or two hundred bytes even in a large program. The saving in time and torn hair justifies every byte.

R. G. Silson,
Near Station,
Tring,
Herts. HP23 5QX

CAPITAL OFFENCE

At the risk of sounding pedantic, I must point out an error in the geography tuition program by the Messrs. Lee: since 21 April 1960 the capital of Brazil has been Brasilia, not Rio de Janeiro, a correction which comes not from a more accurately-programmed computer but from my Brazilian wife!

At a less nitpicking level, the sub-routine for aligning decimal points by Sidney Leleux can be modified thus:

```
110 TAB(T-INT(LOG(ABS(N) ) /2. 3) ); N
```

This should be faster as it saves having to compute the value of LOG(10) every time, 2.3 being close enough for this purpose. The ABS function can be omitted when it is certain that N will always be positive.

Peter Rodwell,
Flat B,
50 Redcliffe Gardens,
London SW10 9HB

LENGTHENED PRINTOUT

Like G. D. Crompton (Letters October 78) I too have a copy of the C.B.M. Pricelist offering a printer for £459.00. This was acquired in April and *printers were expected to be* available June 1978 which I thought would give a couple of weeks practice on the PET first.

Unfortunately the latest date I have for the arrival of the printers is January 1979, which is rather frustrating! This is the fifth different date I have been given by the London Showroom.

I understand that there are compatible printers available, at twice the price, but, as a novice I am reluctant to have two suppliers each able to blame the other for any problems. Helpful suggestions from PET users or others would be appreciated.

J. M. M. Wilkie,
Weylode,
Horton,
Bristol BS17 6QY

UNDERSTANDING EACH OTHER

As you are aware, the standards bodies publish standards so that in any particular subject people can communicate in the same language.

In your excellent publication may I suggest you adhere to the preferred spelling of PROGRAM and the standard flowchart symbols. On page 32 of the October issue the input/output and terminal symbols are incorrect.

Paul Woolley,
Member of the BSI Computer Glossary Committee
Enfield D.P. Education Services,
25 Armfield Road,
Enfield,
Middlesex EN2 0DH

PCW *Contributors, please take this advice* PCW.

THE PDP–11

At the risk of being expelled from the Amateur Computer Club, I must correct Mike Lord's description of the PDP–11 range of processors. Most of the minis (as opposed to micros) that he describes are now not as widely used as the following:—

PDP–11/04 : Bottom of the range minicomputer, similar to PDP 11/05

PDP–11/34 : Very popular medium size computer, used widely for program development. Has extended instruction set and memory management unit for up to 124 k memory.

PDP–11/60 : High performance new addition to the range. Instead of the conventional rack cabinet, it is installed in two 12 - 5 megabyte RK 06 disk drives.

PDP–11/70 : The top of the range. Includes such items as cache memory and can address even more than 124k of memory.

However, I agree with Mike about the excellence of these computers; the instruction set is a model of power and simplicity. The software is also generally very good, although I would criticise Digital for ignoring Hexadecimal totally in favour of Octal, and not providing a disassembler with access to program symbol tables in their interactive machine language debug package.

Rupert Steele,
17 Lawrie Park Crescent,
London, S.E.26

Tid Bits PRODUCTS . . . . COMPANY NEWS . . .

First in Italy
Nearly a year old, **Homic** of Italy deals in personal computers such as the SWTPC 6800, the PET and the Nascom – 1. It distributes and sells magazines and books and has a software service. It has been instrumental in the creation of the Italian Microcomputer club, CICAP, which began activities last October. At the last big fair in Italy, SMALL, held in Milan, Homic scored a distinct success.
Further details from: Dr. Roberto Sanzone, Homic s.r.l., via Dante, 9 Milano. Telephone: Milano 809456.

U.K. Company, Midland Micronics Ltd., "First in World to produce a mini floppy disk system for Commodore Pet Microcomputer"
Midland Micronics Ltd., announce the Model M.M.3 mini floppy disk system for the Commodore PET. The disk system enables programs and data to be transferred to and from the Commodore PET personal computer nearly 1000 times faster than using the internal cassette storage system.

The twin mini floppy disk system is housed in an attractive metal saddle housing which fits onto PET's TV screen preserving the integrated package approach adopted by Commodore for the PET's original design.

A single drive system is also available in a stand alone cabinet. Whichever system is used connection to the PET is through the memory expansion socket; this does not prevent the PET memory being expanded. The system is supplied complete with a chip (PROM) which provides the additional instruction set for addressing the disk system from either a program or the PET keyboard.
Contact:
Mr. Peter New, Midland Micronics Ltd., Oakfield House, Station Road, Dorridge, Solihull, W. Midlands, B93 8HQ. Telephone Knowle (056 45) 6192 Telex: 336513.

The MM3 Mini Floppy Disk System for Commodore Pet.

New Terminal System
Southwest Technical Products announces what it claims is the ultimate terminal system, functional, flexible, reliable and affordable. The terminal, designated the CT-82, will work with almost any modern or computer system interfaced RS-232 serially from 50 to 38,400 baud.

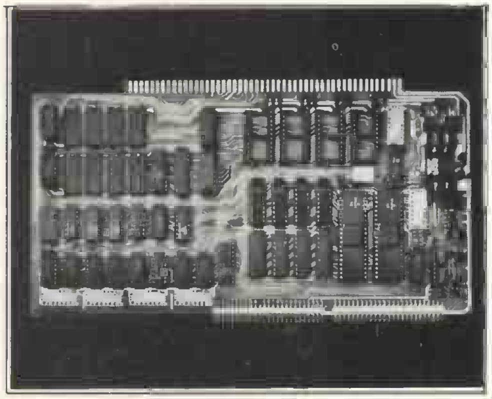
The CT-82 features over one hundred control functions operable from either the CT-82's keyboard or the computer's program. With its low price and graphics capability, it's ideal for business or hobby use.

The terminal design utilizes a Motorola 6802 microprocessor and 6845 CRT controller integrated into a modular system which is simple, reliable and easy to service.

The CT-82

The CT-82 is offered in assembled form only and sells for $795.00 ppd. in the Continental U.S.
Contact: Southwest Technical Products Corporation, 219 W. Rhapsody, San Antonio, Texas 78216. Telephone: (512) 344-0241.

New I/O Board
Thinker Toys announces the SWITCHBOARD™, an I/O board for S-100 systems. It has four parallel ports and two RS232/TTY serial ports plus strobe and attention ports. In addition, there are options for 4K of RAM and 4K of EROM.



Every port is switch programmable for flexibility in interfacing various types of peripherals. Each parallel port can be switched for input or latched output. Both serial ports can be switched to any of sixteen baud rates from 110 to 19K. Each strobe and attention port flip-flop can be switched for positive or negative pulsing. The eight I/O addresses of the SWITCHBOARD can be located on any boundary divisible by eight. Price is $199.00 kit and $259.00 assembled retail.
The design of this board meets the proposed IEEE (American) S-100 standard. Contact: Hilda Sendyk, (415) 524-2101.
Thinker Toys, 1201 10th St. Berkeley, CA94710.

The South Coast: first Microcomputer Systems Supplier.
Xitan Systems has opened a showroom and offices in central Southampton. With 5 different systems on display, Xitan Systems can demonstrate a wide range of equipment suited to all requirements and budgets.

Xitan Systems is an authorised distributor for **Commodore** and **Comart** microcomputer products. It also imports directly the Info 2000 floppy disc subsystem for S100 systems.

Specialising in floppy disc systems, Xitan can advise customers on the most suitable system for their requirements, offering unbiased advice, as many different manufacturers products are supplied. A catalogue is available on request.

Xitan Systems also offers two low cost evening courses for beginners to microcomputing. Topics covered include Basic programming, introduction to microcomputers, floppy discs, system design etc.
New address: 23 Cumberland Place, Southampton S01 2BB. Telephone: Southampton (0703) 38740.

First Edition
The first edition of *Computer Data Processing Equipment,* a new Financial Survey from Inter Company Comparisons Limited, is intended to fill a vital information gap on an important modern industry of future vast and increasing importance.

Profiling in its pages the fortunes of 174 companies operating in the field, the Survey reveals that 59% increased profits in twelve months, against 68% which upped turnover. 82% added to assets and 77% enlarged their liabilities, whilst 60% paid more to directors.

Closer examination of the Survey reveals a surprisingly uneven performance in the sector, with many companies showing startling changes in profits.

Detailing two years' turnover, total assets, current liabilities, profits before tax and payments to directors, the Survey costs £29.80 (VAT zero-rated).

There is an appendix which names 34 companies whose accounts were considered too old to be of use when the Survey was compiled (from the latest accounts filed at Companies House).

Copies of *Computer Data Processing Equipment,* 1st Edition, can be obtained from Mrs Hart at **Inter Company Comparisons Limited, 81 City Road, London EC1Y 1BD (01-253 9736).**

Keenest in the Business

Keen Computers of Nottingham are rapidly expanding their range of pre-recorded software for the Apple II computer.

The range includes a variety of mathematical and statistical packages i.e. Regression Analysis, Data testing, Matrix inversion, etc., and some commercial packages i.e. Addressing and mailing program and personal accounts. These programs are available on cassette (or disk) and come complete with a manual describing each program and a few notes on its usage. A comprehensive catalogue is available on request; also a descriptive pamphlet on the company itself.

Keen Computers are also offering a special package for accountants, based upon their Incomplete Record Accounting System, which includes all the hardware necessary (and Data 100 fast matrix printer) retailing at £4,500 exclusive of VAT.

Contact: Mr. A. Witterich, Keen Computers Ltd., 58 Castle Boulevard, Nottingham NG7 1FN. Telephone: (0602) 45865.

PCW *We have a little sample manual, "Shape-Create", from Keen Computers, and can recommend its lay-out and style.* PCW

A General Purpose Business Package for Microprocessors

Software Architects Limited are currently developing SALZOE (Software Architects Limited Zilog Drder Entry), a powerful business package, written in Zilog's ANSI 1974 Level 1 Business COBOL (containing many Level 2 features) and designed to be used on Zilog MCZ microcomputers under Zilog's RIO operating system. A minimum configuration is represented by 48K bytes core, dual floppy disc drives, one VDU and one printer.

The system has been designed on a transaction-processing basis with the minimum level of computing knowledge required from the user. A simple to use program menu, with a fully conversational mode of operation, leads even the first time user through the facilities.

Currently an invoicing module is live and provides full invoice extensions, credit and stock controls, product and account file maintenance.

On order, for delivery in December and to be exhibited at Compec, is a compatible sales ledger system including daybooks, statement production, aged debts. Also on order are purchase ledger and analysis modules for delivery early next year. Other modules in the planning phase are stock control, job costing, nominal ledger, etc.

The cost for a typical application, i.e. sales ledger with invoicing is of the order of £1,200, thus a complete turnkey solution costing around £8,000 is capable of handling some 2,000 products and accounts on-line at any one time. More disk capacity will increasc this figure for the larger organisation; SALZOE has also been trialled successfully on a hard disk.

Contact: Jeffrey Goldsmith, Software Architects Ltd., 34/35 Dean Street, London W1V 5AP. Telephone: 01 - 734 9402.

Professional Equipment for the Hobbyist's Budget

Especially designed to meet the requirements and budget of the hobbyist, here are two new professional, yet easy-to-operate portable oscilloscopes which guarantee high quality, high accuracy and high performance at very low prices:

British-made, the CALSCOPE Oscilloscope range consists of two instruments: the SUPER 10 Dual Trace and the SUPER 6 Single Trace Oscilloscope. Both instruments are fully transistorised, provide excellent performance parameters, and are always calibrated, with no variables to confuse the user.

Available from: MAPLIN ELECTRONIC SUPPLIES, P.O. Box 3 Rayleigh, Essex. – AUDIO ELECTRONICS, 301 Edgware Road, London, W.2. – MARSHALLS ELECTRONIC COMPONENTS, Kingsgate House, Kingsgate Place, London N.W.6.

SUPER 10 – In use at Marshalls, the Calscope Super 10 Dual Trace Oscilloscope can be applied for easy fault finding and checking.

Low Cost ASCII Printer

A low cost printer is announced by Kimberley Business Records giving correspondence quality output. This will allow the economical development of word processing applications on many home systems. A standard lever operated typewriter mechanism has been used, driven by 240V solenoids, aiming to minimise costs consistent with producing hard copy of a quality which is directly useful.

Designed for parallel data input with handshake control, ASCII coding is accepted for the 88 characters available, operating at a speed of 8 cps from a standard peripheral interface. It is supplied fully built and cased at £200 (including carriage and VAT). Alternatively, as a print mechanism only, requiring all power other than 240V, case, and TTL logic to be added, the cost is £160.

Full details from: Kimberley Business Records, 2 Hartington Road, Gosport, Hants PO12 3AG.

1K X 8 STATIC RAMS from MEMEC

The 8108 is a 1K x 8 Static N-MOS RAM available in a 22 pin pack, Features are single +5V rail, low power <60mW disabled, high speed – 300 nsecs. max access. The pinout has been chosen by JEDEC as a standard for a 22 pin 1K x 8 RAM.

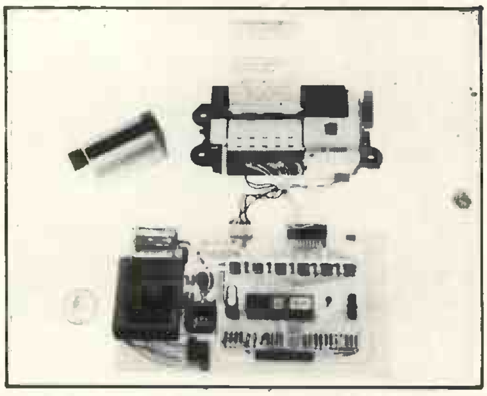
For parts, data and pricing contact:–

MEMEC (Memory & Electronic Components) Ltd., Thame Park Industrial Estate, Thame, Oxon, OX9 3RS. Telephone: 084 421 3146. Telex: 837508.

New Low-Cost, Alphanumeric, Ordinary-Paper, Tally-Roll Printer and One-Chip Microcomputer Interface

Datac Limited announce a new addition to their existing range of digital printers and floppy disc drives.

The PU1100 *printer mechanism* prints 20-columns of 7 x 5 dot matrix characters using ordinary 60mm roll paper and an ink ribbon cartridge. Printing rate is 100 1pm approximately and power input under 500 mA average @ 20v d.c. nominal.



The 310 interface, provides the complete interface, control, character generation, drive, and mains-input power supply for the PU1100 on one compact p.c. board. Data input is serial, V24/RS232C; or optionally 20 mA current loop, @ 1200 baud; or parallel TTL (UPI interface). Power input is 110-250V mains or 20V d.c. nominal.

The interface is based on a one-chip microcomputer and offers is a sub-system, when added to the PU1100, for both OEM's and End-Users. Both are available now, ex-stock.

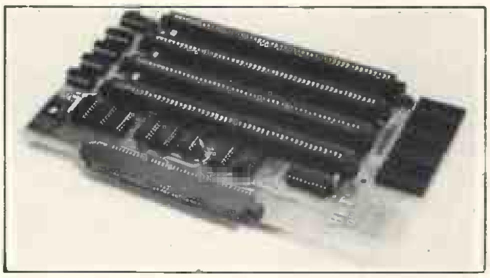
Contact for full details: M. J. Robinson, Datac Limited, Tudor Road, Altrincham, Cheshire WA14 5TN.

SUPER 6 – The Calscope Super 6 in use at Marshalls, triggering from a ringing signal.

SOLID STATE KEYBOARDS with A/D MOS/LSI encoding
Alphameric's new standard 60, 72 and 84 key designs minimise soldered and mechanical connections and eliminate costly PTH boards. Using Alphameric's own A/D MOS/LSI encoders to minimise component counts, these capacitive keyboards provide sit-proof strength, full N-key rollover, tactile feedback, and liquid spill resistance. Code outputs are standard ASCII or PROM-recoded for non-standard encoding. Total design service is also available, including serial, printer, video, telex and microprocessor interface capabilities.
Contact: Jim Denton, Director, Alphameric Keyboards Limited, Manor Way, Old Woking, Surrey GU22 9JX England. Telephone: (44) 4862 71555. Telex: 859131 ALPHAM G.

Keyboard technology from Alphameric.

"**BETSI**" – PET to S-100 Interface/Motherboard. A single circuit board which attaches directly to PET's memory expansion connector and provides both interface logic and four S-100 slots. Operates from any S-100 power supply and is independent of PET's parallel or IEEE ports. Available immediately ex-stock from: **Forethought Products, 87070 Dukhobar Road, Eugene, OR 97401.** Phone (503) 485 8575; with full documentation. Price: $119 Kit; $165 assembled.



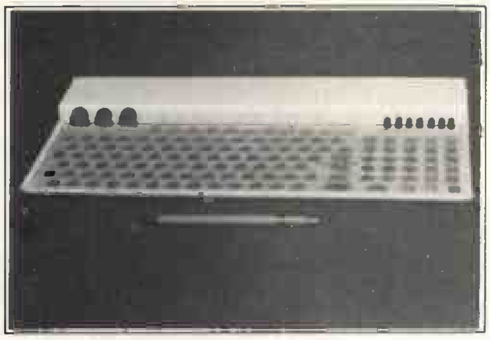
Another new company
Beta – System Process Analysis and Control Engineering Limited (S.P.A.C.E.) is a new company currently based in Bedford.

The company has long-term goals in the cybernetics (robotics) field based on research work completed by two honours graduates over the past five years.

The company currently hires out its "SERIES II INTEL M.D.S. 220 MICROPROCESSOR DEVELOPMENT SYSTEM" at a competitive rate (£100 per week on the customers' own premises).

The company plans to progressively introduce new services and technology in a controlled manner over the next two years.
Contact: L. Hard, 29 Chaucer Road, Bedford. Telephone: (0234) 41685.

A complete Stand Alone Data Input Terminal for the Microprocessor.
The full range of features can be enhanced by optional extras such as internal baud rate generator, parity check bit (bit 8) and earphone socket plus plug in personal earphone.
Available from **Star Devices (QWERTY) Ltd., P.O. Box 21, Newbury, Berkshire,** at a price of £37.50 incl. VAT, p & p.
Contact: P. D. Stubley.



TRUMPET, VOLUNTARY
Responding to TIDBITS information or an advertisement?
Say you saw it in **PCW**
Blow your own trumpet. Mention your magazine.

CONCEPTS OF PROGRAMMING

Barry G. Woolland

Commencing with a discussion of communication systems leading to definitions of language and digital codes, this article continues by outlining the principles of Flowcharting with several everyday examples illustrating the use of symbols and the techniques of flowcharting.

Further Software concepts are discussed, with descriptions of the three levels of programming: machine code, assembler and high level language.

Introduction

Man can communicate with the aid of his five senses, as illustrated by handicapped people who have lost one or more of their senses but are nevertheless highly sensitive with the remaining ones.

Assuming that one person wishes to communicate with another through the sense of hearing and the use of speech, it is clear that there must be some general agreement concerning how a spoken sound will be interpreted by the person who hears it. Over the centuries, different regions around the world have each developed their own ideas with regard to the meaning of specific sounds and their transcription onto paper. We call these ideas a language or even a *foreign language.* Although many different languages exist, only a modest number are in widespread use.

Communication can be defined as the imparting, conveying, or exchange of ideas, knowledge, information, etc. (whether by speech, writing, or signs), and is one of the most important activities of mankind. The capacity of major telecommunication links, as measured by a quantity known as *bits per second,* has paralleled the advances of civilisation, e.g. the capacity of such links has changed from 1 bit per second in 1840 to 50 000 000 bits per second in 1970.

A *language*, which can be defined as the whole body of words and of methods of combinations of words used by a nation, people, or race, is just one form of communication. Egyptian hieroglyphs, choreographic scores, mathematical symbols and equations, American Indian smoke signals, the sign language used by the deaf, and the Morse Code are other forms of communication used by man.

The 'information explosion' would have inundated mankind, had it not been for the use of *two-state coding* to represent all types of information such as the ten decimal numerals (0 through to 9), the twenty-six letters of the English alphabet (A through to Z), operations, symbols, motions, and the like. We call such two-state coding *on-off* or *Binary Coding.* Binary coding can be represented by any type of two-state device, e.g. an on or off light, an open or closed switch, a punched or non-punched computer card, a north or south magnetised magnetic core or region of magnetic tape or disc, two different voltage levels, two different current levels, or two different frequencies, or the abstract symbols 0 (off) and 1 (on). The importance of binary coding lies in the fact that it is possible to construct devices that will change state very quickly e.g. as fast as 5 ns. Such a device could, in principle, manipulate, transmit, or receive information at the rate of 200 million bits per second. Thirty-two such devices, operating simultaneously, could manipulate 6.4 billion bits per second. This is the basic capability that has allowed tremendous quantities of information to be stored, manipulated and communicated.

A *Digital Code* is defined as a system of symbols that represent data values and make up a special language that a digital circuit or computer can understand. Digital codes may be considered to be the digital 'languages' that permit information to be stored, manipulated and communicated. The many codes available may be subdivided as follows:

(a) Codes used by electronic circuitry to perform various digital operations, e.g. binary code.
(b) Codes used to convert the decimal numbers (0 – 9) into digital form, e.g. binary code, BCD, Gray code.
(c) Codes used to convert decimal numbers, the 26 letter English alphabet, symbols and operations into digital form, e.g. ASCII code.
(d) Instruction codes used by large computers, minicomputers, and microcomputers that cause the computer to perform a prescribed sequence of operations, e.g. IBM 370 instruction code, PDP 8/E instruction code, 8080 instruction code.

Principles of Flowcharting.

Any problem which the computer is required to solve must be capable of being *written down as a solution* in a series of clearly defined steps – known as an *algorithm.* The algorithm for the solution of a problem by a machine is the specification of a finite number of instructions which, when executed by the machine, determines the actual solution (if, in fact, a solution is possible). This list of instructions may be represented by a diagram of interconnected symbols – known as a *Flowchart.*

The *Program Flowchart* is a detailed description of the *program* to be used to solve a particular problem, and will invariably reflect the type of computer and the *language* to be used. There are many advantages in preparing a program in this way:

(a) it forces you to analyse the problem before you attempt to produce a solution.
(b) a clear description of how the problem is to be solved is presented.
(c) a record is provided which simplifies the task of finding errors in your solution.
(d) it can be used to describe to other people what has been done.

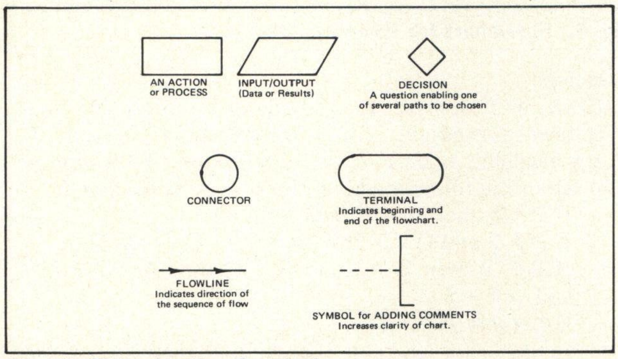


Fig. 1. Flowcharting Symbols.

The Simple Flowchart.
When a flowchart is being drawn, it is advisable to ask the following questions:

(a) What data is available? In what form is the data presented? In what units is the data measured? In what order is the data?
(b) What solutions are required? In what form are the solutions required? In what units are the solutions to be measured? In what order are the solutions required?
(c) What methods are available for the solution of this problem? Which of these methods is the most efficient?

The symbols most widely used in drawing flowcharts are shown in Fig. 1, together with an explanation of their meaning.

We shall now consider some simple examples to illustrate how these symbols are used in drawing flowcharts.

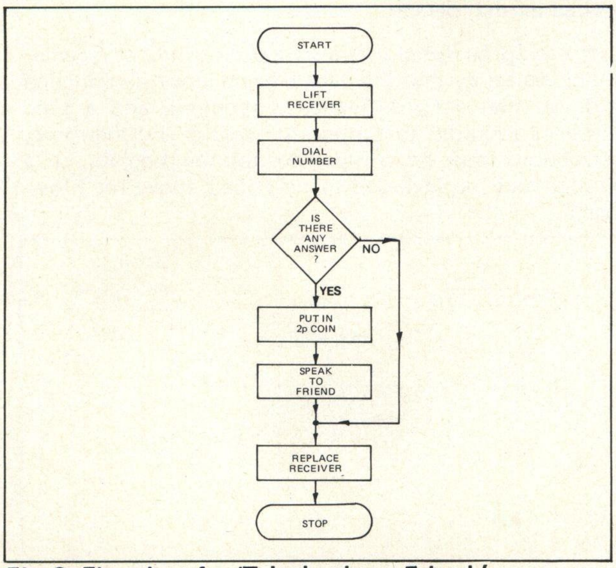


Fig. 2. Flowchart for 'Telephoning a Friend.'

Example 1.
Draw a flow chart for the everyday example of 'Telephoning a Friend', assuming that the call is to be made from a telephone kiosk, and that only one operation can be performed at a time. The solution is shown in Fig. 2.

Example 2.
Draw a flowchart for 'Getting Up in the Morning'.

A solution is shown in Fig. 3, and it should be noted that there are many possible variations. The important points to appreciate are that *you must not take things for granted, i.e. NEVER ASSUME.*

4. **Arithmetic Symbols.**
Although the two flowcharts considered above are for simple everyday tasks, many problems exist in which we are mainly concerned with arithmetic operations. The numbers specified in program flowcharts are generally referred to as *variables* – since their value can change. These variables — as in algebra — may be denoted by letters, or groups of letters, e.g. x, y, a, b, A, N, NUM, ANS, etc. Each variable occupies a storage location in the computer memory and is assigned values during the sequence of instructions specified by the flowchart. The *assignment* is usually denoted by one of the following symbols:

$$= \qquad := \qquad \leftarrow$$

For example, the algebraic statement P = Q is used to denote the assignment of a value of the contents of store location Q to the store location P, *OR* location P takes the value of the contents of store location Q *OR* $P \leftarrow Q$ is an alternative method of showing the same thing.

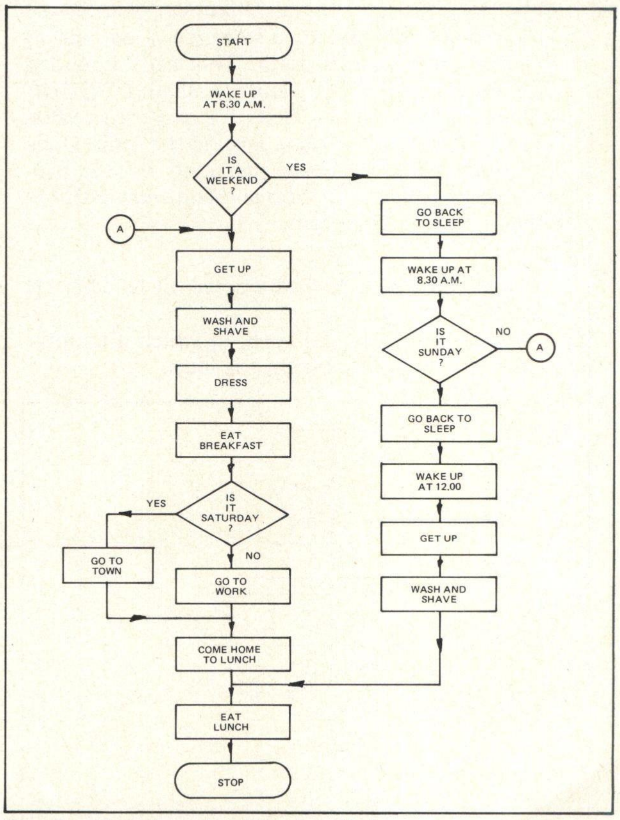


Fig. 3. Flowchart for 'Getting-Up'

5. **Arithmetic Statements**
When arithmetic operations involve the use of variables, we build up the arithmetic statement by writing:

Variable = Arithmetic Expression.

where the arithmetic expression is made up of combinations of variables and standard arithmetic operations, E.g.

$y = y + 1, x = y + z/a, x = (-b + \sqrt{[b^2 - 4ac]})/2.a.$

The complexity of the statement allowed is generally determined by the programming *language* which is to be used.

Decisions are all effected by comparisons, logical operations or arithmetic relationships. Some variations of the basic decision symbol are shown in Fig. 4.

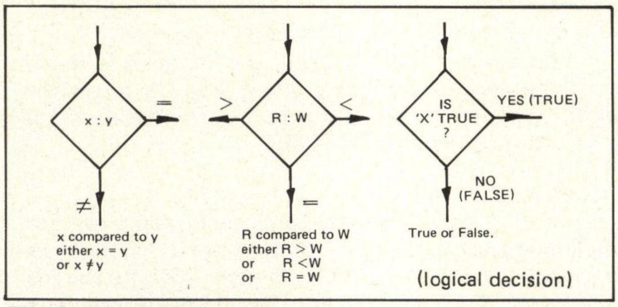


Fig. 4. Decision Symbols.

6. Looping

Looping is a process which enables the repeated use of a section of program. However, when preparing a program in which it is desirable to use looping techniques, it is essential to ensure that we can ultimately *'get out of the loop'*. This can be achieved in two ways:

(a) When we know how much data is being processed, i.e. when we know how many times we have to 'go round the loop', we can include a *counter* which is incremented by one each time we go round the loop. When the counter reaches the predetermined number, we exit from the loop to complete the remainder of the program.

(b) When we are processing an unknown quantity of data, we can add one data item (i.e. a data card) after all the program data. This card is coded with, say –1 or ****, and is called a *ROGUE VALUE*. The program looks for this rogue value every time it reads in data around the loop. Once the rogue value has been detected by the program we exit from the loop and continue with the remainder of the program.

Example 3.

Draw a flowchart to raise Y to the power of N, where N is an integer.

The solution is shown in Fig. 5, in which a counter is used. When M = N, we exit from the loop.

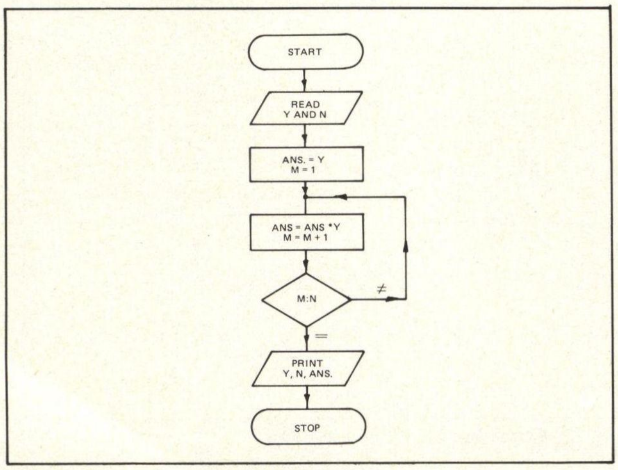


Fig. 5. Flowchart for Example 3.

Example 4.

Draw a flowchart to solve the problem: given two *different* numbers A and B form a number C which is the sum of the largest squared plus the other.

Assume that all the numbers are positive, and that there is an unknown amount of data.

The solution is shown in Fig. 6, in which a rogue value of – 1 is used.

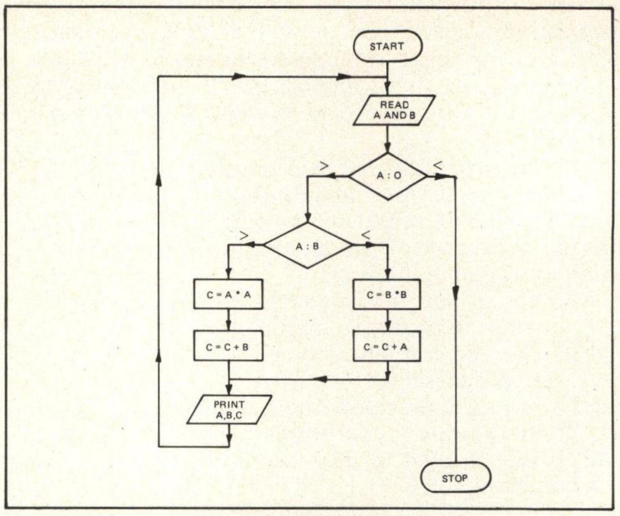


Fig. 6. Flowchart for Example 4.

Exercises.

1. Draw a flowchart to provide a solution for the following problem: Given that P, Q, R, S, and T are available as data input, it is required to compute A when the following conditions must be observed:

   If $P = 2$, then $A = P^2 + Q + R - S$

   If $P \neq 2$ and $Q = 3$, then $A = T^2 + Q + R - S$

   If $P \neq 2$ then $Q \neq 3$ and $R = 4$, then $A = (P + T)^2 - Q - R + S$

   Otherwise $A = +3$

   In this case, we will assume that there is *one* set of data only.

2. Draw up a flowchart suitable for the task of sorting English decimal coins into separate bags.

7. A Simple Program

A *program* consists of a series of precise instructions to the machine. These instructions are loaded into consecutive 'pigeon holes' called *addresses* (or locations for words or bytes) in the computer store. The sequence control register scans these instructions in order and causes the computer to obey them. Arithmetical operations are carried out in the arithmetic unit, the results of operations appearing in a special register known as the accumulator. In general, movement of data in the computer, and to and from the computer, takes place through the accumulator.

A simple 'popular' example for the calculation of wages is shown in Fig. 7, in which it is assumed that each person's data is coded on to two cards, the first is the Rate for the job and the second is the number of Hours worked by that person.

It should be noted that instructions would also generally be necessary that the data is being input via punched cards so that a Card Reader is required, and a Line Printer is required to output the results. Furthermore, instructions must be included to stop the program, i.e. a counter may be included as described above for Flowcharting.

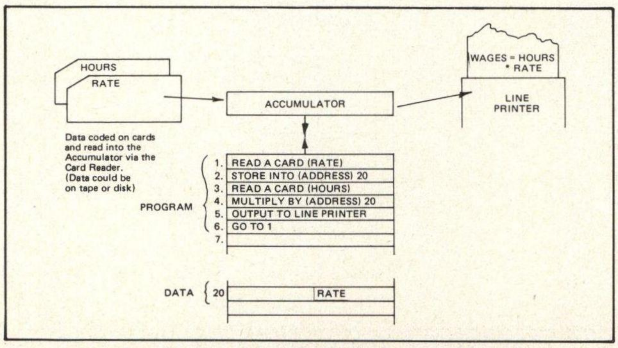


Fig. 7. A Simple Program 'Calculation of Wages'

8. Concepts of Software.

In any computer system, the software provides the interface between the human operator and the machine, and must ultimately result in a sequence of instructions being produced in a form which is acceptable to the machine.

Various *'levels'* of programming have now been established by one or more stages between the programmer and the machine acceptable code, these levels are; Machine Code, Assembly Language and High Level Language.

A program written in *Machine Code* consists of a list of instructions in binary form to be loaded into the computer memory for the computer to obey directly. It is therefore necessary to specify the number of the address of each word (or byte) in memory whether it is instructions or data.

A typical instruction is 'add the contents of store location 50_{10} (binary 110010) to the contents of the accumulator, leaving the contents of store location 50_{10} unchanged:

In this case, the *operation* to be performed is ADD, and the *address* of the data to be operated on is 50_{10} or 110010_2. Assume that the code for ADD is 01, and assume that our word (or byte) length is 8 bits, then this instruction will appear in store as:

Operator	Address
01	11 00 10

Example 5.

Write a program using the simple machine code listed below to solve the problem:

$$Q = P.U + \frac{Q.V}{R} - S.W$$

Machine Code Key:

Operation Code No.	Mnemonic	Command	Meaning
01	CAD	Clear and Add.	Clear arithmetic section and add store location ----- to accumulator.
02	ADD	Add.	Add store location ----- to accumulator.
03	STR	Store.	Store accumulator in store location -------- .
04	SUB	Subtract.	Subtract store location ----- from accumulator.
05	MUL	Multiply.	Multiply accumulator by store location --------.
06	DIV	Divide.	Divide accumulator by store location -------- .
07	PRT	Print.	Print out accumulator.
08	START	Start Computer	Start computer, get the address of the first instruction word from the operation address.
09	STOP	Stop Computer	Stop the computer immediately.

Assume that the values of the variables are in store locations as listed:

P is in the store location 100
Q " " " " " 101
R " " " " " 102
S " " " " " 103
U " " " " " 200
V " " " " " 201
W " " " " " 202

Assuming that the instruction word length is 8 bits then the program (with explanation) is as shown below:

Mnemonic	Instruction Address	Instruction	Content of Accumulator
START	000	08000*001*	O
CAD	001	01000*200*	U
MUL	002	05000*100*	P.U
STR	003	03000*300*	P.U
CAD	004	01000*201*	V
MUL	005	05000*101*	Q.V
DIV	006	06000*102*	Q.V/R
STR	007	03000*301*	Q.V/R
CAD	008	01000*202*	W
MUL	009	05000*103*	S.W
STR	010	03000*302*	S.W
CAD	011	01000*300*	P.U
ADD	012	02000*301*	P.U + Q.V/R
SUB	013	04000*302*	P.U. + Q.V/R – S.W
PRT	014	07000*000*	P.U. + Q.V/R – S.W
STOP	015	09000*000*	

The process of writing machine code programs is obviously a laborious one, and is somewhat difficult. Furthermore, this type of program is time consuming and difficult to modify. However, machine code can be considered as being most appropriate to small dedicated systems, e.g. a microcomputer can be used to directly control an industrial process, in which the program instructions (software) can be stored in ROM, PROM, or EPROM and the data on which the computer acts will be measurement data made within the process and converted from analogue to digital form and fed directly into the microcomputer. This provides a relatively inexpensive system, since no additional peripherals are required for the computer.

An *Assembler* is a special program which allows instructions to be written in the form ADD 50 or SUB TAX to be automatically translated into machine code, generally with one written mnemonic instruction corresponding to one machine instruction. Although programs may take a long time to write using these 'low level' languages, they usually result in very efficient programs in terms of store used and execution times.

A list of typical basic instructions for a Mnemonic Assembly Language is given below, in which:

(A) means the *contents* of the Accumulator.
(n) " " " " " store location n.
N denotes an integer N (assumed to be positive and within the range 0 to 999).

Instruction	Operation.	Comments
LDA n	(n) → A	(n) unchanged
STA n	(A) → n	(A) unchanged
ADD n	(A) + (n) → A	(n) unchanged
SUB n	(A) – (n) → A	" "
MLT n	(A) * (n) → A	" "
DIV n	(A) / (n) → A	" "
LDAN	N → A	This range of instructions deals directly with positive integers and NOT with store locations.
ADDN	(A) + N → A	
SUBN	(A) – N → A	
MLTN	(A) * N → A	
DIVN	(A) / N → A	

Example 6.

Using the above instructions, write the group of instructions necessary to perform the following:

Assuming x is stored in location 12, compute (x + 3). 40 and store the result in location 12.

Solution:

LDA 12	*Put content of 12 (= x) into A*
ADDN 3	*Add 3 to content of A*
MLTN 40	*Mult. content of A by 40*
STA 12	*Store content of A into loc. 12.*

Example 7.

Write the instructions to perform the following operations:

Add the two numbers stored in locations 25 and 26, store the result in location 25 and zeroise location 26.

Solution:

LDA 25	*Load A with contents of loc. 25*
ADD 26	*Add content of loc. 26 to A.*
STA 25	*Store content of A in loc. 25*
LDAN 0	*Put zero into A*
STA 26	*Put content of A into loc. 26.*

Most Assembly Languages have many more instructions which include versatile functions such as 'Jump', 'Modification' and 'Function' instructions which allow more complex operations to be performed in response to relatively simple written (mnemonic) instructions. One commonly used assembly language is the *ASCII* code (American Standard Code for Information Interface), which is commonly referred to as 'ASK-EE'.

High Level Languages are completely independent of the machine, relatively easy to learn and allow the programmer to concentrate on the problem to be programmed. There are two types of high level language, the *interpretive* such as *BASIC* (Beginner's ALL-purpose Symbolic Instruction Code), which is often referred to as a 'conversational' language since the form of instructions and statements are more humanly biased. This type of language is translated into machine code by means of an *interpreter*. The second type of high level language is converted into assembly language by means of a *compiler* before final translation of assembly into machine code. The compiler is a program containing a list of statements used in the problem-oriented language, and for each statement a list of machine instructions necessary to perform that statement. Then, by running the source (problem-oriented) program with the compiler program, an object (machine code) program is produced, which is then used to process the data. During compilation, as each source program instruction is read in, the compiler scans it for errors in the construction of the statements, and gives instructions for these errors to be printed out — the error messages being termed *diagnostics*, which is of great assistance in *debugging* (finding faults in the program and correcting them). Logical errors will not be revealed in this process, only the errors which the machine cannot recognise, i.e. the *form* of the instructions.

9. Preparing the Program

We have briefly examined the principles of Flowcharting, enabling a complete breakdown of the problem, and which can now be used in writing the program to be presented to the machine. When relatively simple tasks are being performed using machine code, the program can be 'written in', one instruction at a time, by setting switches on the front panel. Although this may be useful in a microcomputer prototyping development system, it becomes laborious for anything but the simplest program and is therefore prohibitive.

Programs are therefore hand written onto program data coding sheets, and then punched on paper tape or cards or stored on cassette or disk for subsequent reading in by the appropriate peripheral, or the program can be 'written in' using a teletypewriter (TTY), a keyboard display unit (KDU), or a visual display unit (VDU).

The VDU provides very useful 'Terminal' equipment, allowing the operator to 'write in' to the computer via a keyboard (similar to that of a typewriter) and giving a 'read-out' on the screen (similar to that of a television), so that programs can be written and solved very quickly, the main disadvantage being that a hard copy of the program is not produced.

Chris Ward

Personal Experience

Inevitability is a hard thing to contemplate, especially when it is seen as threatening long established order. But unwillingness to accept the inevitability of technological change does nothing to diminish its steadily engulfing progress.

So it is that such terms as VDU, hardware, software, program, and the like can be heard in the editorial halls of newspaper offices, which are without doubt among the most technologically conservative of places. Not that electronic newsrooms are with us yet — except at the *Nottingham Evening Post* — but they are being seriously considered and actively discussed. Because these are still the pioneering days. The result is that newspapermen are looking at equipment, being shown its capabilities, and struggling to grasp a whole new vocabulary which is alien to them. They may soon learn to cope with such terms as VDU, and concepts such as direct editorial input, but talk to them about a floppy disc and they face a strong temptation to follow the example of Pontius Pilate.

The only knowledge that a journalist needs to be able to operate in an electronic newsroom is an understanding of the concept and the ability to communicate, in very simple terms, with the computer using the keyboard and screen on his desk. But there are those who want to know more, just as there are those who sit back unaware of, or choosing to ignore, the inevitability of technology.

There is much to be done, for the journalist as the eventual user should at least be aware of design capabilities so that he has some ability to influence the choice of the right combination of hardware and software to fit the job.

Advice from the experts might well be that when you have reached that stage you should stop while you are still ahead. Sound as that advice may seem it should not preclude getting to know what makes computers work, or, in some cases, not work. While it may be quite sufficient to know that "the computer is not working", it is surely better to understand a little about the reasons why. Even in these days prior to direct editorial input and screen editing computers are no strangers to the many newspaper offices where phototypesetting is in use or being introduced.

Maybe a greater understanding will come because of the domestic use of computers. Having seen teletext systems such as ORACLE and CEEFAX being demonstrated, and having been particularly impressed and amazed by the potential of the Post Office's Prestel system, I am left in little doubt that the widespread use of domestic computers cannot be far away.

To one who once took part of an A level sciences course such thoughts have the disquieting effect of bringing back misty memories of those academic days, and with them a desire to know a little more, and perhaps remember a little more.

It was with this desire recently freshened by a colleague's departure to sit in front of a VDU that last February I was thumbing through a newsagent's shelves and discovered a copy of the first issue of PCW. I suppose that it was the sight of a Qwerty keyboard and screen on the front cover that clinched the deal, for a quick flick through the pages presented what at first sight appeared to be a daunting challenge rather than a "good read".

Still there was a promise of articles for beginners so I started to work from cover to cover with the reading punctuated with many breaks as my brain tried to assimilate all the information pouring from the pages. As a raw novice it was clear that my distinct disadvantage was a lack of personal software — I had no program to cope with much of the language.

The mists of the past cleared somewhat at the sight of an article on binary numbers and enabled the completion of examples on binary addition, encouraging further exploration.

AND, NOT and OR gates seemed vaguely familiar and the principle of the half adder was not too difficult to grasp. But then came some fateful words; looking quite innocent on the page, but throwing down the gauntlet as plainly as any knight of old.

"You may care, in the meantime, to try your hand at the design of a full bit adder network . . . ."

After several hours in which logic played no part and with six foolscap sheets covered with masses of figures and diagrams it finally dawned that there had to be a systematic or logical way of doing the thing. It also became clear that it was time that I started working in that way or face either defeat or a sleepless night.

Was there not some strange pattern in those countless lists of numbers? Was it not possible to predict the effects of certain combinations of the three components without working out all of them? Some form of logic, although far from perfect, was emerging and it took just one more diagram and one input/output table to clinch it. This was followed by a period of disbelief which was dispelled only by working inversely to try to come up with more logical reasons for what I had done.

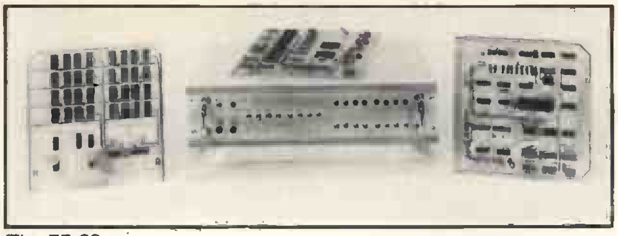
Although I could well be accused of masochism, those hours were, on reflection, most enjoyable, especially when I finally mastered the application of simple logic and produced a solution, elementary as that might be for anybody for whom mathematics have always held a fascination.

So, after a well deserved rest, it was off again through articles dealing with the use of flowcharts, to other features and then on to a section dealing with models.

There was something about the sight of $\frac{dw}{dt}$ in an equation that brought forth a shudder of recognition. On closer inspection the equations did not seem too difficult to understand and when I went on to find the steady state voltage, V_{ss}, in a resistor/capacitor circuit I was pleasantly surprised to come up with a not unfamiliar formula remembered from physics lab days.

However interesting the insights that may be gained from feeding a sinusoidal current into such a circuit, I decided to leave the seeming challenge strictly alone, at least for the time being.

What was needed, I decided, was an appraisal of my existing personal software and an assessment of what would be needed to bring it up to an acceptable standard and beyond so that I could derive more pleasure from the world of personal computers. I was, it would seem, hooked.



The 77-68

Ron Williams

THESE DASHED DOTS.... THOSE DOTTY DASHES___

RANDOM MORSE CODE GENERATION USING A 77–68 MICROCOMPUTER

An article has recently been published (Radio Communication, January 1978) for a hardware design of a morse code practice generator, but it is thought that there are many short wave listeners and radio amateurs who like myself are also interested in micro-computers and own or have access to a suitable machine.

The program to be listed in written for a MC6800 based computer, specifically the 77–68; further details can be obtained from Newbear of 7 Bone Lane, Newbury. It generates five letter groups at speeds of 12, 14, 16 or 18 words per minute by changing the content of various locations as listed in the program.

No attempt was made to generate numerals as they contain five elements each (i.e. 5 dots or dashes) in contrast to up to four elements per letter. The maximum of four elements is conveniently stored in an eight bit word – two bits per element. Therefore 11 represents a dash, 10 represents a dot and 01 and 00 are ignored and used to skip over elements not required. The characters are stored in hexadecimal notation; representing for 'A' say the word BO in Hex, is 10110000 in Binary – 10 being the dot and 11 being the dash, the remainder being skipped.

The output port is assumed to be at location FF and is simply a loudspeaker connected via a step-down transformer to yield an impedance of about 1000 ohms.

The program has been in continuous use now for two months and certain limitations are now apparent in the random number generation, but they are not significant enough to cause a real problem.

All the numerals given in the program are in hexadecimal notation not decimal.

I believe the program is a very useful application of a basic 77–68 micro-computer and hope it will give others the same pleasure that I have enjoyed. Let's hope I meet up with some of the readers in the C.W. section of the bands sometime.

Address	Machine Code	Label	Operator & Operand	Comments
00	8E00F0		LDS #00F0	
03	8601		LDAA #01	
05	9777		STAA RND	Initialize random number
07	7F0078		CLR RND+1	
0A	9677	CRSL	LDAA RND	Begin character selection
0C	48		ASLA	By generating random
0D	2402		BCC STRND	Number
0F	8863		EORA #63	
11	9777	STRND	STAA RND	
13	841F		ANDA #IF	
15	8119		CMPA #19	Check for RND NO.
17	2EF1		BGT CRSL	Less than 27
19	9779		STAA RND +2	
1B	DE78		LDX RND +1	
1D	A67B		LDAA CRCTR,X	Get character
1F	D67A		LDAB LTR CNT	
21	5C		INCB	
22	D77A		STAB LTR CNT	
24	C105		CMPB #05	Check for 5 characters
26	2D09		BLT LTRSUP	To a word
28	5F		CLRB	
29	D77A		STAB LTR CNT	14wpm 16wpm 18wpm
2B	CE6920		LDX # 6920	5A1B 4ED8 4615
2E	09	WOLOOP	DEX	Word space generation
2F	26FD		BNE WOLOOP	
31	CE3490	LTRSUP	LDX # 3490	2DOE 276B 230A
34	09	LTRLOOP	DEX	Letter space generation
35	26FD		BNE LTR LOOP	
37	8C0004	PSHL	CPX #0004	
3A	2709		BEQ SUSL	
3C	08		INX	
3D	16		TAB	Shift word left to get
3E	44		LSRA	at the two bits required
3F	44		LSRA	for next element
40	C403		AND B #03	
42	37		PSH B	
43	20F2		BRA PSHL	
45	4F	SUSL	CLRA	
46	8104	ELSL	CMPA #04	Check for maximum of
48	27C0		BEQ CRSL	four elements
4A	4C		INCA	
4B	33		PULB	
4C	C103		CMPB #03	Select whether dot or
4E	2706		BEQ DASH	dash wanted
50	C102		CMPB #02	
52	2708		BEQ DOT	
54	20F0		BRA ELSL	14wpm 16wpm 18wpm
56	CE2B80	DASH	LDX #2B80	2549 20A0 1D00
59	36		PSHA	Dash set up
5A	2004		BRA TONE	
5C	CE0E80	DOT	LDX #0E80	0C6E 0AE0 09AB
5F	36		PSHA	Dot set up
60	C628	TONE	LDAB #28	
62	09	LOOP	DEX	
63	2708		BEQ SPACE	Tone Generation
65	5A		DECB	
66	26FA		BNE LOOP	
68	4C		INCA	
69	97FF		STAA FF	
6B	20F3		BRA TONE	
6D	CE1A48	SPACE	LDX #1A48	1687 13B6 1185
70	09	SPLOOP	DEX	
71	26FD		BNE SPLOOP	Inter element
73	32		PULA	Space generation
74	20D0		BRA ELSL	
76	00		BUFFER SPACE	
77	000000	RND	RMB 03	
7A	00	LTRCNT	RMB 01	
7B	BOEAEE	CRCTR	A, B, C	Character List
7E	E880AE		D, E, F	
81	F8AAA0		G, H, I	
84	BFEGBA		J, K, L	
87	FOEOFC		M, N, O	
8A	BEFBB8		P, Q, R	
8D	A8C0AC		S, T, U	
90	ABBCEB		V, W, X	
93	EFFA		Y, Z	

PCW The 77–68 was featured in Vol 1, No. 1 PCW

PLAY ON

Computer Music ~ Some Thoughts on Data Storage

Paul M. Jessop

The applications of computers to music are legion but the amateur computer user is likely to be most interested in the generation of simple tunes, whether of his own composition or otherwise. A brief rendition of a well-known piece makes an excellent demonstration to visitors and can convince them that the computer is of some practical use. The same task is also of course very gratifying to the programmer.

The hardware required for the production of music can vary from the absurdly complicated to the ridiculously simple. Into the former category would fall the use of a microcomputer to control a fully-fledged synthesiser, which is very nice but beyond the scope and finances of most of us. The opposite end of the spectrum is represented by a speaker connected to a single bit output port. Whatever hardware technique is used however, there remains the problem of *storing* the tune in the memory of the computer, and the attendant difficulty of putting the data there in the first place, all without using vast quantities of memory space in the process.

The easy way

One very easy way of storing the music is to use two consecutive bytes per note. The first contains a value which can be used directly by a timing loop to define the *frequency,* and the second byte contains a similar value, again used by a timing loop to determine the *duration* of the note. This system clearly uses a minimum of software to support it but it does have a couple of drawbacks. The first is the limited range of frequencies which can be produced because the higher the pitch of the note, the lower the value of the pitch byte and consequently, the lower the accuracy of the note. Only about two octaves are really available and this limits the scope of the system rather severely. Another disadvantage is that the memory is used rather wastefully since eight bits (which can represent 256 different values) are being used to store a variable which can only take about 24 values (or 16 if semitones are not counted). Also, since the duration is stored in absolute terms, only a small proportion of the possible bit patterns will be used.

Despite these difficulties however, the technique can have its uses in short tunes or in music which is outside the scope of conventional western scales and rhythms. Where memory space is at a premium, it becomes practical to use more complicated software to economise on the memory used by the data. On a piano keyboard, there are 88 discrete notes; so to cover all of these would require seven bits, which is very little improvement over the previous method. However, if a musical score is examined, it will be seen that a tune largely stays within the same octave for most of the time. This means that only 13 different notes are required at any one time. Neglecting for the moment the case where a tune leaves its octave, it is clear that all the wanted notes can be represented by four bits, which can in fact take 15 different values (not 16 for reasons which will become clear later); so an extra note can be added at one end, and a rest defined.

This leaves the duration to be dealt with. Fortunately, the system of note durations used by musicians is already a binary coded one so the task is made much easier. If the longest note is a minim, then the other durations are successively one half of the duration of the previous one. Thus one minim is equal to two crotchets, four quavers or eight semi-quavers. Any note can be made half as long again by putting a dot after it. These principles are summarised in figure 1. The duration can easily be coded into four bits by simply coding the length of the note in terms of semi-quavers. Because of the binary nature of the system, the effect is that each bit of the code represents one of the basic note-lengths and these are then added together to produce the final duration.

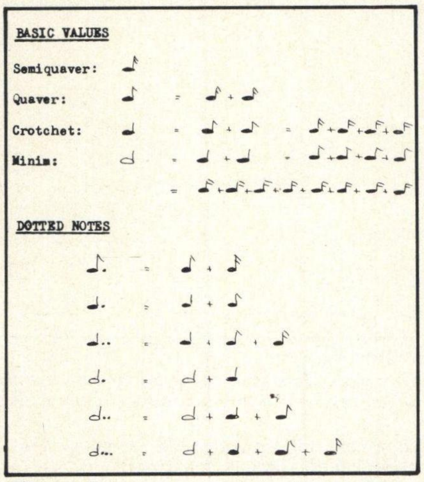


Figure 1

The theory of note durations. See text for explanation.

To turn now to a problem which was glossed over before, it is clear that the notes of a tune will not always fall in one octave, even if they tend to hover around a common octave most of the time. To overcome this, a series of "octave shift codes" are defined. These all have a zero in all the high four bits, where the pitch is normally stored, which explains why only 15 notes are available, not 16 as mentioned earlier. These cause the tune to shift into the specified octave and to stay there until another shift is executed. In addition, one of these codes is defined as a "stop" code which could either cause the program to stop altogether or to return to the beginning of the tune and start all over again. The latter can however lead to early insanity if continued for protracted periods of time!

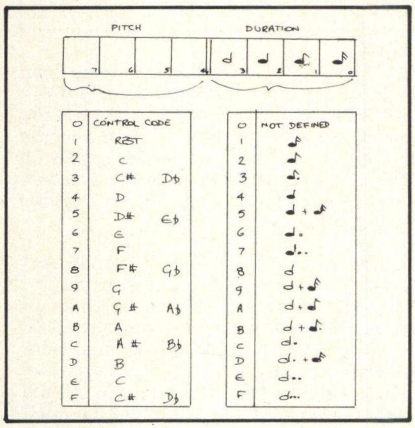


Figure 2.

The codes for normal notes. Reference should be made to both the text and figure 1 for explanation.

Summary of Codes

The codes for all the various functions are summarised in figures 2 & 3, and it will be noticed from figure 3 that not all of the available control codes are used for the octave shift function. These can be used in a variety of ways, as suits the user but a few interesting possibilities are: (1) switching external voicing circuitry to change the characteristics of the note produced, (2) generating synchronising pulses to, for instance, facilitate multi-track recording or (3) to trigger percussion generators. This latter is possibly the easiest to implement since the extra circuitry is very simple.

To generate a sound such as that produced by a resonant body like a drum or woodblock, a simple amplifier with tuned feedback can be adjusted so that it is just below the verge of oscillation. A pulse now applied to the input of the amplifier will trigger it into a state of oscillation which will decay in amplitude since there is no overall gain around the circuit. This produces the same type of sound as a percussion instrument and can be easily built using one gate of a CMOS package as shown in figure 4. This circuit has been around in one form or another for several years and seems to work reliably. The capacitor values need to be changed to vary the pitch of the instrument and generally, C1 and C4 need to be about three times as

0	0	0	0				
7	6	5	4	3	2	1	0

0	Stop
1	Octave shift to C′
2	C
3	c
4	c′
5	c″
6	c‴
7	c^{vi}
8	c^{v}
9	
A	
B	
C	Spare
D	
E	
F	

Figure 3.
The definitions of the control codes used in the system. The use of upper and lower case letters with superscript primes is the normal musical notation to show the octave in question. The codes annotated "Spare" can be decoded by the program and used for any desired purpose. Possible uses are discussed in the text.

large as C2 and C3, values of 0.15uF and 0.047uF being typical for a bass drum.

Of course, all the clever techniques in the world for data storage will not make music. Now we need two things, hard-ware to make the music audible, and software to pull the data from memory and tell the hardware what to do. These will be dealt with in this order.

There are two fundamentally different approaches to the generation of the tones which go together to make a tune. The first is to make use of an IC called a "top octave generator". This has twelve outputs which represent the twelve notes of the highest octave, from C to C, of a piano. By dividing them successively by two the corresponding notes of lower octaves can be produced. It is a simple matter to interface this arrangement to a microprocessor and a basic circuit block diagram is shown in figure 5. One of the tones produced by the chip is selected by a 12 to one multiplexer and this is then fed to a string of binary counters, the output of one of them being selected by another multiplexer, this feeding the output of the unit. The select inputs of the multiplexers are fed by a parallel output port of the microprocessor. If a spare input on one of the multiplexers is connected to ground, there will be no output if that input is selected, so the output can be muted.

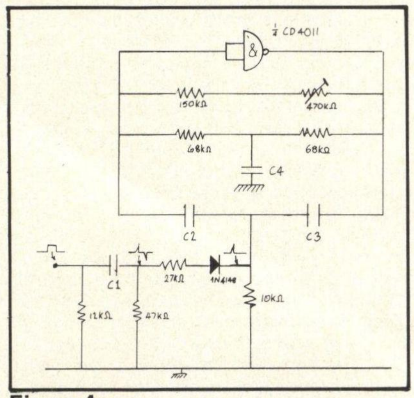


Figure 4.
A simple percussion generator for use with the computer music system described in the article.

Repugnance

To the software man however, the idea of using hardware to do what can easily be done by the processor itself is repugnant. It is not a difficult task to write a program which reads a data byte from memory and looks up in a *table* contained within the program, the value of the time which must be spent in a loop to correspond to the wanted note. This value is then multiplied by an amount which is dependent on the octave of the note which is desired. At the same time as the pitch of the note is being timed, a count is kept of the duration of the note so that the next note may be fetched at the appropriate time. This could lead to timing difficulties, and it is probably simpler to do the pitch timing by use of software timing loops; but to have an external clock which defines the semiquaver rate of the tune. This can be a 555 timer or a simple astable oscillator. This scheme has the advantage that the tempo of the piece can be easily altered, even during its course, without affecting the tune by interrupting the sequence or changing the pitch of the notes.

The easiest way to implement the clock is to use a self clearing input port such as one of the control ports on a 6820 PIA. This input is sensitive to an edge and sets a bit in the status register. However, this bit is cleared the first time that it is read so that the bit will no longer cause confusion. The use of the hardware clock also makes much easier the generation of more than one note at a time. The clock forces the two lines of music to remain in synchrony, provided that they have been put into memory correctly.

The general scheme of affairs from the hardware angle is shown in figure 6. Note that although only two output ports are shown, this can be extended by as much as is wanted, the limiting factor (apart from memory space) being the amount of time within the timing loop available to time all the notes separately, and of course, the mental agility of the programmer! Either of the two techniques described above will only produce one type of note, although the volumes of the two channels can be independently varied. This is a square wave, and when heard sounds thoroughly synthetic — not only because of the waveshape of the note but also because the envelope is rectangular and allows no variation in the attach and decay as a musical instrument would.

One way of giving at least a little voicing to the notes is shown in figure 7. It cannot hope to authentically imitate an instrument as could a proper synthesiser, but it will give the sound produced some character. An extra bit of output port is needed for each channel and this provides a trigger for an envelope generator, a standard synthesiser building block for which many designs are available. This, as its name suggests, generates a voltage corresponding to the envelope of the wanted note. This is then fed to a voltage controlled amplifier which superimposes the envelope onto the note, giving it its characteristic sound. If the envelope generator is made retriggerable, then each note's envelope will eclipse the last one's. Clearly, each separate tune will need experimentation with the attack and decay to achieve the required effect, but the storing of the tune internally enables this to be done without difficulty.

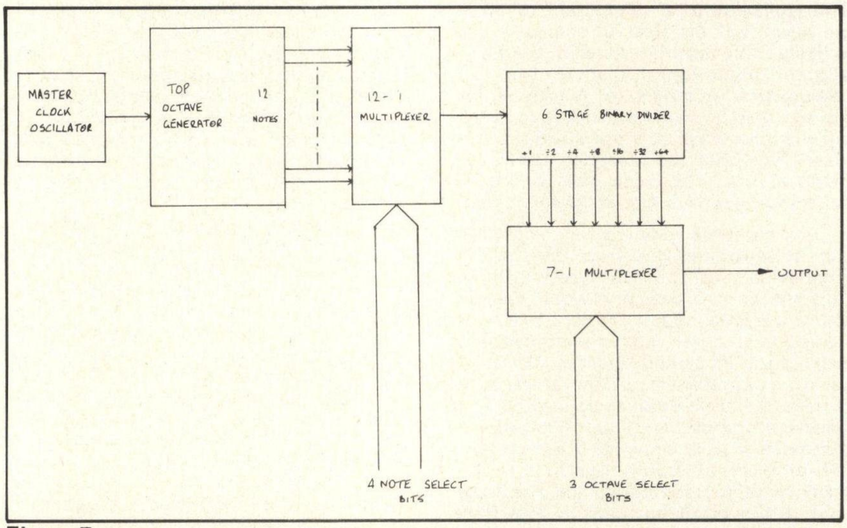


Figure 5.
A hardware note generator. The 12-way multiplexer selects the wanted note while the 7-way one selects the wanted octave.

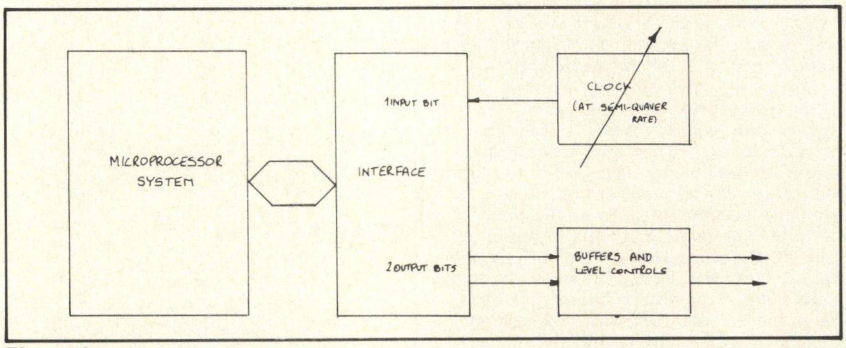


Figure 6.
The basic hardware required to support the system. The number of output channels which can be accommodated is shown as two but this can be increased if desired.

The Weak Link

The weak link in the system as described so far is the stage of transcribing the music

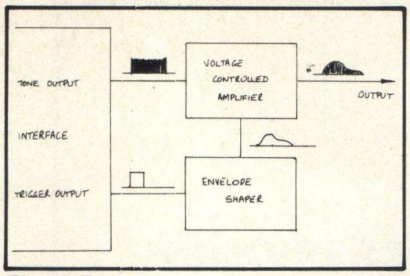


Figure 7.
Voiced notes can be generated quite easily by the use of an envelope generator and a voltage controlled amplifier.

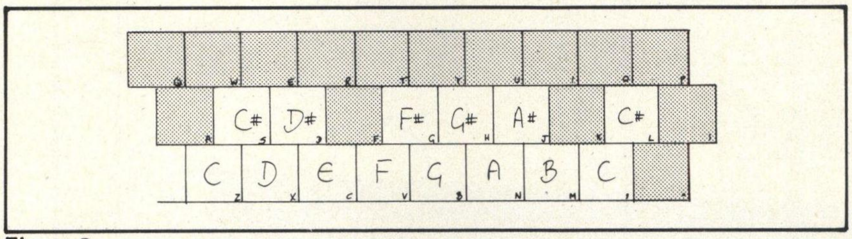


Figure 8.
If some of the keys (the shaded ones) are ignored, a normal typewriter keyboard can be used as a simple piano type keyboard. This can then be used as a means of entering notes into the memory of the computer.

from its written form into the binary form used by the computer. Clearly, what is needed is some kind of loader to cut out the middle step. This could take the form of a normal memory loader where the notes are entered in terms of their letter names e.g. D, C sharp etc., but this would be very prone to error.

An alternative, if very expensive, method would be to use a keyboard of the electronic organ variety with the appropriate encoding circuitry, but this is really out of the range of the budget of the average computerist who wants to make music for friends' enjoyment. A much cheaper alternative is to utilise the typewriter keyboard, which most systems will already use, as a simple piano keyboard. It is possible to use the bottom row (ZXCV. . .) as the white notes and some of the next row (ASDF. . .) as the black notes, as shown in figure 8. This does nothing to solve the question of storing the duration of the note in the computer, but can relieve much of the tedium of translating the notes into hexadecimal.

Soft Music

To turn now to the software needed to actually make the music, figure 9 shows a very basic routine which will read one line of music from memory and play it through a loudspeaker connected to an amplifier fed from a single bit output port. It incorporates no frills and is intended only as a basis on which can be built a more sophisticated system. Where more than one channel is available, this can be used in a variety of ways. One is to encode two lists of notes into memory and scan these separately. In this way, it is possible to play what is known as counterpoint, where the two lines have different tunes which nevertheless blend together to produce an overall pleasing effect. Another way would be to have only one list of notes, but to program the computer in such a way as to have the other channels play the notes necessary to produce a chord under the main note. These rules are in no way simple and require a good deal of work to learn, let alone program! This would however make an interesting project for the very musical.

In a similar way, the percussion generators could either be triggered by the control codes within the list of notes, or sounded according to a predetermined pattern, as is done by LSI rhythm generator chips. It is a relatively trivial task to program the computer to give the operator some idea of the beat; the information that would be given by a conductor in an orchestra. This can simply be an LED which lights on the downbeat or some audible metronome click. A more complex system can easily be envisaged if this is desired. This will enable a conventional instrument to be played with the computer and will help in synchronising when a multi-track recording is being made.

When the music system is commissioned and fully working, it remains but to find some music to play on it. The very talented will no doubt compose a special piece for the occasion; but for the more down-to-earth, it is a matter of using the work of someone else and if necessary adapting it. The most likely candidates for this treatment are songs since the human voice is of necessity monophonic; that is, it can sound only one note at a time. Any good public library should have copies of the scores of musicals and these should provide excellent material. When more advanced, the accompaniment can also be programmed or, alternatively, played separately and mixed with a recorded version of the other track, the result being recorded on a second tape recorder.

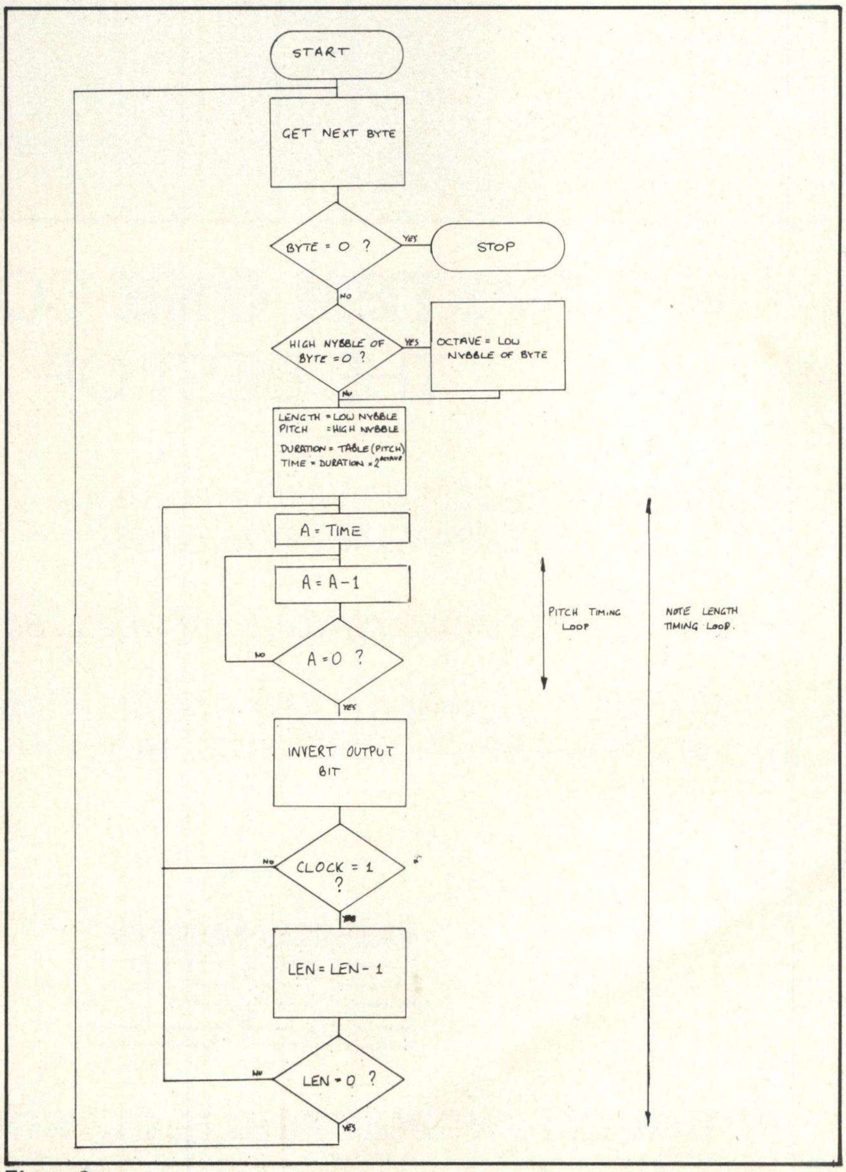


Figure 9.
The software needed to operate the system need not be complicated as is demonstrated by this simple routine.

It is hoped that these notes will prompt some people who have a personal computer to enter the fascinating world of music. Anyone with an idea relevant to this field is welcome to write to the author (with an SAE if a reply is wanted) and if it is justified, another article may be based on these. Happy music making!

PCW *Paul Jessop may be reached at 1157 Warwick Road, Solihull, West Midlands B91 3HQ.* **PCW**

THE MICRO MUSE

Background to Micro Poetry–3

Eric Finlay

"... Who taught you to drill
Those hoes won't slice them, hatchets-won't-crack-them,
Won't be wrenched open, won't be worked loose,
Maddening boxes within boxes, inside the brocade box?"
Kuan Han-Ch'ing. 1220 - 1300

MICRO-POEM II

```
                              ;SC/MP 11
                              ;  7-Segment 8-bit display.
OF12;C2,00   OF1A;C2,02   OF22; C2,04   OF2A;  C2,06
OF14;C9,00   OF1C;C9,02   OF24; C9,04   OF2C; C9,06.
OF16;C2,01   OF1E;C2,03   OF26; C2,05   OF2E;  C2,07
OF18;C9,01   OF20;C9,03   OF28; C9,05   OF30; C9,07.
                                        OF32;  90,DE.
....  ..                                ;main program.
OFF9  OD  P1(H)        OF40   77
OFFA  OO  P1(L)        OF41   79
OFFB  OF  P2(H)        OF42   6D
OFFC  40  P2(L)        OF43   40
; sets pointer-        OF44   66
      registers        OF45   79
                       OF46   31
                       OF47   6F
                       OOOO  ; text begins.
```

The reader should now be able to identify the three sections of the above program, and follow it without much difficulty. The main program repeats the C2;Load Accumulator from Pointer 2 (plus displacement) instruction, which refers to address OF40 in the text data-stack, and then uses the C9;Load Pointer 1 instruction to put the letter 'A' (Hex 77), in the first digit of the 8-bit LED display. The rest of the program is a simple elaboration of that sequence.

Data loading at OFF9—OFFC sets the Pointer-registers, and the text is stored in RAM from OF40—OF47.

The poem is a tribute to the four graces (Darling, Fields, Kelly, and W.C.), whilst also being a luminous version of Debussy's 'La Mer' in gallium phosphide.

I have designed a small program around the 1E (RR) instruction. Rotation is a standard technique in mathematics, and is used in the serial music of such composers as Schoenberg, Webern, and Alban Berg. It is most easily understood by studying a binary number, and observing how the 1E/RR (Rotate Right) instruction takes the binary digit at the right hand end of the number row, and puts it at the beginning of the row.

	0 1 0 1 1 1 0 0	5C	
RR	0 0 1 0 1 1 1 0	2	
RR	0 0 0 1 0 1 1 1	1	
RR	1 0 0 0 1 0 1 1	8B	

To rotate a number, and therefore an image, in the computer we put the display routine address (ODOO) in Pointer-register 1. An address in RAM (say OF90) is set to hold the hexadecimal number we wish to rotate (say 5C); and we put the chosen RAM address into Pointer-register 2.

The rest of the sequence is elementary, the data in P2 is loaded into the Accumulator, rotated right with the instruction 1E; and stored back into P2. Then the new number (rotated once) is loaded back into the Accumulator and transferred from there to the Display Routine, at the position (04) indicated by the displacement data. Three consecutive 8F/FF delay instructions enable us to observe the image, then a Jump instruction (90) makes a loop back of —17 (EF) to the beginning of the program. The sequence is then repeated by the computer through all the possible rotations of the original number.

If the time-delay section is removed from the program, and the program is run in a loop with a new displacement of —11 (F5), pressing and holding the reset button will stop the rotation at an unpredictable sign. In this way a poem can also be used as a roulette wheel, which would have amused Dostoevsky.

Here is the complete program:—

MICRO-POEM III.

```
                      ; Continuous finite rotation
                        of binary number.
                      ; SC/MP 11.
                      ; 8-bit LED Display.
      OFF9   OO  P1(H)
      OFFA   OO  P1(L)
      OFFB   OF  P2(H)
      OFFC   90  P2(L)
                 ; sets pointer registers.
      OF90   5C  ; image
OF20 C2     OF29  8F
OF21 OO     OF24  FF
OF22 1E     OF2B  8F
OF23 CA     OF2C  FF
OF24 OO     OF2D  8F
OF25 C2     OF2E  FF
OF26 OO     OF2F  90  Jump
OF27 C9     OF30  EF  —17
OF28 04     OOOO  OO
                      ; end.
```

The final micro-poem I wish to discuss emerged from an experiment with the running-text program designed by D. Johnson-Davies for the Mk 14 basic SC/MP 11 system. In an effort to comprehend the program more fully I entered the text 'All Quiet on the Western Front' into RAM and ran this without any mistakes. I then altered the timing parameter at OF2F from FO, to F9. The result was astonishing; an acrobatic series of runs, jumps, optical rhythms and other transformations of an alphanumeric and kinetic nature. The program terminated, not surprisingly, in an internal loop (apparently infinite). I reset the CPU, and examined the data in the RAM section devoted to the text. The contents were radically altered. I made a note of the transformed text, then restoring the control section of the program to its original state, I ran the new data.

The text had become 'Quiet on the Western Front', followed by a very good simulation of an artillery barrage. I will give the original text here, and the transformed text for comparison. I have called the new poem "Requiem for G.F.".

```
          ; SC/MP 11 running text program.
          ; d.j.d.
          ; text backwards from OFD5.

OFAO    080; 006, 03E, 067, 040, 038,
OFA6    038, 077, 040, 040, 040, 040,
OFAC    040, 040, 040, 040, 040, 040,
OFB2    040, 040, 040, 040; 040, 040,
OFB8    078, 064, 03F, 050, 071, 040,
OFBE,   064, 050, 079, 078, 06D, 079,
OFC4    07E, 040, 079, 074, 078, 040,
OFCA    064, 03F, 040, 078, 079, 006,
OFDO    03E, 067, 040, 038, 038, 077.
```

```
          ; 'Requiem for G.F.'
          ; micro-poem iv.
          ; transformation of above text:
          ; program step OF2F changed to data FO.
OFAO    01F, 01F, 03E, 01F, 040, 01F,
OFA6    01F, 01F, 01F, 040, 01F, 01F,
OFAC    040, 01F, 01F, 01F, 01F, 01F,
OFB2    040, 01F, 040, 01F, 040, 01F,
OFB8    078, 064, 03F, 050, 071, 040,
OFBE    064, 050, 079, 078, 06D, 079,
OFC4    07E, 040, 079, 074, 078, 040,
OFCA    064, 03F, 040, 078, 079, 006,
OFDO    03E, 067, 01F, 038, 01F, 077.
          ; run text on standard program.
          ; e.f.
```

It is best to program and perform the straight text, alter the program as described, observe the transformation sequence (which is a poem in itself), then reset and debug the original program. Setting the sweep-speed data at OF2D to C4, seems to sharpen the optical analogy in the final version. The text program should correspond to the data given here for an effective performance of the poem; but experiments are encouraged.

Although the CPU I use is not much more complex than an electronic slide-rule, I have gained considerable pleasure and intellectual satisfaction from procedures which seem to challenge all the creative faculties of the mind. The microprocessor is not just a collection of registers and logic circuits. It is an embodiment of the most advanced thinking of our time, and offers a reciprocal interplay at that level of thought. Whatever computers may imply for the future, the microprocessor is the brainchild of the scientific idealism of the 18th Century, and the scientific penetration of the 19th and 20th Centuries. Micro-poetry puts the craft of poetry into a new dimension, just as surely as the computer helped to put man into outer-space.

Hexadecimal Conversion

David Simpson
Department of Mathematics
Trinity and All Saints' Colleges, Leeds

Although hexadecimal addressing of microprocessor memory locations is common, there are occasions when it is useful to be able to convert hex addresses to decimal ones. In Commodore PET BASIC, for example, the PEEK,POKE, and SYS statements refer to decimal addresses. The following program, which, if necessary, could be rewritten as a subroutine of a larger program containing such statements, covers a range of 64K from zero to 65535 (FFFF in hex) and converts from hex to decimal or vice versa.

The number to be converted is entered either as a decimal integer with up to five digits or as a four digit hex integer preceded by a $ character (e.g. $ B5AF or $ 0076). The presence or absence of this character controls the direction of conversion automatically. The simple validation check (line 140 or 240) to which each entered number is subjected could be made more rigorous if desired.

The program, which is written in Commodore PET BASIC, carries out arithmetic on ASCII codes using several string functions and concatenation. Transcription to a version of BASIC which does not possess all these facilities may not therefore be a straightforward process. To save storage, several lines containing multiple statements have been included. These may, of course, be written on separate lines.

```
100 LET S=0: LET A$=""
110 PRINT"WHICH NUMBER";: INPUT N$
120 IF LEFT$(N$,1)="$" GOTO 210
130 LET N=VAL(N$)
140 IF N>65535 GOTO 290
150 FOR I=1 TO 4
160 LET Q=INT(N/16): LET R=N-16*Q+48
170 IF R>57 THEN LET R=R+7
180 LET A$=CHR$(R)+A$: LET N=Q: NEXT I
190 PRINT"IN HEX THIS IS ";A$
200 GOTO 300
210 LET N$=RIGHT$(N$,4)
220 FOR I=0 TO 3
230 LET X=ASC(MID$(N$,4-I,1))-48
240 IF X>22 GOTO 290
250 IF X>9 THEN LET X=X-7
260 LET S=S+X*16↑I: NEXT I
270 PRINT"IN DECIMAL THIS IS ";S
280 GOTO 300
290 PRINT"INVALID ENTRY - ";
300 PRINT"ANOTHER CONVERSION";
310 INPUT A$: IF A$="YES" GOTO 100
999 END
```

Decimal Hexadecimal Binnary Conversion Table

L. F. Heller

	DECIMAL	HEX	BINARY
	0	0	0000
	1	1	0001
	2	2	0010
	3	3	0011
	4	4	0100
	5	5	0101
	6	6	0110
	7	7	0111
(1)	8	8	1000
	9	9	1001
	10	A	1010
	11	B	1011
	12	C	1100
	13	D	1101
	14	E	1110
	15	F	1111
	16	10	0001
	32	20	0010
	48	30	0011
	64	40	0100
	80	50	0101
	96	60	0110
	112	70	0111
(2)	128	80	1000
	144	90	1001
	160	A0	1010
	176	B0	1011
	192	C0	1100
	208	D0	1101
	224	E0	1110
	240	F0	1111
	256	100	0001
	512	200	0010
	768	300	0011
	1024	400	0100
	1280	500	0101
	1536	600	0110
	1792	700	0111
(3)	2048	800	1000
	2304	900	1001
	2560	A00	1010
	2816	B00	1011
	3072	C00	1100
	3328	D00	1101
	3584	E00	1110
	3840	F00	1111
	4096	1000	0001
	8192	2000	0010
	12288	3000	0011
	16384	4000	0100
	20480	5000	0101
	24576	6000	0110
	28672	7000	0111
(4)	32768	8000	1000
	36864	9000	1001
	40960	A000	1010
	45056	B000	1011
	49152	C000	1100
	53248	D000	1101
	57344	E000	1110
	61440	F000	1111
	65535 =	FFFF =	(1111 1111 1111 1111)
			4 3 2 1

DECIMAL	HEX	BINARY
65536	10000	0001
131072	20000	0010
196608	30000	0011
262144	40000	0100
327680	50000	0101
393216	60000	0110
458752	70000	0111
5.524288	80000	1000
589824	90000	1001
655360	A0000	1010
720896	B0000	1011
786432	C0000	1100
851968	D0000	1101
917504	E0000	1110
983040	F0000	1111
1048575 =	FFFFF =	(1111 1111 1111 1111 1111)
		5 4 3 2 1

EXAMPLES

BINARY	0101	1100	0100	1010	= 23626
HEX	5	C	4	A	= 23626

A. BINARY TO HEX — By Inspection

B. HEX TO BINARY — By Inspection

C. HEX TO DECIMAL — (From Table)

Example: Change Hex 5C4A to Decimal

x5000	=	20480
x C00	=	3072
x 40	=	64
x A	=	10
Ans.		23626

D. BINARY TO DECIMAL — Change Binary to HEX by Inspection then to Decimal as in "C" above.

E. DECIMAL TO HEX and BINARY

Example: Change Decimal 23626 to Hex/Binary

Choose the highest number in the Decimal Column of the TABLE that does not exceed the given number. For 23626 use 20480; for 3146 use 3072, Etc.

			Hex	Binary
	23626			
From table	−20480	=	5000	0101
	3146			
From table	− 3072	=	C00	1100
	74			
From table	− 64	=	40	0100
	10			
From table	−10	=	A	1010

Ans. Hex. 5C4A

Binary 0101 1100 0100 1010

4 3 2 1

Getting It Together

Build your own assembler — listing (continued)

Mike Banahan

```
0C76                                 IR<-(HL)
0C76 3E46B6                     A<-≠106‡OR B
0C79 C3F009                     J LO.OX
0C7C F2ED09           LO.SR5:   J P,LO.BEX
0C7F                                 IR<-(DE) OR R<-(BC)
0C7F 79FE07C2ED09               A<-C‡CMP≠7‡J NZ,LO.BEX
0C85 3E0AB4                     A<-≠12‡OR H
0C88 C3F009                     J LO.OX
0C8B CDB10FCDB80F2011 LO.SR4:   CALL EVALUA‡CALL SEMCR‡BR NZ,LO.SR6
0C93 79FE07C2ED09               A<-C‡CMP≠7‡J NZ,LO.BEX
0C99                                 IA<-SYMBOL
0C99 3E3ACDFD11                 A<-≠72‡CALL OUTINS
0C9E CDF809C3F309               CALL OUTHL‡J LO.BEX
0CA4 FE28C2ED09       LO.SR6:   CMP≠'(‡J NZ,LO.BEX
0CA9 CD3601C8                   CALL GETATM‡EX AF
0CAD FE29C2ED09                 CMP≠')‡J NZ,LO.BEX
0CB2 E5                         PUSH HL‡ISAVE VALUE SHOP AT MICK'S
0CB3 21CF0CCDD30FD1             HL<-≠DRTAB‡CALL SEARCH‡POP DE
0CBA E608CAED09                 AND≠10‡J Z,LO.BEX
0CBF                                 IFINE. IT'S IX OR IY
0CBF 3EDDB4CDFD11               A<-≠335‡OR H‡CALL OUTINS
0CC5 3E46B0CDFD11               A<-≠106‡OR B‡CALL OUTINS
0CCB 7B                         A<-E
0CCC C3F009                     J LO.OX
0CCF 4243000000000000 DRTAB:    .WORD 'B,'C,0,0,0,0,0,0
0CD7 4445000000000110           .WORD 'D,'E,0,0,0,0,1,20
0CDF 484C000000000220           .WORD 'H,'L,0,0,0,0,2,40
0CE7 5350000000000330           .WORD 'S,'P,0,0,0,0,3,60
0CEF 4146000000000730           .WORD 'A,'F,0,0,0,0,7,60
0CF7 4958000000000800           .WORD 'I,'X,0,0,0,0,10,0
0CFF 4959000000000920           .WORD 'I,'Y,0,0,0,0,11,40
0D07 00                         .WORD 0
0D08 4100000000000738 SRTAB:    .WORD 'A,0,0,0,0,0,7,70
0D10 4200000000000000           .WORD 'B,0,0,0,0,0,0,0
0D18 4300000000000108           .WORD 'C,0,0,0,0,0,1,10
0D20 4400000000000210           .WORD 'D,0,0,0,0,0,2,20
0D28 4500000000000318           .WORD 'E,0,0,0,0,0,3,30
0D30 4800000000000420           .WORD 'H,0,0,0,0,0,4,40
0D38 4C00000000000528           .WORD 'L,0,0,0,0,0,5,50
0D40 00                         .WORD 0
0D41
0D41                  I         ONEPRO IS DEAD SIMPLE.
0D41                  I         ANY ONE OR TWO WORD OPCODES WITH
0D41                  I         NO OPERANDS ARE SORTED OUT BELOW
0D41
0D41                  ONEPRO:
0D41 F5E5                       PUSH AF,HL
0D43 211F14CDD30F200C           HL<-≠ONETAB‡CALL SEARCH‡BR NZ,ONEBAD
0D48 CDFD11                     CALL OUTINS‡I   A CONTAINS THE INSTRUCTION
0D4E 7CB72809                   A<-H‡TSTA‡BR Z,ONERET
0D52 CDFD11                     CALL OUTINS‡I   GET RID OF SECOND BYTE
0D55 1804                       BR ONERET
0D57 3C32EB12         ONEBAD:   INC A‡ONEERR<-A
0D5B E1F1C9           ONERET:   POP HL,AF‡RET
0D5E 00               EMPTYF:   .BLKZ 1‡IIS LINE BUFFER EMPTY?
0D5F
0D5F                  I         TAPEIN READS A LINE OF TEXT INTO THE LINE BUFFER.
0D5F                  I         IT ALSO DEALS WITH MUCH OF THE LISTING AND
0D5F                  I         ERROR REPORTING.
0D5F
0D5F E5D5F5C5         TAPEIN:   PUSH HL‡PUSH DE‡PUSH AF‡PUSH BC
0D63 3AB701FE022011             A<-PASFLG‡CMP≠2‡BR NZ,T.NP2
0D6A 3AB601B7C4E100             A<-FIRSLN‡TSTA‡CALL NZ,LINPRN
0D71 2F32B601                   CMA‡FIRSLN<-A‡I COMPLEMENT ENSURES NONZERO
0D75 2A6815CDC612               HL<-CURADD‡CALL PR16
0D7B 2148FE1600       T.NP2:    HL<-≠INBUFF‡D<-≠0
0D80 CD140EE67F       T.NEXT:   CALL TCHIN‡AND≠177
0D85 FE7F28F7                   CMP≠RUBOUT‡BR Z,T.NEXT
0D89 FE0A28F3                   CMP≠LF‡BR Z,T.NEXT
0D8D FE002016                   CMP≠0‡BR NZ,T.PUT
0D91 3A5D01B728F9               A<-NULFLG‡TSTA‡BR Z,T.NEXT
0D97 CDA80E4F5554204F46 T.NUL:  CALL PRINT‡.ASCIZ%OUT OF TAPE%
0DA6 C30001                     J BEGIN‡I        DON'T CHANGE THIS
0DA9 CDD10D7732B5012314 T.PUT:  CALL T.CHK‡(HL)<-A‡NULFLG<-A‡INC HL‡INC D
0DB2 FE0D28GD                   CMP≠CR‡BR Z,T.RET
0DB6 7AFE3F20C5                 A<-D‡CMP≠77‡BR NZ,T.NEXT‡I        BUFFER FULL
0DBB 3E0D77                     A<-≠CR‡(HL)<-A
0DBE 21EA12CBCE                 HL<-≠LINLNG‡SET 1,(HL)‡I        LINE TOO LONG
0DC3 325E0D           T.RET:    EMPTYF<-A
0DC6 2A6815226A15               HL<-CURADD‡OURDOT<-HL‡I SET DOT
0DCC C1F1D1E1C9                 POP BC‡POP AF‡POP DE‡POP HL‡RET
0DD1 E5               T.CHK:    PUSH HL
0DD2 CD9A0FB7                   CALL WOTSIT‡TSTA‡I        VALID CHARACTER?
0DD6 2005                       BR NZ,.+7
0DD8 21E312CBCE                 HL<-≠CHERR‡SET 1,(HL)
0DDD 08E1C9                     EX AF‡POP HL‡RET
0DE0 3AEC12ED44C6052B0A LINPRN: A<-INSNUM‡NEG‡ADD≠CWIDTH‡BR Z,L.NCD
0DE9 47                         B<-A
0DEA CDA80F20202000010F7        CALL PRINT‡.ASCIZ%   %‡DBNZ .
0DF3 2148DFCDA80F202000 L.NCD:  HL<-≠INBUFF‡CALL PRINT‡.ASCIZ%  %
0DFC 7E23FE0DC4B70E20F7         A<-(HL)‡INC HL‡CMP ≠CR‡CALL NZ,CHOUT‡BR NZ,.
0E05 CDCC0F                     CALL CRLF
0E08 AF32EC12                   CLA‡INSNUM<-A
0E0C CDEE13                     CALL ERRPRN
0E0F CDFD12C9                   CALL ERRST‡RET
0E13
0E13                  T.POS:    .BLKW‡I POINTER
0E14 E5D5C5           TCHIN:    PUSH HL,DE,BC‡I READ 64 CHARACTERS AT A TIME
0E17 3A130EFE40                 A<-T.POS‡CMP≠100
0E1C 2016                       BR NZ,T.GOT
0E1E 0640                       B<-≠100
0E20 216C15                     HL<-≠TBUFF
0E23 3E04D3DE                   A<-≠4‡OUT TTYCNT,A‡I      START READER
0E27 CDC10E           T.NEX:    CALL CHIN
0E2A 7723                       (HL)<-A‡INC HL
0E2C 10F9                       DBNZ T.NEX
0E2E AF32130E                   CLA‡T.POS<-A
0E32 D3DE                       OUT TTYCNT,A‡I  STOP READER
0E34 3A130E216C15     T.GOT:    A<-T.POS‡HL<-≠TBUFF
0E3A 85                         ADD L
0E3B 3001                       BR NC,.+3
0E3D 24                         INC H
0E3E 6F                         L<-A
0E3F 7E21130E34                 A<-(HL)‡HL<-≠T.POS‡INC (HL)
```

```
0E44 C1D1E1C9                     POP BC,DE,HL*RET
0E48                     INBUFF:  .BLKW 100*!     THE LINE BUFFER
0E88                     SEMCR:
0E88                              !IS THE TERMINATOR END OF STATEMENT?
0E88 3AB811                       A<-TERMIT
0E8B FE3BC8                       CMP#'*#RET Z
0E8E FE0DC8                       CMP#CR#RET Z
0E91 C9                           RET
0E92                     WOTATM:
0E92 3AAC15CD9A0E                 A<-ATBLCK*CALL WOTSIT*! INSPECT FIRST CHAR. OF ATOM
0E98 C9                           RET
0E99 08                  WOTSAT:  EX AF
0E9A                     WOTSIT:  !AF IN 'AF ON EXIT
0E9A E5C5                         PUSH HL*PUSH BC
0E9C 06004F                       B<-#0*C<-A
0E9F 213811                       HL<-#WORDTB
0EA2 09                           ADD HL,BC*!POINT TO WORD
0EA3 08                           EX AF*!DON'T LOSE A
0EA4 7E                           A<-(HL)*!FIND TYPE
0EA5 C1E1                         POP BC*POP HL
0EA7 C9                           RET
0EA8                     PRINT:   *!      PRINT A MESSAGE
0EA8 E3F5                         EX (SP),HL*PUSH AF
0EAA 7EB72806            PR.NEX:  A<-(HL)*TSTA*BR Z,PR.L
0EAE CD870E231AF6                 CALL CHOUT*INC HL*BR PR.NEX
0EB4 F1E3C9              PR.L:    POP AF*EX (SP),HL*RET
0EB7                     CHOUT:   *!      ONE CHARACTER TO TELETYPE
0EB7 F5                           PUSH AF
0EB8 DBDD8730FB                   IN A,TTYSTA*ADD A*BR NC,.
0EBD F1D3DCC9                     POP AF*OUT TTYTR,A*RET
0EC1                     CHIN:    *!      ONE CHARACTER FROM TELETYPE
0EC1 DBDDE6402BFA                 IN A,TTYSTA*AND #100*BR Z,.
0EC7 DBDCE67FC9                   IN A, TTYTR*AND#177*RET
0ECC                     CRLF:
0ECC CDA80E0D0A00C9               CALL PRINT*.ASCIZ <CR>,<LF>*RET
0ED3
0ED3                     !        SEARCH LOOKS THROUGH THE SYMBOL TABLE POINTED TO
0ED3                     !        BY HL AND TRIES TO FIND A MATCH WITH THE TEXT IN
0ED3                     !        ATBLCK. IF A MATCH IS FOUND IT RETURNS WITH THE
0ED3                     !        ZERO FLAG CLEAR, HL CONTAINING THE 'VALUE'
0ED3                     !        ASSOCIATED WITH THE SYMBOL FOUND AND DE
0ED3                     !        POINTING TO THE NEXT ENTRY IN THE TABLE. IF NO
0ED3                     !        MATCH IS FOUND, EXITS WITH ZERO SET.
0ED3
0ED3                     SEARCH:
0ED3 C5                           PUSH BC
0ED4 E5DDE1110800                 PUSH HL*POP IX*DE<-#10
0EDA 21AC15DDE5          SEAR.L:  HL<-#ATBLCK*PUSH IX
0EDF DD7E00B72822                 A<-0(IX)*TSTA*BR Z,SE.NOT
0EE5 0606                         B<-#6
0EE7 DD7E00E67FBE202023           A<-0(IX)AND#177*CMP (HL)*BR NZ,SE.ADD*INC HL,IX*DBNZ .
0EF4                                              !SO WE'VE FOUND IT!
0EF4 DD7EFFE680321 50F            A<--1(IX)*AND #200*MAYCHA<-A
0EFC DDE5E1                       PUSH IX*POP HL
0EFF 5E235623                     E<-(HL)*INC HL*D<-(HL)*INC HL*!POINT TO NEXT
0F03 AF                           CLA*    !ZERO SAYS IT'S BEEN FOUND
0F04 EB1803                       EX DE,HL*BR SEARET*!HL CONT'S VALUE FROM TABLE


0F07 676F3C              SE.NOT:  H<-A*L<-A*INC A*!RESET ZERO FLAG
0F0A 79DDE1C1C9          SEARET:  A<-L*POP IX,BC*RET*!DE POINTS TO NEXT SYMBOL
0F0F DDE1DD19            SE.ADD:  POP IX*ADD IX,DE
0F13 18C5                         BR SEAR.L
0F15                     MAYCHA:  .BLKW
0F16                     GETOUT:  *!      SKIP TO END OF BAD STATEMENT
0F16 CD880EC8                     CALL SEMCR*RET Z
0F1A CD3601                       CALL GETATM
0F1D 18F7                         BR GETOUT
0F1F                     LABPUT:  *!      PUT A SYMBOL IN THE SYMBOL TABLE
0F1F 21B415CDD30E                 HL<-#SYMTAB*CALL SEARCH
0F25 200D                         BR NZ,LAB.OK
0F27 3A150FB72846                 A<-MAYCHA*TSTA*BR Z,LABBAD
0F2D 1B1B                         DEC DE,DE*!POINT TO LOW ORDER BYTE
0F2F                                              !OF THE VALUE
0F2F CD9E0F183E                   CALL PUTVAL*BR LABRET
0F34 010600              LAB.OK:  BC<-#6
0F37 21AC15                       HL<-#ATBLCK
0F3A ED5B6A15                     DE<-SYMNEX
0F3E EDB0                         LDIR*!INSERT THE SYMBOL
0F40 1B                           DEC DE
0F41 3AB811FE3D200A               A<-TERMIT*CMP#'=*BR NZ,LAB.NX
0F48 1AF68012                     A<-(DE)*OR #200*(DE)<-A*!SETS PERMIT CHANGE FLAG
0F4C 13CD9E0F                     INC DE*CALL PUTVAL
0F50 1808                         BR LAB.NY
0F52 13216815            LAB.NX:  INC DE*HL<-#CURADD
0F56 EDA0EDA0                     LDI*LDI*!CURRENT ADDRESS GOES
0F5A                                        ! TO LAST TWO BYTES
0F5A ED536A15            LAB.NY:  SYMNEX<-DE
0F5E 21FF1FAF                     HL<-#SYMLIM*XOR A*!CLEAR CARRY
0F62 12                           (DE)<-A*!MARK END OF TABLE
0F63 ED52300A                     SBC HL,DE*BR NC,LAB.1
0F67 CDA80F53544F0D0A00           CALL PRINT*.ASCIZ*STO*,<CR>,<LF>*!       SYMBOL TABLE OVERFLOW
0F70 C7                           RST 0*! RETURN TO MONITOR
0F71 3C                  LAB.1:   INC A*!CLEAR ERROR FLAG
0F72 C9                  LABRET:  RET
0F73 3AB7011B1B1B1B      LABBAD:  A<-PASFLG*DEC DE,DE,DE,DE
0F7A 3D2006                       DEC A*BR NZ,LABB.1
0F7D                                              !SET MULTIPLE DEF FLAG
0F7D                                              !DE IS
0F7D                                              !NOW POINTING TO
0F7D                                              !PENULTIMATE CHARAC
0F7D 1AF680121BEF                 A<-(DE)*OR #200*(DE)<-A*BR LABRET
0F83 D5ED5B6815          LABB.1:  PUSH DE*DE<-CURADD
0F88 B7ED52                       TSTA*SBC HL,DE
0F8B 280521E412CBCE               BR Z,.+7*HL<-#SEQERR*SET 1,(HL)
0F92 D1                           POP DE
0F93 1AE680                       A<-(DE)*AND#200
0F96 28DA                         BR Z,LABRET
0F98 21E8127718D4                 HL<-#MULERF*(HL)<-A*BR LABRET
0F9E                     PUTVAL:
0F9E CD3601CDB10F                 CALL GETATM*CALL EVALUA
0FA4 7D1213                       A<-L*(DE)<-A*INC DE
0FA7 7C1213                       A<-H*(DE)<-A*INC DE
0FAA C9                           RET
0FAB
0FAB                     !        THE DREADED MATH ROUTINES.


0FAB                     !        IGNORE 'EM, THEY'RE HORRIBLE
0FAB
0FAB                     BRCKNM:  .BLKW
0FAC                     POINTR:  .BLKA
0FAE                     GEVALU:  *!      RETURN WITH 16 BITS IN HL, H ALSO IN A
0FAE CD3601                       CALL GETATM
0FB1                     EVALUA:
0FB1 AF32AB0F                     CLA*BRCKNM<-A
0FB5 C5D5                         PUSH BC,DE
0FB7 ED73AC0F                     POINTR<-SP*!REMEMBER WHAT SP WAS
0FBB 32E712                       EVFATR<-A*!CLEAR FLAG
0FBE CD2B10              EVA.PE:  CALL EVAL
0FC1 3AB811CD880F2847    EVA.TE:  A<-TERMIT*CALL SEMCR*BR Z,EV.CHX
0FC9 FE2C2843                     CMP#',*BR Z,EV.CHX
0FCD FE28283F                     CMP#'(*BR Z,EV.CHX
0FD1 FE5F283B                     CMP#'<-*BR Z,EV.CHX
0FD5 CD9A0EE6D4283F               CALL WOTSIT*AND#MATH*BR Z,EVA.NM
0FDC 08                           EX AF
0FDD E5F5CD3601                   PUSH HL*PUSH AF*CALL GETATM
0FE2 CD2B10                       CALL EVAL
0FE5 F1D1                         POP AF*POP DE*!SECOND ARGUMENT FOR MATH ROUTINES
0FE7 FE2B280C                     CMP#'+*BR Z,ADD
0FEB FE2D280B                     CMP#'-*BR Z,SUB
0FEF FE2A2802                     CMP#'**BR Z,MULT
0FF3 18CC                DIV:     BR EVA.TE*!     DIV AND MULT NOT IMPLEMENTED YET
0FF5 18CA                MULT:    BR EVA.TE
0FF7 1918C7              ADD:     ADD HL,DE*BR EVA.TE
0FFA EBED5218C2          SUB:     EX DE,HL*SBC HL,DE*BR EVA.TE
0FFF ED7BAC0FD1C1CD160F  EVA.BX:  SP<-POINTR*POP DE*POP BC*CALL GETOUT
1008 CBCF32E712                   SET 1,A*EVFATR<-A
100D 676F                         H<-A*L<-A
100F C9                           RET
1010 3AAB0FB720E9        EV.CHX:  A<-BRCKNM*TSTA*BR NZ,EVA.BX
1016 7CB7                         A<-H*TSTA
1018 D1C1C9                       POP DE*POP BC*RET
101B 08FE3E20DF          EVA.NM:  EX AF*CMP#'>*BR NZ,EVA.BX
1020 3AAB0FB728D9                 A<-BRCKNM*TSTA*BR Z,EVA.BX
1026 3D32AB0FC9                   DEC A*BRCKNM<-A*RET
102B                     EVAL:    !WHEN CALLED, ATBLCK
```

```
1028                              !MUST ONLY CONTAIN A VALID
1028                              !NUMBER OR LABEL
102B 3AB811                       A<-TERMIT
102E FE272862                     CMP#''#BR Z,EV.APO
1032 FE3C2011                     CMP#'<#BR NZ,EVNORM
1036 3AABOF3C32A30F               A<-BRCKNM#INC A#BRCKNM<-A
103D CD3601                       CALL GETATM#!GET RID OF<
1040 CDBE0FCD3601C9               CALL EVA.RE#CALL GETATM#RET#!GET RID OF >
1047 3A3501676F           EVNORM: A<-ATLENG#H<-A#L<-A
104C B7C8                                  TSTA#RET Z#!LOOKS AFTER +3, SAY
104E                                                 !BY MAKING IT LOOK
104E                                                           !LIKE 0+3
104E                                                           !NOT SO GOOD FOR *3!
104E D5C5                         PUSH DE#PUSH BC
1050 CD920E                       CALL WOTATM
1053 E6022010                     AND #ALPH#BR NZ,EV.ALF
1057 CD920E                       CALL WOTATM
105A E6082014                     AND #NUM#BR NZ,EV.NUM


105E AF676F2F32E9121808 EBADEX:   XOR A#H<-A#L<-A#CMA#EVERR<-A#BR EVRET
1067 21B415CDD30F       EV.ALF:   HL<-#SYMTAB#CALL SEARCH
106D 20EF                         BR NZ,EBADEX
106F C1D1C9             EVRET:    POP BC#POP DE#RET
1072 3AB811FE5E2805     EV.NUM:   A<-TERMIT#CMP#'&#BR Z,EV.AR#!   FIND THE RADIX
1079 3A32011811                   A<-CRADIX#BR EV.NAR
107E 3E0ACDAD10         EV.AR:    A<-#12#CALL NUMBER#!    RADIX IS 10 DECIMAL
1083 7DFE0238D6                   A<-L#CMP#2#BR C,EBADEX#!          NOT <2
1088 FE1130D2                     CMP #21#BR NC,EBADEX#!  OR >16
108C CD3601                       CALL GETATM#!           SKIP UPARROW
108F CDAD1018D8         EV.NAR:   CALL NUMBER#BR EVRET#!  RESULT IN HL
1094 D5C5               EV.APO:   PUSH DE,BC
1096 2A330146                     HL<-LBLOW#B<-(HL)
109A 237E32B811                   INC HL#A<-(HL)#TERMIT<-A
109F FE0D                         CMP#CR
10A1 2801                         BR Z,.+3#        !CAN'T AFFORD TO MISS CR
10A3 23223301                     INC HL#LBLOW<-HL
10A7 AF676878                     CLA#H<-A#L<-B#A<-H
10AB 18C2                         BR EVRET
10AD                    NUMBER:   !WARNING:         NUMBER ASSUMES IT IS FREE
10AD                                       !TO USE AF,BC,DE,HL,
10AD                                       !ONLY CALL IT FROM EVAL,
10AD                                       !IT MAY SCRAMBLE THE STACK ELSE
10AD 11AC154F                     DE<-#ATBLCK#C<-A#!        SAVE THE RADIX
10B1 210000                       HL<-#0#!                          RUNNING TOTAL
10B4 3A3501B72832                 A<-ATLENG#TSTA#BR Z,NUMBAD
10BA 1AB7C8             NUMLP:    A<-(DE)#TSTA#RET Z
10BD 13                           INC DE
10BE CD9A0FE6402827               CALL WOTSIT#AND #ALFNUM#BR Z,NUMBAD#!   OLD A IN 'A
10C5 D5                           PUSH DE#!        THE MULTIPLY STARTS HERE
10C6 79                           A<-C#!           THE RADIX
10C7 110000                       DE<-#0
10CA EB                 N.MULP:   EX DE,HL#!       HL WILL CONT. RESULT
10CB CB3F3001                     SRL A#BR NC,N.MUL1#!      ALSO SETS Z
10CF 19                           ADD HL,DE#!               Z NOT AFFECTED
10D0 2804               N.MUL1:   BR Z,N.MUDN#!             SO THIS WORKS
10D2 EB2918F4                     EX DE,HL#ADD HL,HL#BR N.MULP#!  DE *2
10D6 08D1               N.MUDN:   EX AF#POP DE#!  BACK TO BEFORE WOTSIT
10D8 D6303810                     SUB #'0#BR C,NUMBAD
10DC FE0A3802                     CMP #12#BR C,NUMB1
10E0 D607                         SUB #7
10E2 B93007             NUMB1:    CMP C#BR NC,NUMBAD#!      MUST BE <RADIX
10E5 856F30D1                     ADD L#L<-A#BR NC,NUMLP
10E9 2418CE                       INC H#BR NUMLP
10EC F1                 NUMBAD:   POP AF#!         BACKSTACK
10ED 186F                         BR EBADEX
10EF 2E454E440008E002   DOTTAB:   .WORD '.,'E,'N,'D,0,0#.ADDR END
10F7 2E424C4B5A00C002             .WORD '.,'B,'L,'K,'Z,0#.ADDR BLKZ
10FF 2E574F524400E002             .WORD '.,'W,'O,'R,'D,0#.ADDR WORD
1107 2E41444452000F802            .WORD '.,'A,'D,'D,'R,0#.ADDR ADDR
110F 2E424C4B41000D03             .WORD '.,'B,'L,'K,'A,0#.ADDR BLKA
1117 2E415343495A2C03             .WORD '.,'A,'S,'C,'I,'Z#.ADDR ASCIZ
111F 2E41534349493603             .WORD '.,'A,'S,'C,'I,'I#.ADDR ASCII
1127 2E424C4B57008303             .WORD '.,'B,'L,'K,'W,0#.ADDR BLKW
112F 2E000000000007702            .WORD '.,0,0,0,0,0#.ADDR NEWDOT
1137 00                           .WORD 0
1138
```

TO BE CONTINUED

TRS-80 LEVEL II BASIC

P. J. Turner

with a few helpful mod's

The TRS-80 Draws Itself.

After seeing three articles on the TRS-80 level 1, I am now about to fill that gap with the level 2 system from the stables of Tandy. It's very noticeable how much better the level 2 is after owning the level 1 for a few months. I had to get my system converted very quickly, and I give Tandy full marks for the speed and helpfulness to make this possible. Thanks to Mr J Bullock, Mr R King and Mr J Ridgeway of Tandy.

What Tandy will do to your old level 1, is to remove two ROM chips and fit a small printed circuit board, containing three chips for level 2, and this board in turn is connected by ribbon link to the old chip position; plus adjust two links and also adjust two preset power pots.

You then get two tapes. One tape is the program conversion, one side is for the 4k systems, and the other side is for the 16k systems, and the second tape is for data conversion, to convert your data record tapes. Lastly, included is the level 2 basic reference manual.

The TRS-80 is a far better system for having the level 2, but in my opinion Tandy at Fort Worth USA have been very crafty in designing this whole system from the start. To give an example of this, after you buy the level 1 for £500, (which in my opinion is a little high) you soon realize the limits of its Basic, in comparison to other Basics, so you spend a further £79 for the better basic level 2. This Basic is very pleasing, but it's not long before you again realize that it's also got limits; i.e. you do not have commands like: DEF FN, TIMES $, INSTR, MID $ LEFT, DEF USR, and ten USR calls, (in level 2 you do get one USR call).

To get these other Basic commands you will have to spend £628 to obtain the interface expansion and a disk unit. After this you have got all of Tandy's Basic. It looks like a USA incentive drive to get the man in the street to buy more, against fierce competition.

Now, as I have a **full circuit diagram** of the TRS-80, (write to me if you would like a copy), I have seen a number of points that could make the system better, to the user's requirement, such as access to NMI, and the clock, etc.

These are about the main grumbles, but one point that keeps on annoying me is the lack of an index, in both manuals. The only way over this, is to type your own index. I've complained to Tandy about this, and they in turn are reprinting new manuals (due out by now) with corrections, with an index. Also, some programs in level 1 cannot be converted to level 2, using the program conversion tape. This again is being corrected by Tandy, so most TRS-80 owners should get a copy. Anyway, the corrections are to be found at the end of this article.

While I am writing about corrections, also note these: To improve the keyboard buttons, so they can operate more quietly, and easier to press, and also to cure the fault of double entries, just lift each key cap, which is a push on type. If a small tool curved at one end is placed under the bottom of the cap, then you can lift it up. You will see the switch contacts, *don't touch them,* all you have to do is to spray into the switch a small quantity of switch cleaner, the type that leaves a film of lubricant. (Something like Radio Spares contact cleaner No. 554. 175 or Kontakt Kleena by Altham Electronics Ltd.) My keyboard now feels like an expensive professional type.

As the Z80 chip has the capability of lower case letters, Tandy will help to modify your board, by fitting a chip, piggy-back fashion onto the board's Z45 chip where the 2102 rams are (for the video memory). Tandy will arrange to carry out this mod for you, for a small fee which will cover the cost of the chip plus labour and time spent (about one hour) which I consider very fair as it is pretty cheap. Also, Tandy are making a proper computer data recorder, which will take the place of the audio recorder that is used at the moment.

The **back room boys at Tandy** are building at this moment in time, a prototype UHF modulator, also a S100, and the RS232, serial interfaces.

When Mike Dennis reviewed the level 1 in the September **PCW**, he ran into a small problem with a program,

```
10 FOR A = 1 TO 32500 STEP 10000
20 PRINT A
30 NEXT A
40 END
```

The print out wasn't as to be expected, but if you try it out on the level 2, then it runs ok, and prints,

```
1
10001
20001
30001
```

One of the functions in Basic always looked for is a full use of variables. In level 2 the variable names can be made up to about 900 using letters A to Z and digits 0 to 9. So, you could use combinations of A1 to A9, to Z1 to Z9, and AA to ZZ; e.g. AG,DR,ZY,SO. A few you may *not* use, and these are found in the reserved words at the back of the manual. The main point is that if you stick to two characters each time, your computer will always be happy.

For variable types level 2 has four, called Integer, Single precision, Double precision, and String variables. The first three store numerical values, with various degrees of precision that you specify.

The string variables can be a combination of letters, numbers, symbols, blanks etc., up to 255 characters long just by adding a string declaration character $, to the name, (a great deal better than level 1 that only had two strings). There are other declaration characters that can be used for precision work, these can be added to variable types such as %,D,#, ! So in one program you can mix them and yet they are still each a distinct variable name, i.e., A$, A%, A!, A#. All very useful.

In arrays you can still use the same declaration characters, so that A$ (A,B,C) would be a 3 dimensional array, containing string values, and A3(I,J) would be a two dimensional array, but of double precision values.

Level 2 graphics is the same as in level 1, i.e., SET, RESET, and POINT, which turn light blocks on and off. Graphic pictures are the weakest point in the TRS-80, but after a time, if you're like me, you'll soon get fed up with graphic pictures, and get down to more serious work, as the rest of the system is far from weak.

One very good point is the feature to convert 64 characters per line to 32 characters per line by hitting the shift key and the → key at the same time. This can be used in a program to bring out important parts by writing CHR$(23) in a line, and you can revert back by writing CLS. For instance, when I play O's and X's or draughts with my children the boards are printed in 32 characters per line, and the instructions are in 64 characters per line.

At the end of this article is a program to draw any picture that you wish once it is in. The controls are, (PRINTING MODE) "L" key to left of screen. "R" key to right of screen. "D" key for down. "U" key to go up.

(NON PRINTING MOVEMENT) W FOR up. Z FOR down. A FOR left. F FOR right.

This draws a line in horizontal or vertical directions, up or down, the cursor point flashes on and off, this helps you to follow it. You can draw *any* picture you like, and you can start at any point of the screen, so it seems there is a way to strengthen the TRS-80 graphics.

The editing mode on the level 2 is a treat to use, I've used other types of editing on other computers, and they all work out the same in the end. But once you have learned the editing in level 2, it's a breeze. In level 1 it was a nuisance, as you had to rewrite the line out again to correct a mistake, but only if you moved to a new line; otherwise, you had to back space to the mistake. As I say, a nuisance.

But with level 2 you can edit anytime or anywhere in the program. To enter the edit mode you write the word "EDIT", then the line number, and then press the "ENTER" key. You can exit in three ways, by the "ENTER" key, and by the "E" key which means (save changes and Exit), and a last way is by the "Q" key. I'll come back to these keys later.

The way to print characters in the edit mode, is the space bar. This prints each character of your program line, till you reach the mistake, or you can jump to the mistake by typing in the number of characters you wish to jump to, by hitting the space bar. Or, you can jump backwards any number of characters by again typing in a number and hitting the "Backspace" key. You can then review the whole line, while still in the editing mode, by hitting the "L" key. If you hit the "X" key you go to the end of the line in the insert mode, ready to start inserting more material, without exiting from the editing mode.

This gets better, as when you hit the "I" key you can insert a character, space, full word, or a line of characters anywhere you wish. But, if you don't like or want the editing changes that you have made, then hit the "A" key to cancel all editing done and restart again. But if the "E" key is hit, this saves all editing done and exits from the mode; or you could hit the "Q" key which cancels all editing and exits. The "H" key will hack out any material that the cursor is set at, and then insert new material at your command. The "D" key can be used to delete single characters at a time using the space bar. Or, if a number is typed in before hitting the "D" key, you can delete whole blocks of characters.

The "C" key will allow you to change that character, or you can type "C" then a number then the "L" key. This will change whole blocks of characters. I think that you now have some idea of the versatility of the editing in the level 2, but there's more. We have a *search* mode in editing, so that when you type in a number and the "S" key, this tells the computer to search for the nth occurrence of the character c, and this will move the cursor to that position. If you don't specify a number then the computer will search for the first occurrence.

Now, last one of all, and one that I like, is that we have a "K" key which is a (search and kill). The computer will search and kill all characters up to the nth occurrence of character c, just like playing Star Trek. And that reminds me, the TRS-80 will play the Star Trek game as seen in many games books, it runs very well in my system.

After the editing system, on to the error messages. In level 1 errors are printed, HOW? SORRY? and WHAT? with a question mark beside the faulty point. But in level 2 it's different, as it prints more specific information. There are 23 error message codes. Each code explains quite clearly what's wrong with your program line, and if you still can't find it, then you switch on the trace system, by typing "tron". This will trace out line by line, loops and all till the error is spotted.

I find this trace system very good, as it's helped me to find errors very quickly. Maybe in the future generation of computers, you could switch on an auto edit, which will have its own TRACE, SEARCH AND KILL, and INSERT CORRECTION all done automatically to correct your mistakes. Still, that would take out some of the fun of programming.

Talking of autos, the level 2 has a auto line print increment; it can print in ones, fives, tens, hundreds or any number you wish, and that's a great help if you have a long program to write. You can stop it any time by hitting the break key, and restart on any number that you wish. Incidentally, if you hit the break key at any time, you can carry on by typing "CONT". But *not* for the auto line.

Now, typing the word "SYSTEM" puts the computer into the machine monitor mode, so that you can load machine language routines or data, such as Tandy's T-BUG, EDITOR/ASSEMBLER, or IN-MEMORY INFORMATION Packages. One of the limits that level 2 has is the statement USR(x), as only one is allowable, but when the disk is fitted, you're then given ten USR(x) commands, so you're given one to whetten your appetite! Pity, its just like level 1 when you were only given two array limits of A$,B$. There's a lot more functions in level 2 that I've not even hinted at, such as a full math function system; 16 standards are in the firmware, but in fact you can make many more derived functions, like INVERSE(X), HYPERBOLIC(X), INVERSE HYPERBOLIC(X), X being any COS,SGN,SIN,TAN,SECANT, COSECANT.

As I've written books on Cosmology, mathematics being the strongest theme, the accuracy had to be good, so I spent three years on a HP25 programmable scientific calculator, which is very good, and I could rely on its answers. So I am looking for that reliability in the TRS-80 level 2, as it can give 16 digit figures with an expected one to two digit rounding off. I am now of the opinion, that I could have saved a lot of time if I had this computer three years ago. Not only can it work out black body energy curves on different quasars, but it can sum up a variation of mixed QSO's and work out the answers, and at the same time give a graphic histogram on the red shift plot.

This reminds me of the instring subroutine MID$ and LEN. This function takes two string arguments and tests to see whether one is contained in the other. When you are searching for a particular piece of data in a large body of text or data, the instring can be a power house. One great loss, is the DEF FN, a mistake by Tandy to omit this statement from level 2 Basic; it's in the disk but it costs money to get it. But you can get round this by substituting the following, at the DEF FN line,

```
line 10  FNx = F0 to F9
line 20  IF N = 0 Go to 40
line 30  RETURN
line 40  N = 1
```

So, whenever the DEF FN is called, then GOSUB to line 10.

In level 1 the CSAVE and CLOAD tape transfer rate was 250 baud, in the level 2 its 500 baud, and so the volume level has to be lower for this faster rate, but it is now Kansas City CUTS, as Kansas City CUTS is 425 to 600. Tandy engineers don't like this, but, one improvement that must be made on the tape recorder is an override switch connected to the remote plug position. The CTR-41 tape recorder is not badly made when you look inside, the mechanics are sturdy and the heads and electronic components are also not bad. I use three cassette recorders on my system, and once set up I have no trouble, but to set up the CTR-41 is the least troublesome. Take good care in choosing your tapes.

The ease of dumping programs one after another is a great boon to me, as during lectures I load the first program under Cload "1", and the program is running while I am talking. Then I clear the program "1" in the computer when ready, and Cload "2" the next program whilst still talking. When I reach the point in my lecture that requires a display of data, I hit the "run enter" for display; it gives the impression that the computer has the answer every time without my help, but in fact the students cannot see the key board, so that they don't see me clear the memory and load the next one in. I usually end on a GOOD-DAY LADIES AND GENTLEMEN AND GOD BLESS in 32 characters per line, and you can gather this type of lecture is very popular with my students.

One thing that I liked in the level 1 was the shorthand dialect, as it saved fingers and memory space. In level 2 you only have three i.e., ' = REM; ? = PRINT; . = CURRENT LINE; but for memory saving it's far better than the level 1, as the programs are faster and take up less space, due to a ROM COMPILER that automatically compresses the incoming data to fewer bytes to about a sixth. This has been done by Tandy for the disk system, but if you don't buy the disk unit, you still get full benefit.

Also, each time you enter a new line it costs you 5 bytes. This can be reduced by writing each line 255 characters long, which is 3½ rows on the screen. The level 2 manual gives eight ways of saving more memory space, and six ways of speeding up the program execution.

You can buy the level 2 with 16k and with an expansion interface fitted, a further 32,768k can be added making 48k in all. Or you can make, like me, a jumbo size S100 system to run 900k!

The Tandy interface will accept the printer or disk system, which has the usual disk commands plus the 9 additional commands, that is missed in the Basic (without the disk) such as ten USR calls, and H hex, and 0 octal, DEF USR, DEF FN, line input, MID$ left, TIME$ and last INSTR (for instring subroutine).

The appendix in the level 2 manual is very good and very important. For a start, you get 14 pages of level 2 summary of command modes, commands, subcommands, in/out statements, field specifiers for print using, program statements, command statements, string functions, math functions and special functions.

There is a page of Basic reserved words, (which I turned into a page index). A page of program limits and a memory overhead is given, plus a page of abbreviated error codes, then two pages of fully explained error messages. The rest of the appendix consists of control codes. This small program allows me to examine any code just by typing the code number:

```
10 FOR C = 0 to 255
20 PRINT CHR $(C),
30 INPUT C
40 NEXT C
```

There are pages of ASCII codes, memory map, video display work sheet, derived functions, and last G/Base conversions. This manual has been put together far better than that for level 1, but it assumes that you know some Basic.

All in all I think level 2, 4k or 16k is well worth the money; 16k gives you a feeling of room to manoeuvre in. It's as powerful as most other standard systems, despite any faults that you could find, as no system is faultless, but at least this sytem is not so hard on the pocket. And level 2 allowed me to write hundreds of programs. I know other systems have better firmware, I am just showing how much better level 2 is over level 1.

. . . . . .

Please note: if you open the keyboard case by removing the center screw that has been sealed, you run the risk of voiding the warranty; the risk is yours.

One experiment that has been tried out with 60% success is to pass programs to your friends, by the GPO phone line. All that's required is first a clear line, then a tape recorder at each end.

The sender must have his volume higher, but watch out for distortion. Then he must cue the other end, to start the recorder and then shut up. The sender will play the program with the mouth piece of the phone receiver close to the speaker on the recorder. Try *not* to hold the receiver, and *do not* place the receiver on the recorder. The best way I find of holding the receiver is to clamp it on to a camera stand, and the other person must do the same, with the ear piece close to the microphone of the recorder. When the program has ended, wait for 10 seconds for the tape to have a space clear.

I've tried this out on long and short programs at distances of 200 miles with no GIGO, one point is to do this at night only, try it out for fun if for nothing else, good luck.

If you don't want to fit a rewind switch to the recorder, then hit the reset button at the back, type "cload" which will switch the tape relay on; then you can rewind back, then hit the reset again, type run, and carry on. This is how Tandy would like you to do it . . .

Please send a SAE if you require the circuit diagram. My address: *P. J. Turner, 14 York Way, Thetford, Norfolk IP24 1EH.*

ADDENDUM: Conversion Cassettes, Level II

The following are brief details concerning Level I to Level II conversions:–

Because of the differences between Level I and Level II, it is *impossible* to make a tape that will do an absolute conversion.

Tandy (Fort Worth) are rushing out an Addendum to the Manual and will later re-word the Manual to include "Conversion Programs" listings that will help the customer to "edit" his Level I tapes for Level II use.

The conversion tape at the moment will do a literal conversion:

The customer must "edit" and add,

1. A DIM A [ ] Statement if A [ ] can go over A [10]
2. THEN Statements in all IF Statements IF THEN was implied.
3. Redo Graphics if wrap around was used.
4. Change commas in print statements to semi-colons if commas were used incorrectly (as often a tab).
5. Input in Level II can only accept a number, not a variable (except as a string).

So if you say Y = 1; Input "Y or N"; A: If A = 1 . . . and answer the input with Y it will not work; therefore you say A$ = "Y or N"; A$: IF A$ = "Y" . . . .

The future Manual will say that this tape cannot fully convert a Level I Program.

*** CURSOR GRAPHICS ***

```
5 CLS
10 X=1 : Y=1
20 B$= INKEY$
21 IF B$<>" " THEN A$= B$
30 IF A$= "L" THEN X = X-1 : GOSUB 1000
35 IF A$= "A" THEN RESET (X,Y) : X = X-1
37 IF X <0 THEN X = 0
40 IF A$= "R" THEN X = X+1 : GOSUB 1000
45 IF A$= "F" THEN RESET (X, Y) : X = X+1
47 IF X >127 THEN X = 127
50 IF A$= "D" THEN Y = Y+1 : GOSUB 1000
55 IF A$= "Z" THEN RESET(X,Y) : Y = Y+1
57 IF Y = >47 THEN Y = 47
58 IF A$= "C" THEN CLS
60 IF A$= "U" THEN Y = Y-1 : GOSUB 1000
65 IF A$= "W" THEN RESET(X,Y) : Y = Y-1
67 IF Y < 0 THEN Y = 0
68 P = POINT (X,Y)
69 SET(X,Y) : FOR F = 1 to 30 : NEXT : RESET(X,Y) :
   IF P= -1 THEN SET(X,Y)
70 GOTO 20
1000 IF X <0 THEN X=0 : RETURN
1001 IF Y <0 THEN Y=0 : RETURN
1002 IF X >127 THEN X = 127 : RETURN
1003 IF Y >47 THEN Y = 47 : RETURN
1010 SET(X,Y) : RETURN
```

INSTRUCTIONS

FOR PRINTING . . .	FOR NON PRINT . . .
L to left	W for up
R to right	Z for down
U to up	A for left
D to down	F for right

A Legion of Entries

ROMAN NUMERALS COMPETITION RESULTS

Sheridan Williams

I am getting an amazing response to the competitions that I have been setting. After only the second competition it is not difficult to see that if the number of entries increases at the same rate, a time limit of two weeks will have to be set to restrict the number of entries. It was during the PCW Show when I had a great many people see me, that I realised that many more people try the competition than actually send in the entries. Many people said that they had tried the problem but not sent in their attempts. This is a pity because I enjoy reading all the entries, and am amazed at the variety of different methods of solution people find. Some people are not even interested in the prize money, and ask for their winnings to be sent to charity.

In this, the second PCW competition, there were approximately 30% more entries than the previous one. Two interesting facts emerge:—

1. There was not a single entry from a girl — a 100% male programming population is difficult to believe, where are all the girls? Please let me hear your reasons.
2. Entries from abroad are few and far between; two from the Continent and two from Eire.

This competition has been very interesting to judge, and I have been astounded by the ingenuity of some of the entries. One of the main points brought home to me is the profusion of BASIC dialects that exist, and also the wide variety of different methods used to solve a fairly simple problem.

The methods used can be divided into the following categories, based on the facilities offered in BASIC:—

1. String functions and string variables used.
2. String arrays, functions, and variables.
3. No string functions or string variables used at all.

Only two entries were received that fit category three; and although I admire their efforts, they were rather long.

One entry in the second category arrived with the statement "the Roman Numerals program would not be possible without the use of string functions".

I will now outline the criteria that I use when judging *any* competition. These criteria are all based on sound reasons which should be obvious with a little thought. The basis for these criteria are that all entries must be possible to compare, regardless of the BASIC interpreter used:—

1. The program should use only statements common to most BASICs; this means that the following statements are inadvisable:
   MAT statements.
   Arrays with more than 2 dimensions.
   Use of AND, OR, NOT inside IF statements.
   Arrays with a zero element.
   Multi-statement lines.
2. The maximum length of one statement is 72 characters. This is important to restrict the use of huge DATA statements, and long complicated mathematical expressions.
3. Programs should be written without the use of files or overlay techniques.
4. From my side, all programs are run (and timed if necessary) on a Research Machines 380Z, running 9K BASIC (BASGF Version 1.3)

As far as this competition is concerned all the entries which did not conform to the above rules have been altered by me in the best way that I see fit.

The winners of both categories are named below; and the winner of the shortest program category was **J. Clark** of Watford who receives the £5 prize for an amazing 6 statement program!

```
10 INPUT A$
20 FOR J=1 TO LEN(A$)
30 X=VAL(MID$("0111344447",VAL(MID$(A$,J,1))+1,1))
40 Y=VAL(MID$("0123212342",VAL(MID$(A$,J,1))+1,1))
50 PRINT MID $(" IIIVIIIXXXLXXXCCCDCCCMMM",1+7*
   (LEN(A $) -J)+X,Y);
60 NEXT J
```

Creditable attempts were received from **D. Gutteridge** and **S. W. Bartlett**. I particularly liked the latter because it didn't use string functions.

```
10 INPUT N
20 FOR I = 1 TO 13
30 READ R$,D
40 IF N<D THEN 70
50 PRINT R$;
60 N=N - D
65 GOTO 40
70 NEXT I
80 DATA M,1000,CM,900,D,500,CD,400,C,100
90 DATA XC,90,L,50,XL,40,X,10,IX,9,V,5,IV,4,I,1
100 END
```

I would like to be able to award a booby prize to the longest program, a staggering 87 statements long, without any REMark statements!

The entries for the second category were few compared with the first category. I received only 10 that were worthy of consideration. I did, however, receive about 10 more from people who thought that the addition of a few REMark statements made their programs fully documented. For those people and to many others whom I have met who do not understand the reason behind supplying documentation I will briefly outline the main items and reasons for documentation.

Documentation should make the program –

(i) easier to understand,
(ii) easier to modify,
(iii) easier to use.

It should contain the following items:–

IDENTIFICATION
Title of program
Name of programmer
Date written
Language used

DESCRIPTION
Index
Purpose of the program
Method of solution
Flowchart (not too detailed, and language independent)
List of variables
Purpose of each variable

TEST DATA AND RESULTS
Suitable test data designed to test the program fully.
Evidence of tests being carried out (i.e. the results).

USER INSTRUCTIONS
Advice on how to run the program
Advice on how to modify the program

This list is by no means exhaustive, but provides hints on some of the more important items to include. All programs should be documented and, in fact, when you buy a program you should also be given the documentation. I wonder how well some of the software advertised in **PCW** is documented, and whether buyers are given documentation too.

The £10 prize for the best documented entry goes to **G. W. Brown** of Ulverston, Cumbria. A very close second was **R. Steele**. Mr. Brown favoured the traditional approach outlined previously; whereas Mr. Steele's concept was to include everything within the program itself, in the form of REMark statements. This is a very good idea as it makes the documentation portable, it goes everywhere that the program goes. It does, however, make the program incredibly long, in fact very likely 4 times as long.

COMPETITION

Write a program in BASIC to produce a successful Knight's tour of a chessboard. Failing success, the program that finds the most amount of moves will win. Efficiency is paramount, and any programs that take longer than 5 minutes (running on a 380Z using 9K BASIC) will be rejected. If it is at all possible please supply evidence of your results; but those without access to a computer may still enter. Judging will be based on a combination of speed and number of program statements (Include REMarks but they will not be counted). **The prize is £10 and entries should be sent within one month of publication to** *S. Williams, 114 Beech Road, St. Albans, Herts AL3 5AU.*

NOTE ON THE COMPETITION

A Knight's tour is one where a Knight starts from anywhere on the chess board and visits in turn each square once, and once only. I do realise that the competition is vaguely defined, this is intentional, as I leave it up to the programmer to interpret the competition as he wishes. I do not want trivial programs where the solution is stored in array or data statement prior to the running of the program.

Statpack

Colin Chatfield

PART 3

Part 3 of this series of statistical programs written in MSI Basic V2.0 includes the most comprehensive program of them all. STAT3 is CHAINed from STAT1 or from STAT2 as are all other programs in the series.

As with the other programs the data which is entered through STAT1, is recovered automatically from the disk upon running. The opportunity to see it is then given so that you can be sure of what the data is.

STAT3 is a composite of information which results in a printout of 12 sets of figures plus the range of numbers in a column. The results are:— sum of items, sum of the squares, the mean, the variance, the standard deviation, the probability of error, the standard deviation of the mean, the coefficient of variation, the lowest & highest numbers and the range, the geometric mean, the harmonic mean and finally the median.

The method used to calculate the results is based on standard principles of statistical calculation and is listed in lines 1400 to 2440. Most of the calculations of a column are carried out between lines 1430 and 1570 and the printout is from line 1580. A 'NO RESULT' will be given for the Geometric and Harmonic means if there is a zero in the column. The median is calculated by the sub-routine at 2220 to 2440 which orders the data and causes a delay in printing the result if there is a lot of data while the ordering takes place.

The recovery, visual and other sub-routines are from line 9000 onwards as they are for all the modules in the series. Part four will follow in a future issue and gives a program for Progressive Averages.

```
LIST     STAT3  FOR  STATPACK   BASIC STATISTICS

WRITTEN BY COLIN CHATFIELD       MICRO-AID  LLOYDS BANK CHAMBERS
                                 CAMBORNE  CORNWALL.   0209-715901

0005 REM STAT3 BASIC AVERAGES
0020 INPUT "      ENTER PORT # ",Z9
0080 LINE= 80
0100 ? TAB(24);"BASIC STATISTICS":?
1010 GOSUB 9600
1020 ? "THANK YOU, YOUR ARRAY IS ";A;"x ";B;CHR$(8);". ";A*B;"ITEMS."
1030 INPUT "      CARRIAGE RETURN WHEN READY",A$
1040 GOSUB 9300: GOSUB1400
1200 GOSUB 9360
1210 IF LEFT$(A$,1)="N"THEN1260
1220 IF LEFT$(A$,1)<>"Y"THEN1200
1230 INPUT "      ENTER `Y' IF BASIC AVERAGES",A$
1240 IF A$="Y"THEN1040
1250 CHAIN STAT1
1260 ? :?TAB(29);"STATPACK END": END
1400 ? #(Z9):GOSUB9380:?#(Z9),TAB(15);"BASIC STATISTICS"
1410 GOSUB 9400: IFB2<1 THENRETURN
1420 A5=0:A7=0:A8=1:H=9.9E-99:L=9.9E+99:N=0:S=0:S2=0
1430 FOR I=1TOA
1440 A4=C(I,B2):IFA4=0THENA7=1:GOTO1445
1443 A5=A5+(1/A4)
1445 N=N+1
1447 IF C(I,B2)>H THENH=C(I,B2)
1450 IF C(I,B2)<L THENL=C(I,B2)
1455 S=S+C(I,B2): S2=S2+C(I,B2)*C(I,B2)
1457 A8=A8*C(I,B2)
1460 NEXT I
1470 V=((N*S2)-(S*S))/N/(N-1)
1475 M=S/N:D=SQR(V)
1480 P=.6745*D: E=SQR(V/N): C=D/M
1560 A6=1/A: A4=1/(A5*A6)
1570 A8=A8^(1/A)
1580 ? #(Z9),"NUMBER","SUM","SUM OF SQUARES"
1590 ? #(Z9),N,S,S2:?#(Z9)
1600 ? #(Z9),"MEAN","VARIANCE","STANDARD DEV."
1610 ? #(Z9),M,V,D: ?#(Z9)
1620 ? #(Z9),"PROB.ERROR","STD DEV MEAN","COEFF VAR."
1630 ? #(Z9),P,E,C
1640 ? #(Z9):?#(Z9),"LOWEST","HIGHEST","RANGE"
1660 ? #(Z9),L,H,H-L
1670 ? #(Z9):?#(Z9),"GEOMETRIC MEAN","HARMONIC MEAN","MEDIAN"
1680 IF A7=1THEN?#(Z9),"NO RESULT","NO RESULT";:GOTO1695
1690 ? #(Z9),A8,A4;
1695 GOSUB 2220:?#(Z9),TAB(33);M
1700 GOSUB 9300:RETURN
2210 REM MEDIAN SUBROUTINE
2220 A4=0:A5=0:N=0
2230 FOR I=1TOA:N=N+1
2240 B3(I)=C(I,B2):NEXTI
2250 REM SORT INTO ORDER
2260 S=0
2270 N=N-1
2280 FOR J=1TON:IFB3(J)<=B3(J+1)THEN2310
2290 A4=B3(J):B3(J)=B3(J+1):B3(J+1)=A4
2300 S=1
2310 NEXT J:IFS=1THEN2260
2400 IF A=INT(A/2)*2THEN2430
2410 M=B3(A/2+1)
2420 GOTO 2440
2430 M=(B3(A/2)+B3(A/2+1))/2
2440 RETURN
9000 REM SUB-ROUTINES
9300 FOR I=1TO5:?#(Z9):NEXTI:RETURN
9360 INPUT "      ENTER `Y' FOR MORE, `N' FOR NONE ",A$: RETURN
9380 ? CHR$(25);:?CHR$(25);:?CHR$(22);:?CHR$(12);:RETURN
9400 IF B=1THENB2=1:GOTO9430
9410 ? :INPUT"      COLUMN # STATISTIC REQUIRED FOR ",B2
9420 IF B2>B THEN?"TOO HIGH";:GOTO9410
9430 ? #(Z9):RETURN
9600 OPEN #10, STATFL1 FOR INPUT: FIELD#10,F=6
9610 OPEN #20, STATFL2 FOR INPUT: FIELD#20,A=6,B=6
9640 SET #10=1:SET#20=1:GET#20
9650 DIM C(A,B),B3(A)
9660 FOR I=1TOA:FORJ=1TOB:GET#10:C(I,J)=F:NEXTJ:NEXTI
9680 INPUT "      ENTER `Y' FOR VISUAL OF DATA",X$:IFX$<>"Y"THEN9690
9685 ? :FORI=1TOA:FORJ=1TOB:?C(I,J);:NEXTJ:?:NEXTI:?
9690 CLOSE #10:CLOSE#20:RETURN

THE GHOST
#


        CHAINSTAT3
            ENTER PORT # ? 1
                                  BASIC STATISTICS

            ENTER `Y' FOR VISUAL OF DATA? Y

        3 2 0 0 2 3 1 1 50 5 0 0 0 1 0 0 0 0 0
        3 1 0 0 2 4 1 1 100 2 0 2 0 0 0 0 0 1 0
        3 1 0 1 0 1 1 1 100 3 3 2 0 1 0 1 0 1 1
        3 1 0 1 0 2 1 1 2 1 0 2 0 0 0 1 0 1 0
        3 1 0 0 2 3 1 3 2 1 0 0 0 0 1 1 0 0 0
        3 1 1 0 0 2 1 1 50 2 0 0 0 0 1 1 0 1 0

        THANK YOU, YOUR ARRAY IS 6 x 19. 114 ITEMS.
            CARRIAGE RETURN WHEN READY?


                        BASIC STATISTICS

            COLUMN # STATISTIC REQUIRED FOR ? 6

        NUMBER          SUM              SUM OF SQUARES
        6               15               43

        MEAN            VARIANCE         STANDARD DEV.
        2.5             1.1              1.048809

        PROB.ERROR      STD DEV MEAN     COEFF VAR.
        0.70742167      0.4281744        0.4195236

        LOWEST          HIGHEST          RANGE
        1               4                3

        GEOMETRIC MEAN  HARMONIC MEAN    MEDIAN
        2.289428        2.05714288       2.5



            ENTER `Y' FOR MORE, `N' FOR NONE ? N

                                  STATPACK END

        THE GHOST
        #
```

PART 4 *(Refer to p.50 of December issue for listing).*

Part 4 of this series of statistical programs written in MSI Basic V2.0 deals with one item only, STAT5, which is Progressive Averages. As with the other modules it can be called independently or CHAINed from STAT1 or STAT2.

The data is recovered automatically from the disc as usual and can be looked at for verification if desired. This program is relatively simple and is designed to show the mean of a column of data as each item is added to the sum of those already dealt with. The item number, its value, the cumulative total and the mean of the cumulative total is printed out in tabular form. This is done by lines 3325 to 3370.

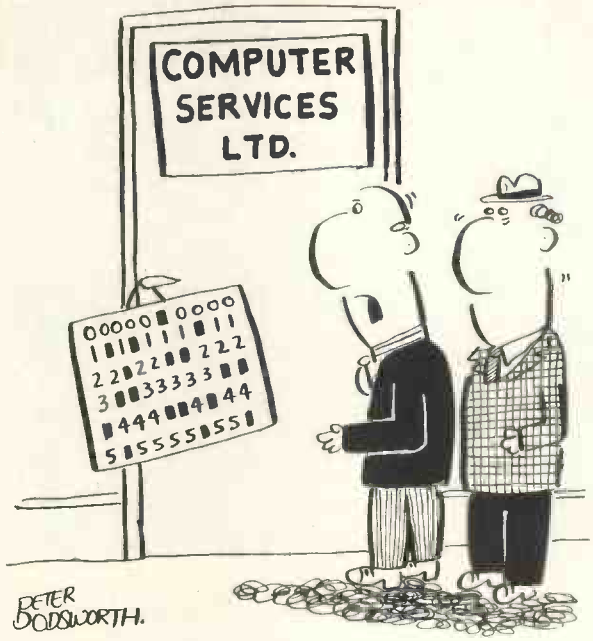
Lines 3380 to 3600 are added in and show in pictorial form the same results. This section could be used in other ways as a program of its own. It is interesting as it shows in columnar form the results which are usually printed in horizontal form in most computer printouts. The highest and lowest numbers of the column are found at lines 3430 to 3460. 12 lines are used in the printout, which fits most screens, and as many columns in width as there are data items. Down the left side a series of numbers are printed as a scale in twelfths of the highest average. This is done by line 3500. The FOR/NEXT loop from line 3540 to 3590 then travels across the screen or page and prints a "*" if the average is equal to or greater than the value on that line for each column. If the number is minimal in comparison to the lowest line, but above zero, a '.' will be printed.

The recovery, visual and other sub-routines are from line 9000 onwards as usual in the series. Part five will follow in a future issue and gives a program for Chi-Square.

Colin Chatfield writes:

We have exciting news in that we applied for a Manpower Services Commission S.T.E.P. program and have been granted £2,557.00 for one year to employ a man or woman to investigate the practical use of a micro-computer in our holiday business. We already have a SWTPC 24K RAM with 8" Floppy disc and ASR43 printer setup which has been in use for nearly a year. This opportunity will be given to any person who has been unemployed for at least six months and is prepared to work in Camborne and has a basic knowledge of BASIC and preferably 6800 assembler. The selected employee will be able to write his own programs, document them and look after a complete system.

Our Address:
Micro-Aid, Lloyds Bank Chambers, Camborne, Cornwall



"Gone to lunch, back at one-thirty."

GIVE HIM A MICKIE

Hugh Price
Consultant Physician, Charing Cross Hospital

Does the doctor really need a computer or would he prefer to be able to sink back into and stay in the past?

Could technology on the other hand drag the Health Service screaming into the 20th Century in spite of the fears of doctors, the majority of whom up to now have been dragging their feet? They need help urgently.

All the help the doctor needs, at the touch of a button, says the computer salesman, demonstrating his special problem orientated system. And if so, why is the doctor still resisting it? There are two main areas of conflict: the tremendous pressures caused by the headlong advances in medical science since the war that both the doctor and his patients hear about — "Tomorrow's World" — but seldom see, and a greatly increased population at risk.

The poor facilities for health care came to a head with the NHS in 1948 when Pandora's Box was opened and it became impossible to halt the rush on the nearest Health Unit. This rush has never been absorbed but has been aggravated by inflation, so that the advantages of modern medicine are spread very thinly over the country, even more thinly than the margarine on hospital canteen bread. Alas, we got the placebo, but never the treatment.

Can we accomplish the original objective of the NHS without spending the money that the inhabitants of the corridors of power have developed ingenious ways of losing?

What goes on?

Observing the implementation of NHS, medicine, as practised at the dim end of the spectrum where the average consultation time is 2½ minutes, what goes on? The interview between many doctors and the inmates of the overcrowded outpatients and the G.P's surgery is almost computerised *without* a computer. It goes something like this:

The doctor says:		The computer would like to say:
What's your name	—	log in and identification
What's wrong with you	—	objective data
Had it before What tablets did you have then	—	functional historical data
Do you want the same again	—	data input drugs
Do you want a certificate	—	data input social
Next patient	—	log out

With the average consultation time being so short there is no time to develop the famous doctor-patient relationship (in spite of BMA braying), and even less for recording. The result: hurried supermarket medicine, over prescribing, excessive use of antibiotics and tranquillisers.

What is required is instant help when needed. Help in this context means communications. The largely medieval communications are still accepted because nobody has yet demonstrated a practical and economic alternative. What is badly needed today is a strategy to reintroduce computers to doctors in a simple, easy and acceptable mode, without depriving them of their status.

The doctor should record his activities for clinical decisions whenever the patient is seen. Today, with a larger population, this function becomes boring, repetitive, and time consuming with aged and chronically sick attending more frequently.

Let us develop an information system from where the patient first encounters the doctor and starts the diagnostic decision pathway with the patient's record and from there to vital information about diagnosis and treatment that he can (if necessary) obtain from other doctors.

The history of the patient's condition should provide guidelines; or the cybernetics for the next section of the consultation, which will be the physical and technological investigations and the results. Prior to the last war there was little other than a bedside manner and good nursing care. The laboratory investigations were comparatively few, even for those patients who could afford to pay; the others had to rely on natural selection as their only hope. So in those halcyon days the patient's notes at least had the merit of being simple and clearcut.

The Provisional Diagnosis

As a result of these preliminary activities the doctor sets up a hypothesis or value judgement — the Provisional Diagnosis. This hypothesis he validates as far as possible by physical examination and test procedures, occasionally by the response to treatment. He should record the signs of biological system failure which enable him to recognise its possible cause, for the whole process of illness may be due to the individual coming into contact with infection, pollution, violence, stress or any other reason. The medical record must be current and mirror any such changes. But the doctor requires not only information about the history of the present illness but information on back-up facilities for his patient, as well as contact with the latest advances in medicine. He needs:

1. Help in routine patient interrogation so that he has time to discuss special aspects and to probe the validity of the patient's responses.
2. Data information about the patient's past, family, social and clinical history.
3. To be assisted in recording all the positive findings of the consultation.
4. To order the test procedures and record the positive results.
5. Facilities for rapid recall of all this data and the facility to validate, monitor and update.

All this information is essential to the clinical function and, though vital, the collection, collation and recording process is very time consuming. But the medical record is the essential source document for medical and administrative care and a modern method of generating it must be found. For with the rise of modern medical science, as epitomised by the autoanalyser, the test results extend to several pages of printout which often accounts for two thirds of the casefolder. The doctor has to find time to examine this plethora of data just in case one or two results are abnormal.

Only about 1% of all practising doctors are sufficiently motivated to be prepared to carry out the input routines required by all existing computers and especially the large main frame systems that have been developed in hospitals. The evidence for this is seen by the absence of a computer clinical record, in any of the world's hospitals, being used for routine medical care. There are also several examples of expensive failures scattered over the medical scene where the large computer system has dropped the medical record, and is used instead for purely hospital administration and laboratory back-up services. How can we seduce the 99% to these new and unusual methods of recording? For most doctors do not approve of the additional, and what they consider unnecessary, tasks involved in the various methods of inputting data to the computer which have confronted them in the past. So the large hospital computer retreats from practical medicine and becomes part of the administration, the housekeeping and the laboratory, a purely back-up function using the computer as an expensive desk calculator.

Too busy to be efficient

Another objection doctors have to the computer is that inputting data takes up more time than the traditional pen and paper method, is quite different from the standard method of recording; so many steer clear of this facility. Many systems also require data to be input that are of little or no use to the routine clinician but is thought up by the administrators, computer boffins or the statisticians.

And another point, why concentrate on storing all one's patients' problems in one vast data bank, with all its management and confidentiality problems?

A terminal operator or clinical secretary to input the data from manual notes means an additional chore after the end of the clinical session or General Practice Surgery, and the inclusion of another individual in the information pathway increases the chances of error and loss of confidentiality and, of course, the cost.

So we must provide doctors with the simplest of recording systems which can be expanded as required.

However, is it really so difficult and are doctors so exceptionally resistant to change? The results of a feasibility study showed they are not, if they are included in all discussion stages and state their own special requirements. In 1973 in our medical unit we completed a study on the possibility of creating a keyword medical record using remote job entry (RJE) in a batch mode, this could be input to a computer (IBM 360/155). This feasibility project was designed for a terminal operator to input the keyword data filled in by the doctor on a card during the consultation, a printed summary being available for updating and validation when required.

In the feasibility study retrospective surveys were carried out on case records and it was discovered that some doctors wrote an essay, only of use in the waste paper bin, others recorded impressions, moral judgements and irrelevant items, some only a few words. A few, however, recorded just key function words and this technique formed the basis of the newly designed computer record. So first encourage them to record just these few words, and make it easy for them to do this, changing their mode of recording as little as possible. How was this done? The general practitioners and hospital doctors involved (12 in all) designed *their own* keyword summaries for their routine clinical use. These were mutually accepted and used routinely. This summary record is still being used in one of the general practices which was involved in the study. Any new automated record must be very simple, easy to use and non time consuming if it is to interest the 64,000 practicing doctors in this country.

Keyword histories

Wherever it is carried out, the first part of the consultation is the history taking procedure. A computer terminal is being used to do this in the Department of Medicine at Charing Cross Hospital. It interrogates patients and provides a printed summary or keyword history at the end of an interview which may last 40 minutes. The patients like it and the doctors are now coming to realise that the computer is an ally, picking up all the relevant information. It has alerted them on several occasions to other important symptoms. This application has special use in the follow-up of chronic illness, where the interview tends to be prolonged, monotonous and repetitive. It is now being employed in the investigation of environmental and occupational disease, for no doctor can possibly remember to ask about all the occupations and processes and associated disease; but the computer can, and does. In these circumstances the computer is complementing not replacing the doctor.

Computers as cybernetic tools could revitalise communications, provide opportunities for doctors who are capable clinicians and interested in the changing pattern of medicine, to move with the times, without disturbing their megalithic colleagues who are more comfortable with their Stone Age activities. Let's move data not people. Leave the doctor with his individual craft skill, and instead of linking him expensively to a large data bank where he is forced to do what everyone else does, disperse the computer facility *to* these highly and expensively trained people. In other words use distributed processing with personal computers attached to the problem solving doctor, in the mode he prefers. This is now feasible and possible but also economically viable; for microcomputers do not require vast capital outlay. MICKIE could provide the answer. Such microprocessors, and especially the concept of the personal computer, could provide each doctor with his own individual information processor linked to those of his colleagues only by the patient's unique number code, and the programming, help to overcome the confidentiality problem. He can carry all his records in his briefcase. If he agrees, his MICKIE can be physically linked to others in the practice or hospital, and the information required for other consultations can be released with his consent. Data for management and statistics can be spun off without revealing the patient's identity.

The price of MICKIE and its successors is already within the means of every doctor. The individual patient record can be recreated in a new electronic form out of the chaos of the hospital and practice record departments. Could we look forward to the running down of the hospital record and computer departments in the future, with regional computers collecting the necessary central statistics?

The heart of the matter

The essence of the new revolution in computing is the appreciation of the potential of this new medium of communication on the part of the user and the vast changes and benefits which it will bring, recording function directly as the activity is carried out, rather than having to provide data that describes function manually. To do this it will be necessary to consider function in a different way from the traditional pen and paper method.

But will the doctor still need to use the same input routine that has been the curse of the large system? For there will still remain those doctors who will find even the simplified input routine required for MICKIE too laborious, time consuming or difficult for them. Voice input *could provide the answer to both of these problems.*

This new procedure was the subject of a feasibility study in 1976 in our chest unit, in conjunction with the National Physical Laboratory, using patients' replies to an automated interrogation system. In this experiment the computer obtained a 74% acceptance rate to patients' replies.

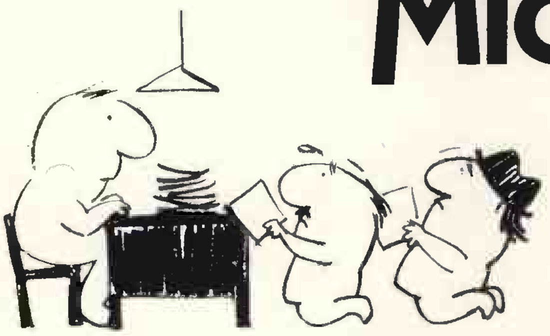
We are currently preparing for a trial with voice input for the *doctor himself* to create a keyword summary record at the same time as he is carrying out the consultation which can be merged subsequently with the results of the physical examination and the test procedures by the microcomputer. We shall be comparing this computer generated summary record with that generated at the same time with a tape recorder that will be transcribed, and with the traditional pen and ink generated case record.

At the present moment we are merely on the threshold of medical computing. Let's get over this hurdle of recording, with more MICKIES for the doctors and then really get going.

PCW *This is a companion article to the one by Nigel Bevan in the last issue.* PCW

Softly... Softly...

Selling Microcomputer Software



Julian Allason


Julian Allason, Software Director of PETSOFT, with his wife Jessica, who is also one of the partners in the business, and PET. Photo: David White, Daily Mail.

Once the initial excitement of your new microcomputer has worn off, you will probably find yourself wanting to get down to some serious programming. There are several ways in which you could profit from it.

One is to write programs for distribution and sale by a microsoftware house; there are several in existence. Seek guidance from them on the sort of programs that will sell well; they will be only too pleased to advise you on the preferred format.

You might expect a royalty of around 7½ to 10% of the recommended retail price on relatively straight forward programs; perhaps more on business software packages. This may not sound a great deal but the publisher will advertise, manufacture and distribute the program to dealers who typically receive a 40% discount off the retail price. Good programs should go on selling for a considerable period of time.

Alternatively you may be offered a flat sum for outright purchase of the rights to your program. This is a sure sign that the program is expected to sell well, and unless the offer is substantial, you will probably do better to demand a royalty.

The second method has become known as the 'Lemonade Computer Service' in the United States. Essentially it involves setting yourself up as a (very) small scale localised computer bureau. Many experienced observers feel that this is the route of the future, with hundreds of micro-bureaus dotted around the country, exchanging software and providing a useful service to their communities.

The great advantage of running your own Lemonade Service is that you can work when you want, with whom you want. Payrolls, VAT Book-keeping, Statistical Analysis, Mailing Lists — these are just some of the jobs which local traders may ask you to undertake. One point to bear in mind: confidentiality is everything. You should never disclose anything learned, whether of a commercial or personal nature, in the course of your computing.

The third and most difficult course is to open your own micro software house and sell to fellow computer users. Here the pitfalls are legion. You must be adept at programming — your customers certainly will be — and you need to know about business. Many one-man software outfits in the United States ran into trouble through over-expansion. If you think you can run a service of this sort in the evenings and at weekends, you are almost certainly mistaken. And without sufficient capital, it may not be a good idea to throw up your job.

Still interested? The first problem will be the software itself. You can't copy someone else's, unless you plan an appearance in the law courts and a secluded vacation afterwards, so you will either have to write it or buy it in. In any event it must be good. The competition is hot, and poorly written business programs invite negligence suits. We use a panel of programmers and computer users to screen all new software for possible copyright infringements, bugs and the other creepy crawlies that inhabit the software demi-monde.

It is a good idea to provide as much documentation as possible. Free data sheets for prospective customers will be essential. You will probably be concentrating on standardised programs so be sure to include plenty of REM statements so that customers can follow the construction of the programs and make alterations as necessary.

Mail order regulations are strict in Britain. So be prepared to process orders fast, and offer a no-quibble money back guarantee. The credit card companies do not regard firms who have been trading for less than six months, with much enthusiasm.

When you come to costing software there are several hidden extras to include. Inevitably, bugs will be discovered after numerous copies of a program have been sold. You are going to have to mail all those customers an update, alerting them to the problem and telling them how to implement a software patch. You may even have to replace the whole program. This can prove both expensive and time consuming.

Bear in mind that quite a lot of time will be spent answering technical enquiries, by letter and on the telephone. Include the cost of your time in calculating the overheads. Once you are overwhelmed with work, you will probably have to pay someone to take on the more routine administrative tasks. The costs soon mount up.

Fortunately piracy is not (yet) a very great problem in the UK. Our copyright laws appear to offer better protection than those of the United States. Nevertheless it may happen, so be prepared to serve a writ – fast. It is advisable to consult a solicitor specialising in this field, who will advise you of the form of words to be used in copyright notices. You may also consider adopting our practice of including serial numbers, both overt and covert, in the software itself. If there is an outbreak of piracy you will at least have an indication as to the source.

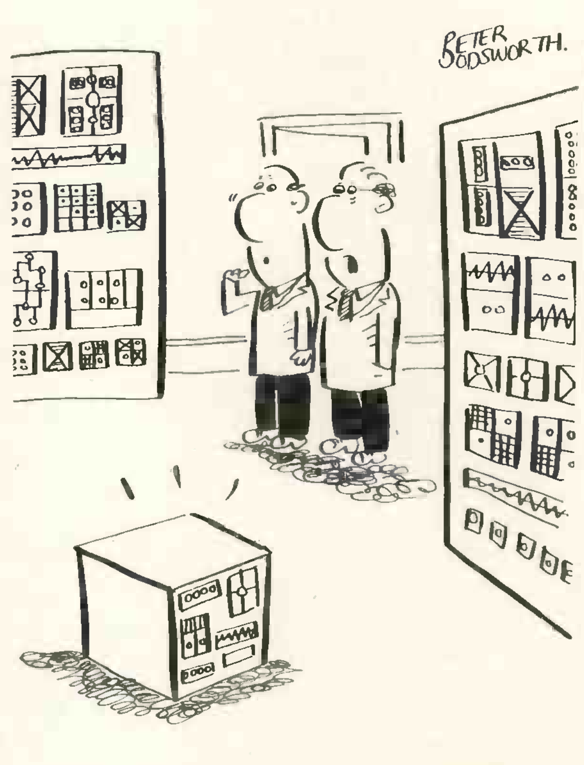
Publicity is essential, and the best possible medium is the newsletter published by the user group for your particular computer. The magazines will also be interested in your progress. And of course you can always advertise.

It is important to recognise the limitations both of the hardware itself, and the service that a small organisation can offer. It seems likely that in the future there will be a growing divergence between micro software houses specialising in certain subjects, such as hotels and catering, and those supplying a general purpose list of games, tutorial and business software. Although large numbers of games have been sold, recent trends point towards a rapid growth in small business software.

Traditionally, software costs have exceeded that of the hardware itself. As we all know, micros are now relatively inexpensive. But there has been no corresponding decrease in software prices, programming being one of the most labour intensive activities imaginable.

The answer must lie in standardisation. To a large extent this can be achieved through the disciplined use of high level languages, such as BASIC and COBOL. Programs must be designed in such a way as to facilitate alterations. It is certainly a great deal cheaper to alter an existing General Ledger package, for example, than to write it from scratch. Typically this might involve some alterations to a client's own accounting procedures; but in many cases this will be a small price to pay, when the alternative is no computerisation at all.

PCW Julian Allason is a contributor to a book on personal computing to be published by "Input Two Nine" and edited by Martin Banks of *Computer Weekly*. PCW



"Don't ask me, it wasn't there when I left last night."

WHERE THE LAW BITES

An occasional look at the legal situation in the computer world

Brian W. Haines.

What does the law have to say about computers? Precious little; and that makes the topic extremely interesting for here is a whole new area of activity that is of premier importance and has no specific legislation to govern it. So the sort of questions most commonly asked like 'Is it possible to own a computer program' have to be answered with an evasive 'Well it rather depends'.

The reasons for this murky situation aren't hard to find, computer technology and its' applications is an advancing technology rapidly outstripping the capacity of many people to assimilate it as part of their daily lives. This includes the legislators themselves. Our law deals in the main with human relationships and with situations after the event, so it always lags behind. It would be expecting too much to ask Parliament to anticipate developments before they have happened. The best that can be done is to adapt existing legislation to the new situation; and when it becomes unworkable, a new law is proposed.

The situation is now giving cause for concern especially in respect of the 'software'. Proposals are being put forward by various bodies around the world to standardise the situation, meanwhile we in Britain have to struggle on with an adaptation of the existing law, and that calls for a fascinating juggle of concepts.

The computer itself causes little difficulty; it is after all, only a machine. As such it comes well within the scope of the present law.

The problems start to manifest themselves when output and input are considered. Suppose for instance that music or some abstract design has been created by the computer by a process of random selection, so that it is the computer and not a directing human agency that has been the source of creation, and then that these creations have become capable of great commercial exploitation. In other words they can be sold at high profit. How can they be protected against unauthorised copying? One would imagine the owner of the machine, or the programmer, or perhaps the operator at that time should be the owner of the particular creation. The trouble is they may all be different people, a friend may have borrowed the installation for the evening or something of that nature. But even harder to solve than that: there may not be any protection available anyway because the creation is not by a living person. No-one knows until some lawyer is paid to present such a case in a court which will decide between the opposing views.

The same sort of difficulties arise in respect of the program itself. No-one needs to be reminded of the amount of mental work that goes into the creation of a program. The enthusiast is generally quite happy to swap ideas with interested enquirers. By the nature of things there will come a time when a particular program or a language perhaps is commercially viable. By coincidence exactly the same thing has been created by a commercial institution who would want to protect it; or what may be worse, they have taken advantage of shared information. So, can an individual take and make use of a published program, or borrow from a friend and sell it or is there any form of legal protection available?

In the past certain types of program have been given patent protection. A patent however suffers from the peculiar drawback that until it is challenged in the High Court no-one can be sure if it is really a valid patent or not. The patent is also of limited duration with an increasingly expensive fee being charged for each year. With the new Patent Act it is possible that this form of protection will not be available as it excludes mathematical systems as subjects of protection. This followed from the recommendation in the earlier 'Report on the Patent System' that programs should not be patentable.

The other alternative is copyright protection. This is a different concept and arises out of the Copyright Act and was originally designed to protect artistic and literary endeavour. This is a right that arises automatically upon creation and lasts a lifetime plus fifty years. The big difference between a Copyright and a Patent is that more than one person can hold a copyright in identical pieces of work if each creator arrived at the original work by his or her effort. The one important restriction is that there is no copyright in an idea alone, it must be set down and there must be some element of creative effort.

The problem is to establish if all programs, which are without doubt the product of intellectual creation, are more than mere ideas and are also creative; and further, are they capable of copyright protection. The Act lays down the specific areas covered and the people called authors in the Act, who are protected. Computer programs are not listed amongst the areas as they did not exist as a topic of sufficient importance when the Act was drafted.

Most informed opinion believes, and there is little case law to go on, that programs can be brought within the ambit of the Act by taking little sections of the stated areas. Even so such protection is fairly minimal although the Courts have recently extended the remedies available. Because of this uncertainty and the great importance programs are likely to achieve, anyone who dreams up something really beautiful should be careful in parting with information about the dream child. The best protection is of course to show it to no-one; and if that is not practical then get every-one who sees it to agree they do not divulge its' contents except by express permission. This brings it under the head of contract and breaches of confidence which is a well tried area of the law.

A NOTE TO CONTRIBUTORS

Some Manuscripts coming to us have been of admirable content and execrable appearance. We do not have a cryptographer in the office. Please be clean. Circuit diagrams must be absolutely legible. Listings must be crisp.

And don't forget — we welcome contributions.

ON THE LINE

David Hebditch

1. TELECOMMUTING

"The city is obsolete: ask the computer". *Marshall McLuhan*

The increasingly widespread use of personal computers in the home and in the office is an exciting prospect. The advance of telecommunications technology will probably have just as significant an impact on our way of life. The symbiosis of the two could well be the realization of McLuhan's concept of the 'global village'. This series of articles is about the practical technicalities and socio-political realities of the Personal Computer Global Village. (Note for Editor: new name for the magazine?)

I think it was Murray Laver (former head of data processing at the Post Office) who coined the expression *telecommuting*. If he didn't, he might well have done. Traditional forms of commuting are time-wasting and soul-destroying, especially on those Southern Region cattle-trucks. They are also very inefficient in their use of energy when compared with telecommuting, the fundamental idea of which is that you take the information to the human being rather than the other way around. People have measured it; in spite of the high set-up cost, it is 'no contest'.

To implement telecommuting, you need to be able

- to move large amounts of data from place to place very quickly
- to manipulate data in a sophisticated way at or near your own home
- to support advanced communications techniques such as videophone, high-speed facsimile and teleconferencing
- to restructure administrative organizations
- to overcome social inhibitions 'going' to work
- and so on.

Clearly, there are prohibitive problems in achieving this effectively using the present generation of telecommunications technology, but in the next article we shall review the likely advances in this field over the next couple of decades.

I strongly believe that telecommuting is a development which should be planned for now at a social and political level. It is an area about which there is great confusion amongst our political 'leaders'. After the last war (I am told) great sums were invested in developing New Towns and city-dwellers (particularly London-dwellers) were induced to Go North! Well, at least as far as Welwyn. In more recent years we have had the proposed or actual relocation of government or quasi-government departments to places like South Wales and Scotland. Nobody seems happy with this state of affairs; the new towns are described as social deserts, the old cities suffer from planning blight and the Government decides that it might have overdone things and announces a programme of investment in the inner cities to persuade people to come back again.

A newspaper reporter once asked Henry Ford:

"What shall we do about the problem of the car and the city, Mr. Ford?"

"Ban the city!" was the sharp reply.

No, I am not one of those who think we can actually get rid of our great cities (even if we would want to), but we can certainly try to stop them becoming more like New York, Los Angeles or Birmingham. Everyone wants to live somewhere nice. Can we work somewhere nice as well?

Most people who work in city centres work in offices (closely followed by shopworkers). It is this kind of job, data handling, that would seem to lend itself most of all to telecommuting. OK, I know you also deal with people, but that is mostly by mail or telephone and can be done from anywhere.

So that is the scenario; will tomorrow's 'automated office' eventually consist of powerful personal computers installed in employees' homes and inter-connected by a high-capacity national (and worldwide) communications network designed for handling data, facsimile and video transmission as well as lowly speech?

I venture to suggest that the technology for this is with us right now. Two things tend to inhibit any rapid progress. Firstly the cumbersome and unimaginative nature of the Post Office management/union infrastructure (more of which, next month), and secondly the crass inability of governments over the years to get their priorities right on this kind of long-term planning. If the money spent on Concorde had been spent on the development of our telephone network everyone would have benefitted instead of being financially penalized. British Airways would have been quite pleased too.

So there we are. Those are some preliminary ideas for you. But all solutions generate new problems of their own and we should keep in mind some of the possible social impacts of telecommuting as far as we can anticipate them. I would not want anybody to be accused of starting a 'telecommuting blight'.

Next: Developments in Telecommunications.

PCW *This is the first of a series of wide-ranging articles on the telecommunications aspects of personal computing. Subsequent articles will include: — trends in telecommunications — how to transmit data on the telephone network — a program for data transmission — protocols for personal computer 'hamming' — prospects for teletext, viewdata and cable television — long distance game playing — special equipment offers . . . and much more.* **PCW.**

TEXT PROCESSING DEMONSTRATED ON THE NASCOM 1

How the Microcomputer can help improve efficiency at home and in the office

S. D. DANTON

PCW We approached the author at our Show and asked him to write for us. Result - first appearance in print of text processing work being done for the personal computer by *ICL Dataskil*. **PCW.**

A letter editing system using a Nascom 1 microcomputer and a Centronics Micro 1 line printer was a feature of the Personal Computer World Show. The system was demonstrated by Dataskil, the Reading based software house, whose appearance on the Nascom Microcomputers stand was at the invitation of Nascom's managing director Kerr Borland.

Dataskil has become increasingly active on the microcomputer software scene, though this was the first time the company has been present at such an exhibition. The company has a team dedicated to microcomputer software development and works closely with manufacturers and potential users of microcomputer systems.

In the course of this work, Dataskil has developed a generalised software framework designed to aid system implementation. It was to show how some of these facilities may be used that members of the Dataskil team prepared the simple letter writer system that came to be demonstrated at the Personal Computer World Show.

This demonstration had special merit in being readily understood by the general public. The software was produced specifically for demonstration purposes, though a version could be developed for a specific customer requirement.

Soft Texture

The facilities of the system demonstrated provide for the creation of text, storage of the text onto standard audio cassettes, retrieval of text from the cassettes for re-use or editing and the printing of text on the line printer.

Examples of its capabilities, as shown on the Nascom stand, included pro-formas (typically purchase orders, invoices and payment reminders), thus showing the private owner and the small business man the possible aids which the microcomputer can provide to improve office efficiency.

The proof of the success of the demonstration was the barrage of ideas that came from the visitors to the stand. People from all walks of life associated with the possible ways of using the letter writer to improve their "housekeeping" efficiency at home and at work. Ideas flowed and included the maintenance of medical or dental records, household budgets, school reports, salary and wage records and in fact applications of all sorts. The idea of having a system of this power for less than the cost of an office typewriter appealed to many a businessman.

The letter editing system was designed to be easy to use. The screen is organised into a text area of 13 lines, a status/command line and 2 other lines used as line buffers. The top line, the status/command line, gives the user information about the current machine status and is used in a conversational manner with the user.

Systematic Prompting

When awaiting a task the top line displays a request to the user:—

SAVE, PRINT, AMEND, EDIT OR CREATE? TYPE 1ST CHAR

To respond, the user keys the first character of the function he requires.

The SAVE function is used to record the contents of the letter writer's text buffer onto a cassette. After the user has keyed S the letter writer responds by displaying on the top line:—

TAPE READY? Y/N

This gives the user time to set up the cassette recorder. When ready the user types Y and the text is saved onto the cassette. When completed the request message is displayed again.

Similarly if PRINT is requested the letter writer responds by displaying on the top line the message:—

PRINTER READY? Y/N

The user then checks that the printer is in fact ready, and if it is types Y and the text is printed. Again, when completed the request message is displayed.

If AMEND is requested the first 13 lines of the text in the buffer are displayed. The entire text may be edited using the standard facilities described below.

If EDIT is requested the letter writer responds by displaying on the top line:—

TAPE READY? Y/N

The user then places the required cassette in the cassette recorder and when ready types Y and plays the cassette.

Once all the data has been read in from the cassette, the response display on the top line is:—

KEY Y WHEN TAPE OFF

On the user keying Y the first 13 lines of text are displayed and the entire text may be edited using the facilities described below. If, however, the automatic read checks built into the letter writer detect an error, the top line will display the message:—

FAIL ? RETRY? Y/N

The user may now repeat the process by reloading the cassette and typing Y.

If CREATE is requested the screen is left blank and the editing facilities described below can be used to create text.

If in any of the cases a Y response was requested and was not received, the complete request line is displayed giving the user an alternative choice.

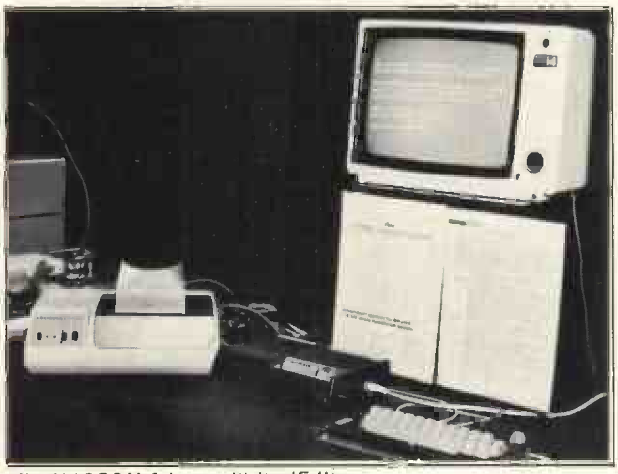
Example

As an example, a typical editing cycle could be as follows:—

A pro-forma is prepared using CREATE. Once it is completed it is printed using PRINT and if any errors are apparent in the text it is recalled using AMEND so that it may be corrected. Once the pro-forma is satisfactory it is saved on a cassette using SAVE, its position on the cassette being noted (thus a standard cassette may be used for many such pro-formas or letters). Some time later the same pro-forma is read back into the micro-computer using EDIT. The pro-forma is filled as required and when it is ready may be printed using PRINT. The completed pro-forma may now be stored on another cassette for purposes of record keeping or it may even be further amended for other purposes.

When in text editing mode the user has the choice of several control functions as well as two modes of data entry; the control functions provided allow the cursor to be positioned anywhere within the text area. The cursor is the delete symbol and is non destructive, merely replacing the character in that position.

Character and word deletion are only invoked by the request of the user. The control functions implemented on the demonstration letter writer are set out below. There being no control key on the Nascom 1 keyboard, all control functions except NEW LINE and BACK-SPACE are invoked by keying @ followed by an alphabetic character. COMMAND is displayed on the top line as an aid to the user on @ being keyed.



The NASCOM-1 Letter Writer/Editor

LETTER EDITOR DEMONSTRATION – Control Functions

NEW LINE – Move the cursor to the start of the next line of text. If the cursor is already on the bottom line of the text area create a new blank line scrolling the original text up by one line.

BACKSPACE – Move the cursor back one character within the text. If the cursor is already at the first position of a line then position the cursor at the last position of the previous line. If the cursor is on the top line of the text area the bottom line of the text area is treated as the previous line.

@ THEN A – ADVANCE CURSOR. Move the cursor forward one character position within the text. If the cursor is already at the last position of a line then position the cursor at the first position of the next line. If the cursor is on the bottom line of the text area the top line of the text area is treated as the next line.

@ THEN B – BACKSPACE WORD. Move the cursor to the first character of the current word. If the cursor is already at the first position of a word then position the cursor at the first position of the word previous to it in the text. If required the search for the word will proceed from the top of the text area to the bottom of the text area.

@ THEN C – DELETE CHARACTER. The character at the cursor position is deleted; the text to the right of the cursor within the line is moved one position left thus replacing the deleted character.

@ THEN D – DELETE LINE. The line in which the cursor lies is deleted. Those lines below the cursor are scrolled up by one line to replace the deleted line. If in Edit or Amend mode the bottom line of the text area may receive further text if any remains in the input buffer; otherwise it is left blank.

@ THEN E – END EDIT FUNCTION. The screen is scrolled until all the text on the screen and the input buffer is stored in the output buffer; the prompt line is then displayed on the status/command line requesting the user to select the next machine function.

@ THEN H – HOME CURSOR. The cursor is positioned at the first position of the text area i.e. at the first column of the first line of text.

@ THEN I – INSERT LINE. All the text in the cursor line and those above it are scrolled up by one line leaving a new blank line at the cursor position.

@ THEN L – SCROLL UP. All lines are scrolled up by one line. As in all the cases where a scroll causes a line of text to be removed from the top line of the text area, this information is stored inthe output buffer. The bottom line of the text area may receive further text if in Edit or amend mode; otherwise it is left blank.

@ THEN M – TOGGLE ENTRY MODE. Character entry takes place in either OVERWRITE or INSERT mode. In Overwrite mode the character at the cursor position is replaced by the keyed character. In Insert mode the character at the cursor position and all those characters to the right of the cursor in that line are displaced one position to the right, the character keyed being placed in the gap so created. In either mode the cursor is advanced by one character position within the text area after the character keyed has been displayed. If the cursor was at the last position of a line it is positioned at the first position of the next line. If the cursor was at the last position of the text area its position remains unchanged.

@ THEN N – FORWARDSPACE WORD. Move the cursor to the first character of the next word. If required the search for the word will proceed from the bottom of the text area to the top of the text area.

@ THEN R – DELETE TO END OF LINE. All the characters from the cursor position to the end of the current line are replaced by spaces.

@ THEN S – SCROLL SCREEN. All the text within the screen is scrolled up 13 lines thus placing the contents of the screen into the output buffer. If in Edit or Create modes and text remains in the input buffer, the screen will now contain new text, otherwise the screen is left blank.

@ THEN T – TAB. The cursor is moved right within the line to the next tab position on the line, the tab positions being every 8 characters on the line. If the cursor is already beyond the last tab position of the line no cursor movement takes place.

@ THEN U – CURSOR UP. The cursor is moved up by one line. If the cursor is already on the top line it is repositioned on the bottom line.

@ THEN V – CURSOR DOWN. The cursor is moved down by one line. If the cursor is already on the bottom line it is repositioned on the top line.

@ THEN W – DELETE WORD. All the characters from the current cursor position to the end of the current word are deleted, the text remains in the current line moving left to replace the deleted characters.

All control functions were implemented using a subset of a system software suite designed and developed by micro-specialists in Dataskil's Special and High Volume products department. This suite of programs was designed to be portable and may be implemented on most microprocessor based hardware.

They Really Can Work For Us

In a sense no more than a "spin-off" from the team's programme of systems development work, the creation and demonstration of this text-handling system showed clearly that there is *much more to microcomputer systems within the personal computer field than games.* Here was proof indeed that personal computers are not mere playthings; *they have a serious and growing function within society.*

The hardware

In its standard form the Nascom 1 microcomputer board is equipped with 2K of RAM and sockets for 2K of 2708 EPROM. The first 1K of EPROM (addresses OOOOH to O3FFH) is equipped with an EPROM containing the operating system for the Nascom 1 known as NASBUG; the function of which is well known, for its set of standard I/O routines and its debugging facilities, by all Nascom 1 users. A socket is provided for the second 1K of EPROM (addresses 0400H to 07FFH). The screen memory occupies the area of RAM from 0800H to 08FFH. Nasbug uses the area of RAM from OCOOH and OC4FH for scratchpad and stack purposes which leaves the area of RAM between OC50H and OFFFH for use as user program, data and stack.

To make the most effective use of this available area of RAM the letter editing demonstration was prepared to run in EPROM. We prepared a stripped down version of Nasbug, leaving only those parts of the program required for initialisation, cassette I/O and keyboard management. We added to these a handler for the printer which used the PIO facility of the Nascom 1 board and the letter writer control program. This was programmed into a 2708 EPROM which was plugged into the socket usually occupied by the Nasbug EPROM. The second EPROM socket was occupied by an EPROM containing a set of routines (all of which are not used in this demonstration) which controlled the manipulation of text within the screen memory area using an algorithm devised at Dataskil. These routines are the kernel of the demonstration program and a whole generation of programs in the Dataskil microcomputer system software suite.

Dataskil is one of the first companies to become an authorised consultant under the British Government's £15 million MAPCON scheme, which encourages UK industry to apply microprocessor techniques to a wide range of products and production processes. Companies who seek professional advice under this Microprocessor Application Scheme, administered by the Department of Industry, become eligible for a refund of up to £2000 of the cost of approved consultancy work.

Conclusion

Although specialising in commercial business systems, and in INTEL and ZILOG hardware, Dataskil's software expertise is both broad and deep. A consultancy service is available to provide advice on the potential for microprocessor installations and processes within customer's own manufacturing units, and on the incorporation of microprocessors in their end-product design.

PCW *The author has been working with microprocessors for over five years and is currently a Principal Consultant at ICL Dataskil, Reading.* PCW.

PCW Book Review

Michael James

MICROPROCESSOR INTERFACING TECHNIQUES
(Second Edition)
A. Lesea, R. Zaks, 1978; 416 pages
(Sybex Inc., 5½" x 8½", £7.50)

A microprocessor system is analagous to an iceberg. The CPU chip represents the visible tip of the iceberg sparkling in the light of publicity, while the memory devices, control circuits, interface components and peripherals lie hidden beneath the water and are often neglected in microprocessor textbooks. *Microprocessor Interfacing Techniques* tries to redress the balance by dealing exclusively with the components required to turn a microprocessor into a system complete with input/output devices ranging from VDU's to floppy disks.

The book begins with the basics of microprocessor systems design: address decoding, bus drivers and receivers, serial and parallel interfaces using standard chips. Three fundamental techniques of controlling the flow of data between peripherals and microprocessor (polled, interrupt driven, and direct memory access) are described in Chapter 3. The largest section of the book, Chapter 4, discusses the interface requirements of a wide range of peripherals, including keyboards, cassette interfaces, VDU's and stepper motors. Some important peripherals are dealt with in detail. For example, an extensive section on the floppy disk subsystem covers the disk drive itself, the format of the recorded data, and the drive controller. Several circuit diagrams of commercial floppy disk controllers are given.

Chapter 5 is devoted to digital to analogue and analogue to digital conversion techniques, a subject which is very important if the microprocessor is to be used to control the real world. In Chapter 6 both serial and parallel bus standards are described, including the popular S100 bus used by many 8080/Z80 microprocessor systems. The most interesting part of this chapter is the description of the IEEE-488 bus used to connect systems together on a common data highway. Although the IEEE-488 bus is designed for micro-processor-controlled instrumentation, one home computer manufacturer has adopted this standard for use by their microprocessor system (PET), and Motorola have introduced an interface chip for the IEEE bus. In Chapter 7 the case study of a microprocessor based multiplexor ties together many of the ideas discussed in previous chapters. The book ends with a useful chapter on testing microprocessor systems, a topic often omitted from other textbooks. A wide variety of test procedures are described, including the use of logic analysers and signature analysers.

Microprocessor Interfacing Techniques is a mine of information covering a wide range of topics, and is very useful for the enthusiast wishing to have an overview of microprocessors. However, on the cover of this book it says, "This book will teach you how to inter-connect a complete system, and interface it to all the usual peripherals." This is not entirely true. The description of many of the interfaces is too sketchy to permit the reader to design his own interface without considerable effort. Where complete circuit diagrams are presented, they are often taken straight from the manufacturers' literature with little or no additional explanation. For example, the circuit of the Thomson-CSF CRT controller is provided without an adequate explanation of how it works, and a block diagram of the functions of the CRT controller chip is given without a definition of symbols used in the diagram. The book appears to be compiled from a large number of manufacturers' data sheets, which leads to an uneven treatment of the subject of interfacing techniques. The defence open to the authors is that they have tackled such a wide range of topics that had they allocated to each of them the amount of space it deserved, the book would have reached encyclopaedic proportions.

Alan Clements

PCW OPEN PAGE

The Amateur Computer Club View

Mike Lord

G–79 ; A PLEA FOR A SMALLER SLOWER, BUS

Most Computer Club meetings I go to these days seem to end up with a heated argument about the relative merits of different bus structures. You know the kind of thing; "S100 – a triumph of marketing over design, E–78 – it may work but it doesn't exist, NASBUS – what's that?" and so forth until closing time reminds us that the only bus that really matters is the last one, which we've just missed.

But at a recent gathering, as the discussion drifted towards the possibility of organising a bulk buying group for 64k RAM chips, I began to wonder whether the whole question of high speed busses a la S–100 isn't becoming a subject fit only for historians. At least as far as the amateur is concerned.

Because if one were to design the next generation of hobby computer now, using components which will be coming onto the market during the next six months, then one could get a very powerful machine onto a single printed circuit board. With a minimum of 16k RAM, 16k ROM holding a good BASIC interpreter plus an Editor/Assembler, semi-graphics VDU drive, ASCII keyboard and cassette interfaces, all controlled by one of the new CPU chips, it could sell for around £250, and still have room on the board for expansion to say 64k RAM, plus some other goodies such as a floppy disc controller and several IO ports. And such a system would surely be more than adequate for most amateur computing enthusiasts, any further expansion being in peripheral equipment such as a hard copy printer, floppy disc drives, and perhaps a PRESTEL interface.

And for those who complain that 64k is not very much RAM, well – given a year or so, then INMOS chips will surely allow us to put 256k bytes into the same space.

So, with up to 4 1M byte floppy disc drives daisy-chained to a single chip controller, and an on-board memory mapped VDU controller, we find that the traditional type of high speed CPU-memory bus has vanished.

But we will still have to have some way of connecting relatively low speed peripherals onto the computer card. It is this data connection problem that I believe to be worth examining.

For simplicity of the computer board itself and the interconnecting cables, some form of bus structure would seem to be better than having one cable running out to each peripheral. So the bus will have to carry addresses as well as data. Treating the data as 8 bit bytes seems reasonable, although whether these bytes would be transmitted in bit-serial or bit-parallel form is debatable. One such bus structure already exists, in the form of the IEEE488 (H–P) specification, and it is worth noting that the PET has an IEEE488 compatible port, but perhaps a simpler version would be more suitable.

Whatever the final structure, if international agreement could be reached on the definition of such a low speed bus for connecting future personal computers to their peripherals, then surely the manufacturers of peripherals and computers would benefit as well as their customers.

I would, therefore, be interested to hear from anyone who would like to help pursue these ideas, and, for want of a better inspiration, christen this new bus structure G (for Global) 79, in anticipation of an interesting year for the amateur computing fraternity.

PCW *Scoop! In the next issue,* James Cunningham, *Chairman of the Amateur Computer Club, gives concrete advice on setting up a local group.* PCW

RESEARCH REQUEST

I am an undergraduate student in my final year at the University of Wales Institute of Science and Technology, and I am currently researching a dissertation on microcomputers, and their applications.

However, my main problem in researching this subject is the lack of published information, in particular in the area of market research.

Any empirical evidence available, with respect to the above, or indeed any advice or information of relevance, would be greatly appreciated.

Graham J. Woolfman
29 Malefant Street, Cathays, Cardiff

MSc. COURSE IN MICROPROCESSORS

The Department of Electrical and Electronic Engineering, Aston University, announce the commencement of a 1 year MSc course in Microprocessors. The course will run annually from October, 1979. Applications will be welcomed from Honours degree holders in Computer Science or Electronic Engineering and from those practising engineers who can demonstrate a similar level of attainment. The course will cover both hardware and software design, with particular emphasis on the application of microprocessors in a real-time environment. Following six months of course work, a six month project will give the participants a firm grounding in the practice of their new skills. For full details write to Dr. R. G. Wilson, Department of Electrical and Electronic Engineering, 19 Coleshill Street, Birmingham B4 7PB.

NEW COMPUTER CLUB

The Oxford and District Amateur Computer Club has just been formed. If you're interested, contact S. C. Bird at 139 The Moors, Kidlington, Oxford OX5 2AF.

MERSEYSIDE MICROCOMPUTER GROUP

PCW has just received its first newsletter from the Group. A very active set of people – there is already talk of forming special interest sub groups. Man to contact is The Secretary, John Stout, Department of Architecture, Liverpool Polytechnic.

8-Bit Analogue Input for PET

D. P. Siddons *and* A. R. D. Rodrigues
University of London King's College

The circuit diagram shows a simple A to D convertor providing 8-bit conversion in under 1 mS. The convertor uses an inexpensive 8-bit D-A convertor I.C. which has an integral counter. This counter is incremented by the clock comprising IC 4a . . . The output of this convertor is compared with the signal to be converted, and the clock stopped when the D-A output exceeds the signal. The contents of the counter then represent the input voltage. A differential amplifier is used to provide balanced inputs and again, allowing a wide range of input sensitivities to be achieved. The convertor plugs straight into PET's user port and handshake takes place via CA1 and CB2. To manage the handshake a machine code routine and loading program is given in lines 1000 →.

The remainder of the program turns PET into a 'chart recorder' display, making about 10 samples per second. The speed is BASIC limited, and more extensive machine code would speed things up considerably.

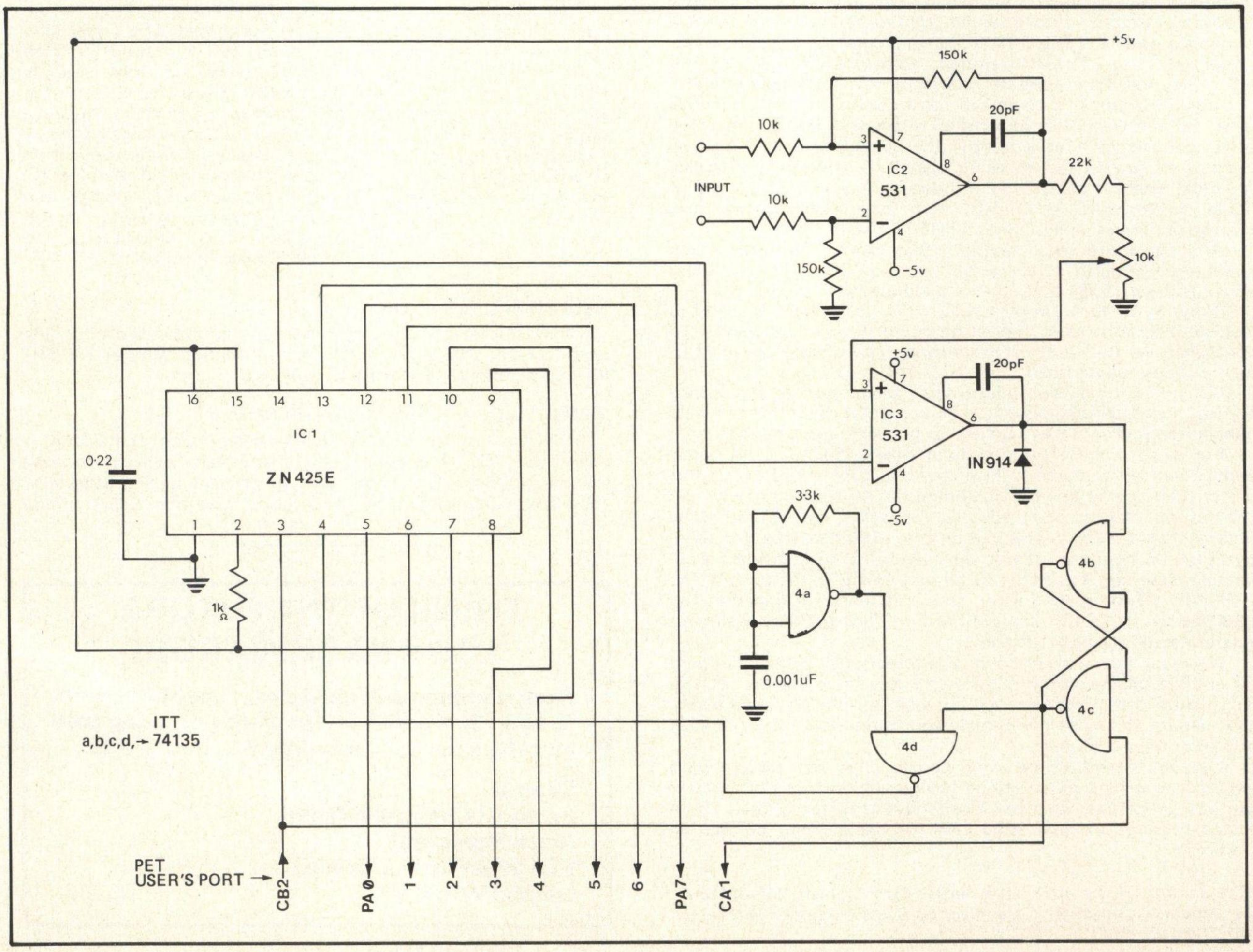
```
BEGIN   PCR  =   76,232
        PFR  =   77,232
        LDA imm.    192
        STA       PCR          :  CB2 Low
        LDA imm.    224
        STA       PCR          :  CB2 High

LOOP    LDA abs   PFR
        AND imm.      2        :  CAI Negative edge ?
        BEQ          -7        :  if not, LOOP
```

Fig. 2. MC program for handshake (lines 1020 & 1030 of Fig. 3).

```
10    Z   =   59456        :  POKE Z + 12,224
20    POKE Z +  11, 0  :  POKE Z + 3, 0
30    SYS (826)  :  A  =  PEEK (Z + 1)
40    PRINT A  :  PRINTTAB (A*40/255); "*"  : GOTO 30
1000  RESTORE : FOR I = 1 TO 18 : READA : POKE 825
      + I, A  :  NEXT
1020  DATA 169, 192, 141, 76, 232, 169, 224
1030  DATA 141, 76, 232, 173, 77, 232, 41, 2, 240, 249, 96
```

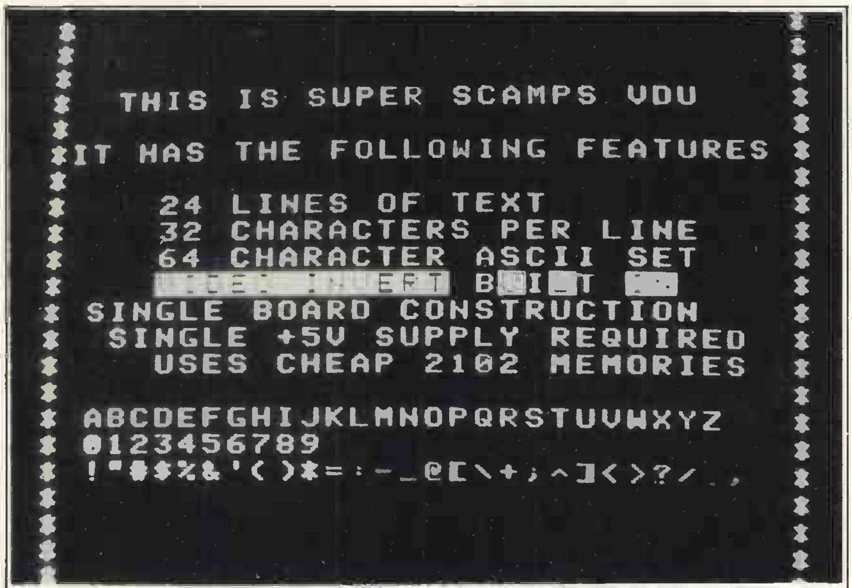
Fig. 3. 'Chart recorder' program: first type GOTO 1000 (Return); then RUN (return).



Super Scamp's V.D.U.

A Simple Micro Output Device

W.G. Marshall



A Typical Display

Introduction

The visual display unit (VDU) is an alternative to the teletype for obtaining an intelligible output from the home computer system. Unlike commercial items that can cost hundreds of pounds, this can be made at home with very little effort and cost. If you own an old television set which is seldom used then you are already half way there, as the circuit to be described plugs into an unmodified set via a simple UHF modulator. The design has been kept as simple as possible, with all cursor functions left to program control.

Memory Mapped RAM

The characters (or rather their ASCII codes), to be displayed are stored in random-access memory operated in the memory-mapped mode. This means that the computer accesses the VDU memory as if it were just another 1k block of main store. In fact if display is not required, the former may be used for program or data storage.

The Display Features

The picture produced is of the following form:–

32 characters per line
24 lines of characters
64 types of character (i.e. Full upper case ASCII set.)
White-on-black (normal) or black-on-white (inverted) available on any of the displayed characters.

Circuit Principle

The average black and white TV set produces a picture by varying the intensity of a scanning beam of electrons, which moves across a phosphor coating on the back of the screen. The phosphor at the tip of the beam glows with a brightness proportional to the latter's intensity. The tip or 'spot' starts in the top left hand corner of the screen, moves across to the right corner whereupon it flys back quickly to the beginning of the next line, fractionally down on the first.

The beam spot flies back to the top left hand corner when it has scanned all the way down to the bottom right.

In order to forestall abuse from the TV experts I will say here that this VDU design does not use interlacing as the picture definition is quite good enough without it. As a result black lines alternate with modulated picture lines. We are only interested in two levels of brightness, black and white, so digital techniques are ideally suited to this task.

Character Format

Each character is built up as a series of black and white dots in an 8 x 10 matrix. (fig. 1), ignoring the interlaced black lines. The character size is actually 5 x 7 dots, the rest providing the border separating the figure from its neighbours. The border is black for normal video and white for inverted.

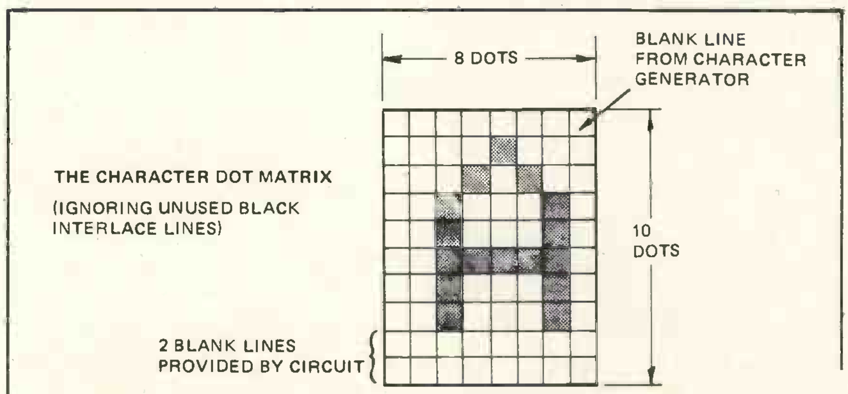
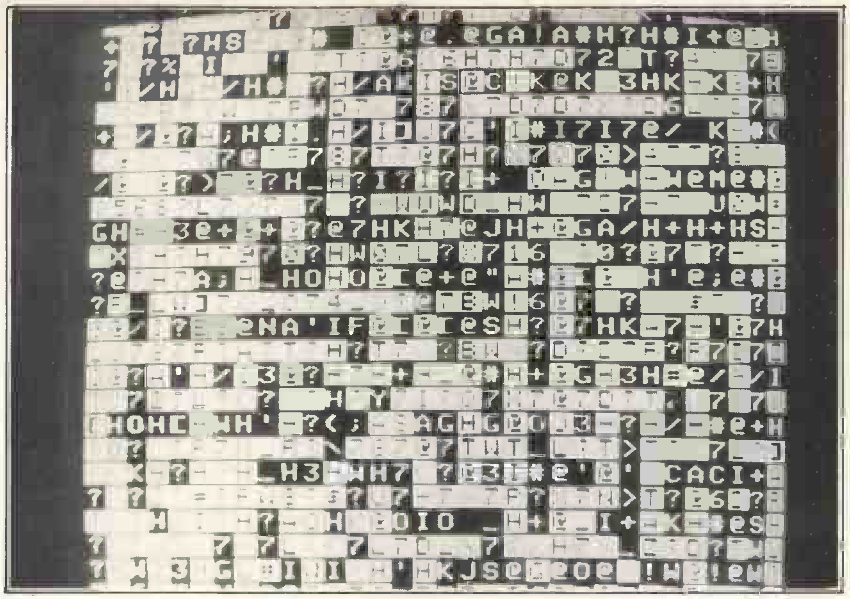


Fig. 1



Random Display when Power first applied

At this point note that X14 only produces a character 5 dots wide by 8 dots deep. Hence two scan lines must be blanked between each row, as the row counter X4 counts up to 10 rows before returning to zero. This is easily done by inhibiting the load latch during these two picture lines. X11a and b perform this function and also inhibit the latch during the horizontal blanking interval. Horizontal blanking is thus inserted here and not at the video mixer as is usual. The top line of all characters provided by X14 is blank anyway, which gives us a total of 3 blank lines (not including the black unused interlace lines) between each character row. Horizontal character separation is 3 dots, as three of the 74165 parallel inputs are tied low.

The TV Signal

The signal we need to drive the TV set via the UHF modulator is called the composite video. It is called composite because it is a mixture of three other signals: picture video, synchronising pulses and blanking pulses. The 'sync' pulses are those which trigger the spot fly-back and occur at the end of every line and 'field' scan. The blanking pulses are there to black out or 'blank' the screen while the spots fly back. But how does the TV tell the difference between the random video producing the picture, and the sync signals? The answer lies with the voltage levels of the composite video. (fig. 2) It can be seen that to produce a white spot on the screen at a particular point requires the signal level to be about +1v. For black and blanking, then, the signal is at 0v, and the sync pulses go down to about 1v below the black 'pedestal' level. The diagram (fig. 2) does not show the vertical sync pulse, but this merely amounts to a longer horizontal sync pulse occurring once in every 320 of the latter. This is taking a slight liberty, as the true figure should be once in every 312, but the logic is simpler and TV set tolerance can usually allow this discrepancy. I have also ignored equalisation pulses, etc. as unnecessary in this application.

Circuit Description — Display Section

The display section (fig. 3) consists mainly of a master oscillator or 'dot clock' running at 5 MHz. It drives a counter series or frequency divider chain. One period of the dot clock, 200 ns, is the width of a dot on the screen. Output c of the first counter X2, yields a frequency of 5 x 10^6 divided by 8. This signal clocks the load latch and so a load pulse is generated and sent to the video shift register after every 8 dot periods. That is, after every character on the scan line. Back to the divider chain, and division by a further factor of 40 yields $\overline{\text{HBLK}}$. This signal waveform has a period of 64 μs — the width of a TV line scan and is at a logic '0' for 52 μs of that time — about the visible picture width. Invert this and we have our horizontal blanking signal. Further division by 320 yields VBLK, nearly a 50 Hz square wave, suitable for generating our vertical sync pulses. Selected outputs from this divider chain, MA0 — MA9, go over to the address multiplexer board to provide addresses for the RAMs which hold the codes for the desired screen characters.

The character generator, X14, has two sets of inputs. One set selects the character to be displayed at a particular point, and the other tells the device which line of that character is currently being scanned. X14 then puts out a pattern on its outputs appropriate to that character line. These outputs are clocked by the previously mentioned load signal into the video shift register X13. The dot clock then shifts them out serially to the video mixer. Data bit D6 if set (= 1), will cause this serial stream to be inverted by X7a to yield a black on white character if desired.

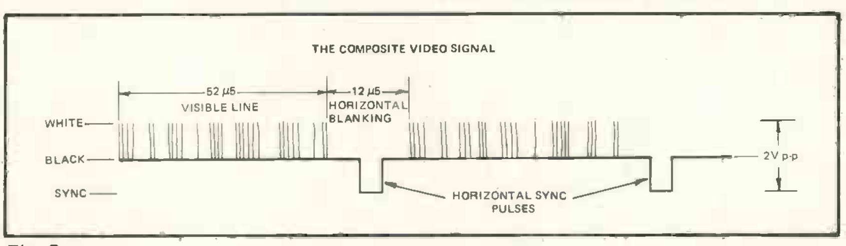


Fig. 2.

Sync Signals

The horizontal and vertical (field) sync signals are generated by monostables X6 and X12 respectively. At the end of a line scan $\overline{\text{HBLK}}$ goes high to begin line blanking. This rising edge is delayed by X6a before triggering X6b. The output of the latter goes high for about 5 μs and is inverted by X1d to produce the horizontal sync pulse. VBLK generates the vertical sync pulse in a similar way, except that the pulse is nearly 1 ms long.

Computer Access — The Address Multiplexer Section

The VDU divider chain produces the memory address lines MA0—9, and these are connected to the 2102 address inputs via multiplexers X17, X18 and X19. (fig. 4) A0 — A9 are the computer system address lines, SD0 — SD7 connect to the computer bi-directional tri-state data bus, and $\overline{\text{VSEL}}$ when low allows the computer to access the RAMs. Lines ID0 — ID7 go to the data inputs of the 2102s and may be connected to SD0 — SD7 if the clear screen facility is not required *(see* Options). The read/write strobe R/W strobes data into the RAMS off ID0 — 7, when low. Data from the 2102s is placed on the computer bus while the tri-state enable signal, $\overline{\text{TSE}}$, is low. When $\overline{\text{VSEL}}$ is high, the divider chain addresses the RAMs, when low the computer does. This means that wrong data is fed to the character generator while the computer has control, resulting in white 'flecks' all over the display. These white flecks are blanked (i.e. turned black) by monostable X29 which is triggered by $\overline{\text{VSEL}}$ going low. The $\overline{\text{Q}}$ output thus blanks the video for a period set by C3 and VR1, the latter being adjusted to suit the instruction time of the computer.

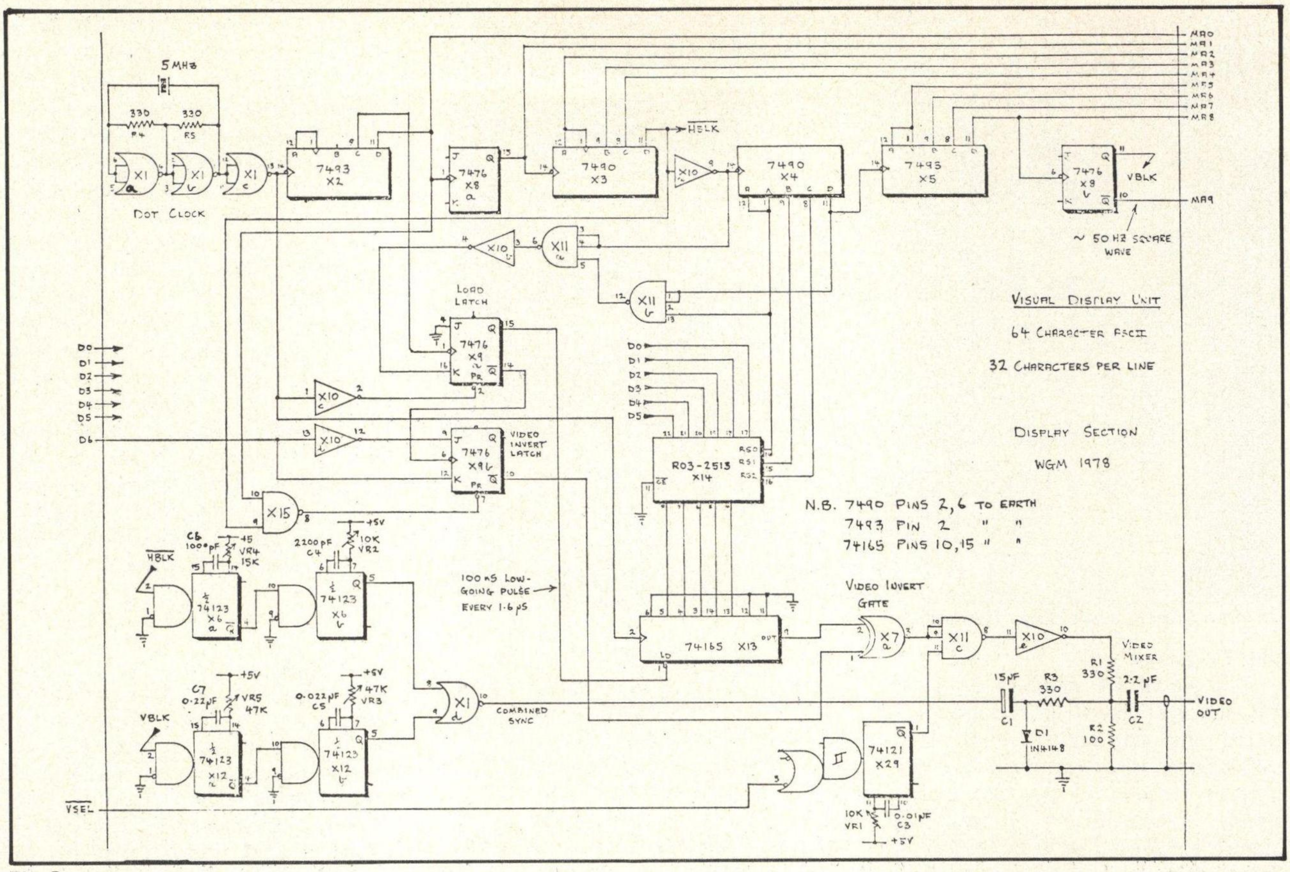


Fig. 3

Construction

Component layout is not critical and the prototype was constructed on a piece of tracked veroboard 4½" x 6½". Lay the I.C.s out on the plain side of the board at right-angles to the track direction, with five holes between adjacent ones. Cut the tracks between the pins of each I.C. and also halfway between adjacent I.C.s. Each leg should now have a copper 'pad' with holes for two wires. I have found it better to cut all the tracks in this way, and to wire the whole circuit with fine stranded wire, than to use the tracking for carrying signals. (fig. 5). Use heavy gauge wire for the power rails, and decouple the latter every fourth I.C. with 0.1 µF disc ceramic capacitor. Unless you are very shaky with your soldering, don't bother with sockets for the TTL I.C.s – save them for the 2513 and 2102s.

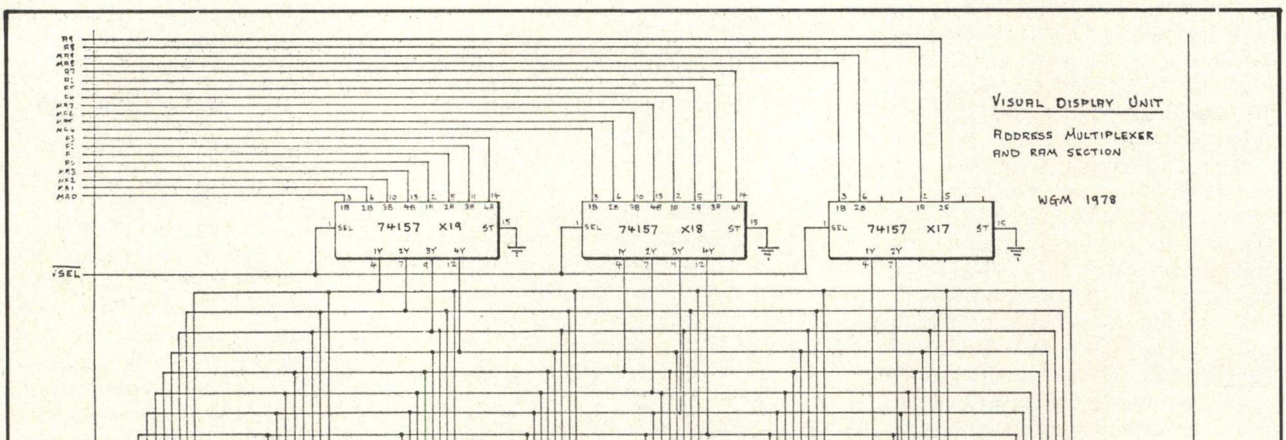


Fig. 4

Setting Up & TV connection

The video output may be connected to one of the available UHF modulators which then plugs into the aerial socket of the TV set. If you have one of the small video monitors appearing on the second-hand market, then the circuit will drive it directly without the need for a modulator. Setting up consists of adjusting the various potentiometers until a stable picture is obtained. The TV set horizontal and vertical hold controls may also be adjusted.

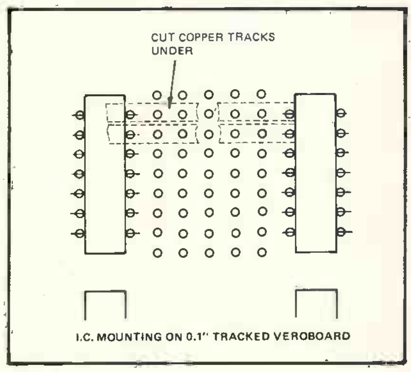


Fig. 5

Options

Two 'add-on' features have so far been devised and built for this VDU. These are a clear screen button and a repeat button. A further article will give the details, together with suggestions for interfacing and software routines for a SC/MP system. The repeat facility is really attached to the keyboard interface and the latter's connection to a KB6 will be discussed as well.

Sources of Information

1. Jean-Daniel Nicoud, Alphanumeric TV Display Interface, MICROPROCESSORS, vol. 1 no. 4 April 1977.
2. Modify your TV for Video Input, ELECTRONICS TOMORROW – an ETI special issue. (To enable the composite video to be injected directly into the set without a modulator.)
3. Don Lancaster, TV TYPEWRITER COOKBOOK, (A mine of useful information on VDUs, keyboards and their interfaces.)

Components . . . .

Integrated circuits

X1	7402
X2, X5	7493
X3, X4	7490
X6, X12	74123
X7	7486
X8, X9	7476
X20, X21	74125
X29	74121
X10	7404
*X11	7410
X13	74165
X14	R03–2513
X15	7400
X17, X18, X19	74157
X22-X28, X31*	2102

Resistors

R1, R3, R4, R5 330 Ω R2 100 Ω
CR1, VR2 10k Ω VR3, VR5 47k Ω
VR4 15k Ω

Capacitors

C1	15µF	C2	2.2µF	C3	0.01µF
C4	2200pF	C5	0.022µF	C6	1000pF
C7	0.22µF				

plus 6 x 0.1µF disc ceramic capacitors distributed on power rails. (see text).

Misc.

D1 1N4148 silicon diode 1 x 5MHz crystal
5v @ 1A dc power supply

*Note: X11 may be a 7400 and the spare gate used to replace X15. Three input gates were used on the prototype as new facilities may be added later. RAM X31 may be omitted if the VDU memory is never used as main store.

IF YOU CAN'T BEAT THEM :-

VECTOR GRAPHIC INC

VECTOR V18A slot Motherchassis accepts the wide range of Vector S100 cards and makes an ideal base to build a microcomputer system. Computing power is available to perform a wide range of tasks from industrial control to small business.
£350

PR2 12K PROM/RAM card holds a comprehensive monitor program for system testing and configuration. Normal operation is in conjunction with a serial terminal via I/O card.
£160

FLASHWRITER, memory mapped VDU with graphics, allows a system to be built without a terminal or I/O card. Specify version EV of monitor program.
£150

I/O, Switchable 110 to 9600 baud serial interface plus two 8-bit parallel I/O ports.
£125

High resolution graphics interface bit-maps 8K of RAM to 256 x 256 points, or 128 x 128 with 16 level grey-scale.
£150

Z80 Processor card	£140	8080 Processor card	£120
8K Static RAM 4MHZ	£140	16K Static RAM 4MHZ	£300
Analogue Interface	£70	Precision analogue interface	£250
Rackmount 18 slot motherboard	£150	Rackmount power supply	£90

8K Computer System £895

MICROPOLIS

Micropolis disk drives employ higher standards of engineering to pack either 143K or 315K bytes per diskette, formatted. Supplied complete with controller card, cables, manual and software they plug directly into the S100 bus; 8080 or Z80.

Extended disc BASIC, mnemonic editor and assembler are provided, to run under the powerful MDOS operating system.

Add-on units are supplied to extend the system to four drives and one drive per system may be powered from the S100 bus.

143K System S100 powered	£439	143K Add-on S100 powered	£279
143K System Mains powered	£499	143K Add-on Mains powered	£339
315K System S100 powered	£649	315K Add-on S100 powered	£349
315K System Mains powered	£699	315K Add-on Mains powered	£399
Twin drive System 630K	£1159	Twin drive Add-on 630K	£859
S100 bus regulator	£14	Diskettes per five	£24

Unmounted drives available from £225

JOIN THEM !

VECTOR MZ

£2300

Combining the best features of the VECTOR GRAPHIC computer and twin MICROPOLIS 315K byte drives. The Vector MZ produces, in one package a powerhouse of Microcomputer ability.

The VECTOR PROM monitor bootstraps directly to either MDOS, for housekeeping and Assembly language operation, or to BASIC to run high-level user programs. Provision is made to immediately attach a printer, for example one of the extensive range from Centronics sold by Sintrom, enabling use of the powerful printer-related features in the MICROPOLIS Software.

Applications Software for the VECTOR MZ now in preperation will perform a wide variety of business functions; stock control, invoicing, ledger and mailing lists. Further applications exist as a microcomputer development system, and low cost replacement for minicomputer control and instrumentation.

VECTOR MZ configuration includes:

Twin disk 630K minifloppy.
Full Micropolis disk software.
Z80 4MHZ 32K processor.
1 Serial port, 2 Parallel ports.
12K PROM RAM card with extended monitor.

And will support:

Flashwriter, Graphics interface, Analogue interface
Additional RAM, additional minifloppy drives.

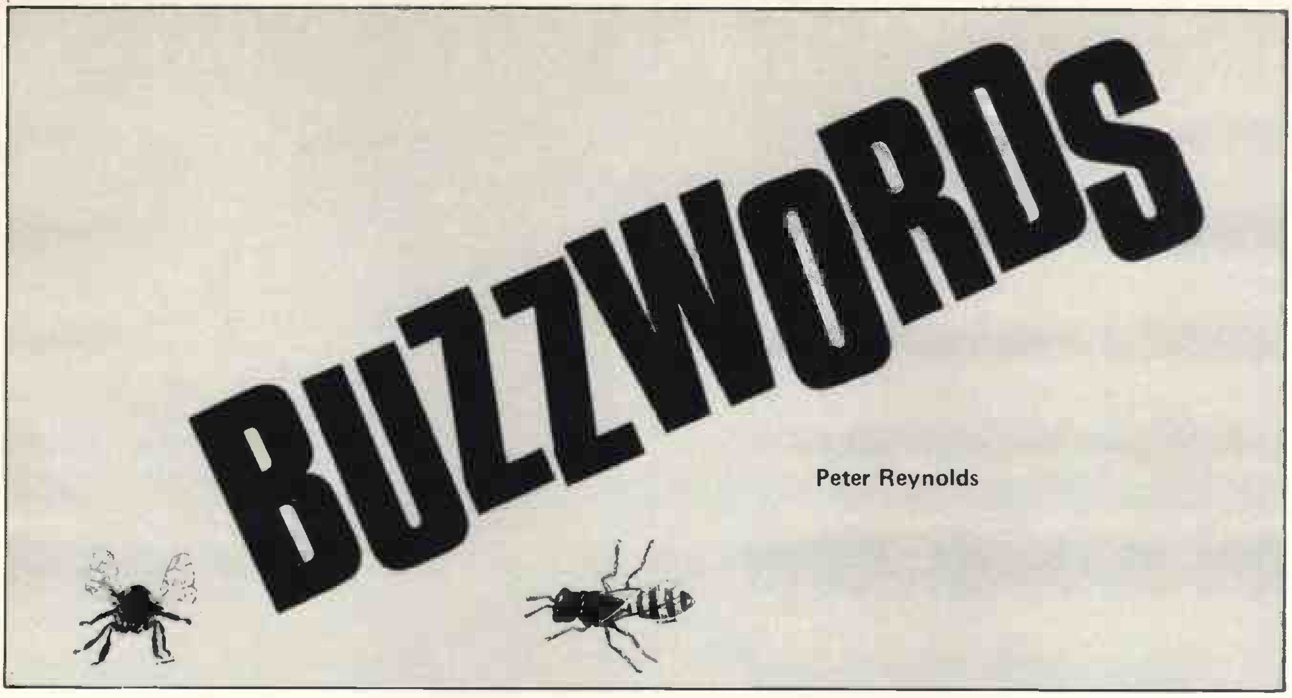
Centronics Microprinter	£398
Centronics 779	£780
ADM3A VDU	£620

SINTROM GROUP

PRICES EXCLUDE VAT
ALL EQUIPMENT FULLY ASSEMBLED AND TESTED
OEM AND DEALER ENQUIRIES WELCOME

DEMONSTRATIONS IN OUR SHOWROOM

Sintrom Microshop
14, Arkwright Road,
Reading, Berks. RG2 OLS.
Tel: Reading (0734) 84322
TELEX: 847395
CABLES: SINTROM READING



BUZZWORDS

Peter Reynolds

D/A Digital to Analogue.

DAC. Digital to Analogue Converter.

Daisy Wheel — a set of printer characters arranged in the form of a serrated disc of plastic, resembling a flower. Rotation of the wheel brings the required character in front of an electromagnetic hammer which hits the type against a ribbon to print the character. Daisy wheels can be exchanged readily, to print a different typeface or size: they give good quality printing.

Dartmouth (Basic). Dartmouth College (USA) gave its name to the official version of the BASIC programming language. Most BASIC interpreters cover the Dartmouth standard but computer manufacturers tend to add a few extensions and improvements of their own devising.

Damping. Reducing the amplitude of electrical oscillation or mechanical vibration.

Data. Elements of information. In a digital computer every collection of two or more *bits* comprises data. But the term is often used in a special sense to signify facts and figures which are to be operated upon by the computer's stored program. In this sense "data" refers to input, "information" to output. "Data" is synonymous with *facts as opposed to instructions;* these facts are generally expressed in terms of numbers, letters and symbols.

Data Bank. A relatively large store of data in a form directly accessible for electronic data processing. A data bank typically holds all the information necessary for a user's system. The data is normally held in backing storage, such as discs, with relatively fast random access and transfer rates.

Data Base. Data for input to an E.D.P. system with particular reference to its structure or organisation. Integrated processing systems require more complex and more complete data bases than more simple applications do. For instance, stock control could require identification of supplier and re-order level for each item. Sales invoicing might also need catalogue identification and unit value. An integrated system would require all these factors together with any further data relevant to other aspects of the system.

Data Bus. The *bus* or *wiring* that carries data between different parts of a computer system.

Data Capture. The collection of computer input data by some automatic means. For example, a computer used to control a network of traffic signals would have direct connection to various sensors detecting the number and speed of vehicles passing various points. This data would not only control the signals but be retained for subsequent traffic analysis as required. Comparable arrangements are commonly met where a computer is used to control some factory process, and devices such as cassette recorders are sometimes linked to accounting machines and cash registers; but it is growing more popular to link the cash registers etc., directly to a computer so that they become specialist terminals rather than mere data capture devices.

Data Density. The volume of data which can be accommodated in a given area of storage, for exmple, 1" run of magnetic tape. Synonymous with *packing density.*

Data Logger. A device which captures information as it arises and records it — by pen on moving paper (as in a barograph), on magnetic tape or otherwise.

Data Preparation. The preparation of incoming information (for example, about events such as sales, production, hours worked) so that it can conveniently be fed as data into a computer system. This is commonly achieved by punched cards and *this preparation* distinguishes batch working from real-time working.

Data Processing. The arrangement of facts into a form suitable for automatic manipulation (sorting, calculating, recording, etc.) and the performance of these and related operations on the prepared data.

Data Processing Manager. Commonly the senior computer specialist in a commercial E.D.P.— using concern. He usually reports to an established member of top management (typically, commercial director or chief accountant) and controls all data processing staff. D.P.M. is the favourite of several rival titles for this responsibility, one alternative being *Manager, Information Services,* while Computer Manager may be more limited in scope, when there are also other methods of processing or providing information.

Data Terminal. A device which can send or receive data along a link, such as a telephone wire. For instance, a teletypewriter or a visual display unit.

Data Transmission. Sending information between an input/output device and a remote central processor. Data transmission can be achieved over distances that can be spanned by radio and telegraphy. It occurs, for instance, between spacecraft and their ground-based control stations, and in commerce between a central processor at head office and terminal devices at branches.

Data Validity Check. An automatic check within a computer to see that input data conforms to certain general *parameters* — for example, that a name must be expressed in alphabetical characters and not in numerals, the day of the month should not be a figure greater than 31, the monthly pay an individual should not exceed £999, etc. When data fails to pass a validity check, processing may be halted and an indication given on the operator's *console;* alternatively, the work may be allowed to proceed and the rejected item may be thrown up for separate consideration at the end of the run.

Datel. The Post Office system for data communication via the telephone network, including high speed or wide band channels. Two-wire speech circuits will currently work at up to 300 bauds while four-wire connections typically operate at up to 9600 bauds.

DC. Direct Current.

Debug. To trace and correct errors in hardware or software.

De-bugging Routine. Software to help locate the causes of hardware or program errors. For example a diagnostic routine may test all the memory locations, to show if any is faulty. Similarly trace instructions may be used to display the value of variables at intermediate stages in the operation of a program, thus helping to find the precise point at which some unintended effect occurs.

Decade. A sequential group of ten. A decade switch will have ten positions, each corresponding to a value in the range 0-9, or possibly 1-10.

Decay Time. The time in which a voltage or a magnetic field decreases to one-tenth of its original value.

DEC. A manufacturer of computers and peripheral devices.

Decimal. The familiar numbering system in the scale of 10, using the ten digits 0 to 9 to express any required value. Computers often employ *hexadecimal, octal* and *binary* numbering.

Decision Table. A system for compact expression of the variety of actions required to follow particular combinations of circumstances. See Fig. 1

Deck (1) Tape — A device for reading, writing and transporting magnetic tape, including tape in cassettes. A computer cassette deck may be identical with a low-priced audio deck or it may be engineered for specially rapid, frequent and precise starting and stopping.

Deck (2) Punched Card — A pack or set of cards relating to a particular file or program.

Decode. To reverse a previous process of encoding; to change data from one form of notation to another.

Decoder. A device that *decodes*, generally translating from binary or machine language to the *alphanumeric* and *decimal* form which can more readily be understood by machine users. A decoder can also take the form of a matrix or network of devices arranged to select one or more specific output channels in response to a particular combination of input signals.

Decollator. A machine which takes multi-part continuous stationery, such as emerges from a fast-line printer, separates the various sheets and at the same time removes any interleaved carbon. An attachment or separate *edge-cutter* may also be required to trim off one or both margins, carrying not only sprocket-holes but the edge where one-time carbon paper is gummed or crimped to the sheet below. A decollator should be distinguished from a *burster*, which separates the end of one form from the beginning of the next.

Decrement. *n.* Amount by which a quantity is reduced; opposite of "increment". *vb.* To make a reduction, e.g. in a counter each time a program goes through a loop.

Degradation. Lowering of quality as the result of processing. Rounding or truncating for example, especially if repeated in several successive stages, will degrade the accuracy of the final solution. See also *graceful degradation*.

Delay. The time by which a signal is retarded. This may be done deliberately and precisely, as an alternative to putting the signal into memory.

Delete. A character or series of characters in a computer system may be erased, via the input keyboard, using the appropriate ASCII character.

A complete line may be deleted with another ASCII character (typically generated by pressing Control key and X simultaneously on the keyboard.)

Delimit. To mark the point of change from one group of characters to the next a *delimiter* character (reserved for that purpose) may be used instead of the space (which computers often ignore). The comma and oblique stroke are often chosen as delimiters, so the sequence of numbers 9 19 123 might be entered as 9,19,123 or 9/19/123.

Demand Processing. Computing which begins immediately when data is entered, without entering a job queue.

Demodulator. Device for separating an informative signal from the carrier on which it has been imposed for convenience of transmission. Radio and television signals are generally added to (that is, used to modulate) carrier waves at the transmitter, and the process is reversed in the domestic receiver. A similar process is used when data is transmitted over G.P.O. telegraph and telephone lines.

Destructive Read-Out. A type of computer memory wherein information is automatically erased when it is read for transmission to some other part of the computer configuration. This contrasts with *non-destructive read-out* where, as with a gramophone record, the message remains virtually unchanged by successive readings or playings. Computers with destructive read-out systems are normally equipped with hardware arrangements to reinstate the information taken from memory unless this is to be deliberately left blank, or overwritten.

Diagnostic Routine. Using a standard procedure to trace a malfunction in a computer, especially a hardware fault.

Dichotomising Search. The binary system of searching data, in which the area being looked at is successively halved and quartered and divided into eighths. See *binary search*.

Dictionary. A look-up or translation table associated with a program which specifies with the precision required by the processor the details implied by brief data-names.

Digit. A character used to represent one of the integers smaller than the radix or base of notation; for example, in decimal numbering (radix 10) a digit is any one of the characters 0 to 9, and in binary systems (radix 2) it is either 0 or 1.

Digital. Pertaining to numerical digits and, therefore, capable of working in discrete numbers, as opposed to *Analog*.

Digital/Analogue Converter. A device whose input is digital and output analogue, e.g. a continuously variable voltage which might control the speed of a motor.

DIN Plug. A type of low-current electrical connector, typically with five or six pins, popular for audio equipment and also met in some personal computers.

Diode. An electronic semi-conductor which allows current to flow in one direction but not in the other.

Diode Transistor Logic (DTL). An early form of computer circuitry whose elements included several diodes associated with one transistor. Superseded by *Transistor Transistor Logic*.

DIP. Dual In-line Package. A standard package or physical presentation of an integrated circuit, connections are made via two rows of pins with typically eight or twelve pins in each row, spaced at .1 inch intervals.

Direct Access (Store). Data or instruction storage in a computer which can be read or overwritten with minimum delay (in a matter of micro-seconds). In current practice direct access storage generally takes the form of CMOS RAM; this contrasts with backing storage, which may comprise magnetic tapes, etc., and will be very much slower of access. Synonymous with *immediate access* and *random access*.

Direct Address. An address which can be identified without reference to any form of *index* or *modifier register*. An *absolute* address (for example, location 8040 of working store) is a form of direct address but a direct address need not be written in machine language and could, therefore, take up a different position in working store each time a program is fed into a multi-program computer. Synonymous with *first level* address.

Direct Entry. Descriptive of data which goes directly to a computer (where its general validity is probably checked at the time of entry) without going through the off-line processes of data preparation which typically involve the punching, verifying and controlling the data by batches prior to processing proper.

Directory. Same as *dictionary*.

Disc. A form of data storage, similar in function to magnetic recording tape but in the form of one or more discs which are kept continuously revolving, so that random access to any part of the record can be achieved readily by a form of moving pick-up holding the read/write *head*. See also *floppy disc*.

Disc Pack (various proprietary spellings). A set of perhaps half a dozen rigid magnetic discs which are held together and spaced by a central vertical column so that they can operate in synchronisation on a single turntable device.

Display. A computer device for transitory communication with the operator. The most common form is the television-like *video monitor* but it may consist of a row of say 32 alpha-numeric characters similar to those used in a digital watch or even a group of simple lights representing the bit pattern in an *accumulator*.

DMA. Direct Memory Addressing.

DMM. Digital Multi-Meter.

Documentation. Preparing adequate notes to describe the reasons for each step in a computer program. This is very important, as it frequently happens that programs are amended after they have been prepared and not all the consequential adjustments are followed through. If the original programmer cannot be consulted or if he has forgotten what he did, and why, it can be very time-consuming to discover the purpose of a particular step in a program, and it could equally be time-consuming or hazardous to amend it or to delete a *loop*. On the other hand, programs left unamended may become unwieldy and inefficient.

Document Reader. A peripheral device capable of taking directly into a computer information recorded on sheets of paper. The reading process is typically optical and the data read may be in the form of marks or printed characters or even hand-written numerals.

Down-Time. The period when a device is unable to operate.

DOS. Disc Operating System.

DPDT. Double Pole Double Throw — descriptive of a mechanical electrical two-way switch controlling two circuits.

Conditions	Rule						
	1	2	3	4	5	6	7
Credit Rating..	Best	Best	Best	Best	Good	Good	Poor
Payments in past year ..	⩾£100	<£100	⩾£100	<£100	—	—	—
Balance owing ..	<£200	<£200	⩾£200	⩾£200	<£100	⩾£100	—
Actions							
Allow credit ..	<£1000	<£300	<£500	<£100	<£100	< £50	No
Print list for scrutiny ..	—	—	Yes	Yes	—	Yes	Yes

The symbol < signifies is *less than;* and ⩾ means *is equal to or greater than.*

Fig. 1. *This decision table illustrates a credit control system which fits every application into one (and only one) of seven categories and appoints specific actions for each case.*

DPST. Double Pole Single Throw – as DPDT (above) but one-way (or on/off) only.

D.P.M. Data Processing Manager.

Drain. The element, e.g. in a *field-effect transistor,* to which current runs from a *source.*

DRAM. – Dynamic Random Access Memory.

Drive. Mechanism for moving magnetic tape or disc.

Drop-in. The accidental addition of one bit to a data group.

Drop-Out. The accidental omission of one bit from a group of data in process in a computer or its peripheral devices. Drop-out, like *drop-in,* is usually detected by *parity check.*

Dry Joint. A soldered connection which, while appearing satisfactory, does not always make a good, low-resistance contact with one of the conductors thus joined.

DTL. Diode-Transistor Logic.

Dual-In-Line. A standard arrangement and spacing of connector pins in two rows, as currently found with the majority of integrated circuit *chips.* An alternative is *quad-in-line.*

Dummy. Computer instruction or data superficially similar to the real thing but not capable of being operated upon. Dummy quantities, and more particularly dummy instructions, are sometimes incorporated in programs to make provision for some later development of the program or to fill out to standard size a block of instructions.

Dump. To copy the contents of memory on to some other storage medium or to list it for examination.

Duodecimal. A system of numbering based on 12. The British system of twelve pennies to the shilling was half way to the duodecimal; but a real duodecimal system would require two extra digits in addition to the familiar digits 0 to 9.

Duplex. A mode of communication between computer and terminal whereby each can send and receive simultaneously. A terminal operating in duplex mode will not display what the operator keys in unless it is echoed back to the terminal by the computer. This provides evidence that the data has reached the computer correctly but an inexperienced operator may be confused and slowed down if there is noticeable delay before his key-strokes appear on the display or printer.

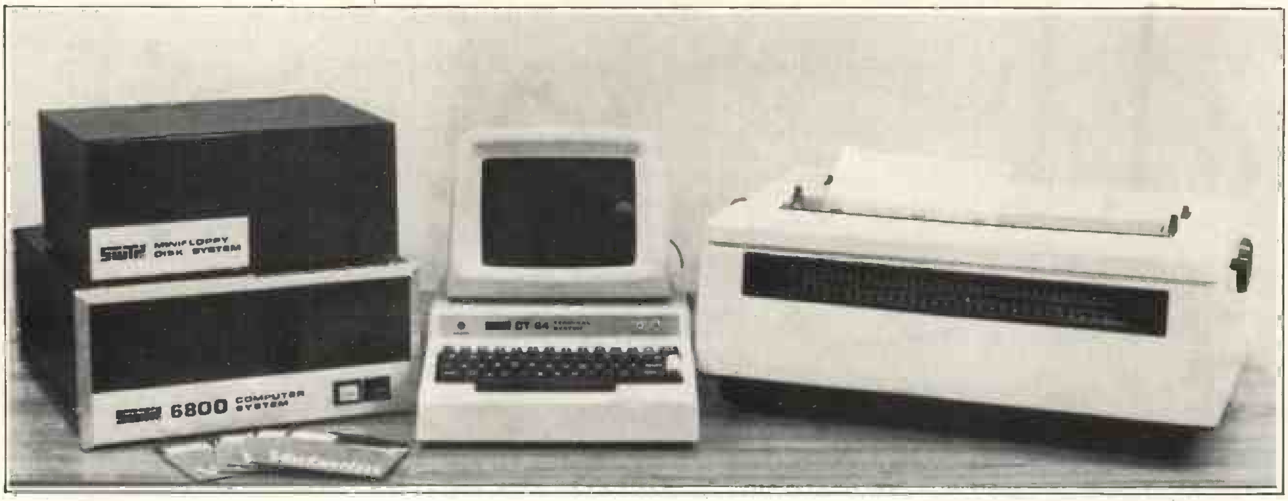
Dynamic Addressing. A system of varying the address referred to by a program for locating the data under process. This may be used to permit a sequence of program instructions to operate repetitively on changing data.

Dynamic Dump. A dump of data from one area of storage, performed periodically under program instruction. For example, information stored on magnetic disc may be dumped or reproduced on magnetic tape at the end of every operating shift, or the contents of part of a store may be dumped by printing out a hard copy. This may be part of a security system to allow valuable data files to be restored if the original files are accidently lost in anyway.

Dynamic Memory. A form of memory in which the bit pattern tends to decay but this is prevented by repeated refresh signals.

Dynamic RAM – see *dynamic memory.*

Dynamic Stop. A *loop stop* consisting of a single *jump* instruction. This leaves the computer ready to resume working immediately, possibly after the operator has touched a switch or a peripheral device sends an awaited signal.



He had never seen my computer and he was obviously impressed by the pile of perfectly typed overdue account letters it had just produced.

"How can you possibly afford a computer system in such a small company?" he asked, in that direct way suppliers have when they think that you may be overspending.

I had been anticipating the question. I had seen him glancing enviously at the Cash Flow Forecast, Sales analysis Report, and Back Order Schedule I had been referring to since he arrived.

He had realised that this was the first time ever that I had been able to put my finger on the facts which I need to schedule my next three months deliveries from him.

"I'll buy you lunch if you can get within £1,000 of the cost of the system," I said, generously, because it was his turn to pay today. *"And I'll tell you as much about it as you want to know."* I added.

"Well I can see it does the job one of those word processing machines does, and it's doing most of your accounting — but what does it actually consist of?"

Here was my chance to impress him with my very limited knowledge of the equipment itself.

"Well — here's the visual display terminal with the keyboard. As you can see it has upper and lower case characters and you use the keyboard like a typewriter. This box here is the computer itself which has 40K of RAM," I said quickly because that's all I know about it and I was hoping to avoid his next question. However, he butted in —

"What does that mean?"
"Er, well it's the amount of memory it's got."
"It couldn't be much in a box that size," he said.

"Well all I know is that it certainly seems to be enough to cope with any of the programs I use," I said defensively, *"and besides these disk drives hold over half a million characters of information which the computer can read whenever it needs them."*

"What's that in terms of names and addresses for instance?" he asked.

"Assuming 150 characters for each one it's about £3,800. And this is the printer which gives a typewriter quality letter or report."

"What else can it be used for?" he asked.
"Well this system is the top end of the range," I said proudly, *"but other cheaper models are used for everything from process control to medical interviewing, from playing games to student instruction, and from statistical analysis to travel booking."*

"You'll be telling me it can talk next," he said with a hint of sarcasm in his voice.

"Oh did I forget to mention that?"
"Oh no, you've told me enough already — I know it must be cheaper than I would expect because otherwise you couldn't have afforded it, without being rude, but even so it must have cost at least £10,000."

"Well you're right," I said tantalisingly, *"it is cheaper than you would expect. Even with the Speech unit it only cost me £5,673.24 including the Chancellor's 8%."*

"How come I always end up buying you lunch?" he said.

PUNCHLINES

Software for controlling the Westrex Punch (see July issue)

Mark Colton

```
                          NAM     PTP
3200                      ORG     $3200
                   *
                   *
                   * WRITTEN BY MARK COLTON
                   * REV. 1.0    10/2/78
                   *
                   *
                   * THIS IS A COLLECTION OF
                   * ROUTINES TO CONTROL THE
                   * WESTREX PUNCH.
                   *
                   *
                   * ENTER IS THE CONTROL ROUTINE
                   * WHICH PROMPTS THE USER, INPUTS
                   * COMMANDS, AND CALLS THE
                   * APPROPRIATE ROUTINES. THE WHOLE
                   * PROGRAM STARTS AT ENTER
                   *
                   *
3200 7E 71 03  CNTRL   JMP     $7103
3203 7E E0 7E  PDATA1  JMP     $E07E
3206 39        RTS     RTS
3207 BD E1 AC  INCH1   JSR     INCH       GET COMMAND CH.
320A 81 4F             CMP A   £'O        O : GOTO DOS?
320C 27 F2             BEQ     CNTRL
320E 81 18             CMP A   £$18       CNTRL X ?
3210 26 F4             BNE     RTS        YES : START AGAIN
3212 8E A0 40  ENTER   LDS     £STACK
3215 8D 40             BSR     PCRLF
3217 CE 34 67          LDX     £LDR
321A 8D E7             BSR     PDATA1
321C 7F 33 E1          CLR     LDRFLG
321F 7F 33 E2          CLR     SFLG
3222 8D E3             BSR     INCH1      GET COMMAND CH.
3224 81 59             CMP A   £'Y        YES OR NO ?
3226 26 03             BNE     NXT
3228 7C 33 E1          INC     LDRFLG     YES : SET FLAG
322B 8D 2F     NXT     BSR     INADDR     GET SAVE ADDRESSES
322D 8D 28             BSR     PCRLF
322F CE 34 56          LDX     £SORX      PROMPT : S OR X ?
3232 8D CF             BSR     PDATA1
3234 8D D1             BSR     INCH1      GET COMMAND CH.
3236 81 58             CMP A   £'X        X FORMAT ?
3238 26 03             BNE     S
323A 7C 33 E2          INC     SFLG       YES (X FORM.)
323D 7D 33 E1  S       TST     LDRFLG     IS TITLE REQUIRED ?
3240 27 02             BEQ     NOLDR      NO
3242 8D 3B             BSR     TITLE      YES : DO IT
3244 8D 7B     NOLDR   BSR     NUL150     BEGINNING LEADER
3246 CE 33 07          LDX     £XFORM     PUNCH THE TAPE
3249 7D 33 E2          TST     SFLG       REQUIRED FORMAT
324C 26 03             BNE     X1
324E CE 33 6A          LDX     £SFORM
3251 AD 00     X1      JSR     X          DO THE PUNCH
3253 8D 6E             BSR     NULNOS     END TRAILER
3255 20 A9             BRA     CNTRL      FINISH : TO DOS
                   *
                   * PRINT CR/LF ON SYSTEM VDU/TTY
                   *
3257 CE 34 39  PCRLF   LDX     £CRTXT
325A 20 A7             BRA     PDATA1
                   *
                   * GET START & END ADDRESSES
                   * STORE THEM IN TW, ENDA
                   *
325C 8D F9     INADDR  BSR     PCRLF
325E CE 34 19          LDX     £BEGTXT
3261 8D A0             BSR     PDATA1
3263 BD E0 47          JSR     BADDR
3266 FF 33 DD          STX     TW
3269 8D EC             BSR     PCRLF
326B CE 34 29          LDX     £ENDTXT
326E 8D 93             BSR     PDATA1
3270 BD E0 47          JSR     BADDR
3273 FF 33 DF          STX     ENDA
3276 20 DF             BRA     PCRLF
                   *
                   * INPUT TITLE TEXT, PUNCH TITLE,
                   * FULL BEING THE ERROR CONDITION
                   * FOR A FULL TAPLETS BUFFER
                   * THE ROUTINE STARTS AT TITLE
                   *
3278 8D DD     FULL    BSR     PCRLF
327A CE 34 77          LDX     £FULTXT
327D 8D 84             BSR     PDATA1
327F 8D D6     TITLE   BSR     PCRLF
3281 CE 34 45          LDX     £STTXT     PROMPT : "ENTER TEXT"
3284 BD 32 03          JSR     PDATA1
3287 8D CE             BSR     PCRLF
3289 CE 33 E7          LDX     £BUFFER
328C BD E1 AC  TITL1   JSR     INCH       GET A CH.
328F 81 08             CMP A   £8         BACKSPACE ? : DELETE
3291 27 09             BEQ     TITL2
3293 A7 00             STA A   X          STORE IT
3295 08                INX
3296 08                INX
3297 8C 34 19          CPX     £ENDBUFF   IS BUFFER FULL ?
```

```
329A 27 DC             BEQ     FULL       YES
329C 8C 33 E7  TITL2   CPX     £BUFFER
329F 27 01             BEQ     TITL3
32A1 09                DEX
32A2 81 0D     TITL3   CMP A   £$D        WAS IT A CR ?
32A4 26 E6             BNE     TITL1      NO : GET ANOTHER
32A6 8D 19             BSR     NUL150     PUNCH CLEAR LEADER
32A8 CE 33 E7          LDX     £BUFFER
32AB 20 09             BRA     PUN4
32AD FF 33 E3  PUN3    STX     XTEMP1     SAVE POINTER
32B0 8D 2B             BSR     CONV       PUNCH CH. IN TAPLETS FORMAT
32B2 FE 33 E3          LDX     XTEMP1     REGAIN POINTER
32B5 08                INX
32B6 4F        PUN4    CLR A              PUNCH A NULL
32B7 BD 33 5E          JSR     PUNBYT
32BA A6 00             LDA A   X          GET NEXT CH.
32BC 81 0D             CMP A   £$D        WAS IT A CR ?
32BE 26 ED             BNE     PUN3       NO : LOOP
32C0 39                RTS
                   *
                   * PUNCH 150 NULLS
                   *
32C1 8D 0A     NUL150  BSR     SETUP
32C3 C6 96     NULNOS  LDA B   £150
32C5 4F        NUL1    CLR A
32C6 BD 33 5E          JSR     PUNBYT
32C9 5A                DEC B
32CA 26 F9             BNE     NUL1
32CC 39                RTS
                   *
                   * SETUP THE PIA, START THE
                   * PUNCH WITH CLEARED DATA LATCHES
                   *
32CD CE 80 2A  SETUP   LDX     £PIA
32D0 6F 01             CLR     1,X
32D2 C6 FF             LDA B   £$FF       ALL OUTPUTS
32D4 E7 00             STA B   X
32D6 C6 2C             LDA B   £$2C       RUNNING SETUP
32D8 E7 01             STA B   1,X
32DA 6F 00             CLR     X
32DC 39                RTS
                   *
                   *
                   * THIS ROUTINE CONVERTS THE
                   * ASCII CHARACTER IN ACCUM. A
                   * TO A POSITION IN THE LOOKUP
                   * TABLE, WHICH CORRESPONDS TO
                   * THE CHARACTER, AND PUNCHES
                   * THE 5 BYTES ON TAPE WHICH
                   * CONSTITUTE A TAPLETS CHARACTER
                   *
                   *
32DD 80 20     CONV    SUB A   £$20       REMOVE CNTRL CHS.
32DF 2D 25             BLT     CNTRCH     FROM ASCII CH. VALUE
32E1 5F                CLR B
32E2 36                PSH A              MULTIPLY BY 5
32E3 48                ASL A
32E4 59                ROL B
32E5 48                ASL A
32E6 59                ROL B
32E7 30                TSX
32E8 AB 00             ADD A   X
32EA C9 00             ADC B   £0
32EC 31                INS
32ED BB 33 E6          ADD A   TAB+1
32F0 F9 33 E5          ADC B   TAB
32F3 CE 33 DB          LDX     £XMSB
32F6 A7 01             STA A   1,X
32F8 E7 00             STA B   X
32FA EE 00             LDX     X
32FC C6 05             LDA B   £5         PUNCH THE 5 BYTES
32FE A6 00     CONV1   LDA A   X
3300 8D 5C             BSR     PUNBYT
3302 08                INX
3303 5A                DEC B
3304 26 F8             BNE     CONV1
3306 39        CNTRCH  RTS
                   *
                   *
                   * THIS ROUTINE PUNCHES TAPE
                   * IN BINARY FORMAT WITH "X1"
                   * AT THE BEGINNING OF EACH BLOCK
                   * AND "X9" AT THE END OF THE
                   * WHOLE SAVE
                   *
                   *
3307 8D 49     XFORM   BSR     SUBADD
3309 27 02             BEQ     XPUN2
330B C6 FF             LDA B   £$FF
330D 86 58     XPUN2   LDA A   £'X        PUNCH X1
330F 8D 4D             BSR     PUNBYT
3311 86 31             LDA A   £'1
3313 8D 49             BSR     PUNBYT
3315 37                PSH B
3316 5F                CLR B
3317 30                TSX
3318 8D 30             BSR     PUN        PUNCH NO. BYTES
331A 32                PUL A
```

```
331B 4C                   INC A                  INCREMENT BYTCNT
331C B7 33 E2             STA A   BYTCNT
331F CE 33 DD             LDX     £TW
3322 8D 26                BSR     PUN
3324 8D 24                BSR     PUN            PUNCH BEGA
3326 FE 33 DD             LDX     TW
3329 8D 1F       XPUN3    BSR     PUN            PUNCH TILL
332B 7A 33 E2             DEC     BYTCNT         BYTCNT=0
332E 26 F9                BNE     XPUN3
3330 FF 33 DD             STX     TW             SAVE LAST ADDR.
3333 53                   COM B
3334 37                   PSH B
3335 30                   TSX
3336 8D 12                BSR     PUN            PUNCH CHECKSUM
3338 31                   INS
3339 FE 33 DD             LDX     TW
333C 09                   DEX
333D BC 33 DF             CPX     ENDA           FINISHED?
3340 26 C5                BNE     XFORM
3342 86 58                LDA A   £'X
3344 8D 18                BSR     PUNBYT         PUNCH X9
3346 86 39                LDA A   £'9
3348 20 14                BRA     PUNBYT
334A A6 00       PUN      LDA A   X              GET BYTE
334C 8D 10                BSR     PUNBYT
334E EB 00                ADD B   X              UPDATE CHECKSUM
3350 08                   INX
3351 39                   RTS
                 *
                 *
                 * WORK OUT BLOCK LENGTH
                 * ACCORDING TO HOW MANY BYTES
                 * LEFT TO PUNCH, 256 BEING
                 * THE MAXIMUM NUMBER OF BYTES
                 * IN A BLOCK
                 *
                 *
3352 CE 33 DD    SUBADD   LDX     £TW            FIND SAVE LENGTH
3355 E6 03                LDA B   3,X
3357 E0 01                SUB B   1,X
3359 A6 02                LDA A   2,X
335B A2 00                SBC A   X
335D 39                   RTS
                 *
                 *
                 * PUNCH THE BYTE IN THE
                 * "A" REGISTER
                 *
335E 7D 80 2A    PUNBYT   TST     PIA            CLR FLAGS
3361 7D 80 2B    PUN1     TST     PIA+1          WAIT FOR COMPLETE
3364 2A FB                BPL     PUN1
3366 B7 80 2A             STA A   PIA            PUNCH BYTE
3369 39                   RTS
                 *
                 * PUNCH AN "S" FORMAT TAPE IN
                 * BLOCKS, BEGINNING EACH BLOCK
                 * WITH AN "S1", SEPARATING EACH
                 * BLOCK WITH A CR/LF, 4 NULLS
                 * ENDING THE SAVE WITH AN "S9"
                 *
336A 8D E6       SFORM    BSR     SUBADD
336C 26 04                BNE     SPUN2
336E C1 10                CMP B   £16
3370 25 02                BCS     SPUN3
3372 C6 F        SPUN2    LDA B   £15
3374 5C          SPUN3    INC B
3375 F7 33 E2             STA B   BYTCNT
3378 CB 03                ADD B   £3
337A F7 33 E1             STA B   FRMCNT
337D CE 34 39             LDX     £CRTXT         PUNCH CR/LF/NULLS/S1
3380 8D 52                BSR     PUNDAT
3382 08                   INX
3383 8D 4F                BSR     PUNDAT
3385 5F                   CLR B
3386 CE 33 E1             LDX     £FRMCNT        PUNCH NO. BYTES
3389 8D 2B                BSR     PUN2
338B CE 33 DD             LDX     £TW
338E 8D 26                BSR     PUN2           PUNCH ADDRESS
3390 8D 24                BSR     PUN2
3392 FE 33 DD             LDX     TW
3395 8D 1F       SPUN4    BSR     PUN2           PUNCH DATA
3397 7A 33 E2             DEC     BYTCNT
339A 26 F9                BNE     SPUN4
339C FF 33 DD             STX     TW
339F 53                   COM B
33A0 37                   PSH B
33A1 30                   TSX
33A2 8D 12                BSR     PUN2           PUNCH CHECKSUM
33A4 33                   PUL B
33A5 FE 33 DD             LDX     TW
33A8 09                   DEX
33A9 BC 33 DF             CPX     ENDA           END?
33AC 26 BC                BNE     SFORM
33AE 86 53                LDA A   £'S
33B0 8D AC                BSR     PUNBYT         PUNCH S9
33B2 86 39                LDA A   £'9
33B4 20 A8       PUN5     BRA     PUNBYT
33B6 EB 00       PUN2     ADD B   X              UPDATE CHECKSUM
33B8 A6 00       PUN2H    LDA A   X              CONVERT HEX TO
33BA 8D 05                BSR     PUNHL          ASCII & PUNCH
33BC A6 00                LDA A   X
33BE 08                   INX
33BF 20 04                BRA     PUNHR
                 *
                 * CONVERT HEX TO ASCII
                 * PUNCH ASCII ON TAPE
                 *
33C1 44          PUNHL    LSR A
33C2 44                   LSR A
33C3 44                   LSR A
33C4 44                   LSR A
33C5 84 0F       PUNHR    AND A   £$F
33C7 8B 30                ADD A   £$30
33C9 81 39                CMP A   £$39
33CB 23 02                BLS     PUNCH
33CD 8B 07                ADD A   £$7
33CF 20 E3       PUNCH    BRA     PUN5
33D1 8D FC       DAT1     BSR     PUNCH          PUNCH ON TAPE DATA
33D3 08                   INX                    POINTED TO BY X REGISTER
33D4 A6 00       PUNDAT   LDA A   X
33D6 81 04                CMP A   £4
33D8 26 F7                BNE     DAT1
33DA 39                   RTS
                 *
                 * EQUATES, RMB'S
                 *
E1AC             INCH     EQU     $E1AC
802A             PIA      EQU     $802A
E047             BADDR    EQU     $E047
A040             STACK    EQU     $A040
33DB             XMSB     RMB     1
33DC             XLSB     RMB     1
33DD             TW       RMB     2
33DF             ENDA     RMB     2
33E1             LDRFLG   EQU     *
```

```
33E1             FRMCNT   RMB     1
33E2             SFLG     EQU     *
33E2             BYTCNT   RMB     1
33E3             XTEMP1   RMB     2
33E5 34 83       TAB      FDB     TABLE
33E7             BUFFER   RMB     50
3419             ENDBUFF  EQU     *
                 *
                 * TEXT STRINGS
                 *
3419 53          BEGTXT   FCC     /START ADDRESS $/
3428 04                   FCB     4
3429 45          ENDTXT   FCC     /END ADDRESS   $/
3438 04                   FCB     4
3439 0D          CRTXT    FCB     $D,$A,$15,,,,,4,'S,'1,4
3444 04                   FCB     4
3445 49          STTXT    FCC     /INPUT TITLE TEXT/
3455 04                   FCB     4
3456 53          SORX     FCC     /S OR X FORMAT ? /
3466 04                   FCB     4
3467 54          LDR      FCC     /TAPLETS TITLE ?/
3476 04                   FCB     4
3477 42          FULTXT   FCC     /BUFFER FULL/
3482 04                   FCB     4
                 *
                 * LOOKUP TABLE
                 *
3483 00          TABLE    FCB     $00,$00,$00,$00,$00,$00,$00,$5F
```

```
348B 00                   FCB     $00,$00,$00,$07,$00,$07,$00,$14
3493 7F                   FCB     $7F,$14,$7F,$14,$24,$2A,$7F,$2A
349B 12                   FCB     $12,$23,$13,$08,$64,$62,$36,$49
34A3 56                   FCB     $56,$20,$50,$00,$00,$07,$00,$00
34AB 1C                   FCB     $1C,$22,$41,$00,$00,$00,$00,$41
34B3 22                   FCB     $22,$1C,$22,$14,$7F,$14,$22,$08
34BB 08                   FCB     $08,$3E,$08,$08,$00,$80,$60,$00
34C3 00                   FCB     $00,$08,$08,$08,$08,$08,$00,$00
34CB 40                   FCB     $40,$00,$00,$20,$10,$08,$04,$02
34D3 3E                   FCB     $3E,$51,$49,$45,$3E,$00,$42,$7F
34DB 40                   FCB     $40,$00,$62,$51,$49,$49,$46,$21
34E3 41                   FCB     $41,$49,$4D,$33,$18,$14,$12,$7F
34EB 10                   FCB     $10,$27,$45,$45,$45,$39,$3C,$4A
34F3 49                   FCB     $49,$49,$31,$01,$71,$09,$05,$03
34FB 36                   FCB     $36,$49,$49,$49,$36,$46,$49,$49
3503 29                   FCB     $29,$1E,$00,$00,$14,$00,$00,$00
350B 80                   FCB     $80,$68,$00,$00,$08,$14,$22,$41
3513 00                   FCB     $00,$14,$14,$14,$14,$14,$00,$41
351B 22                   FCB     $22,$14,$08,$02,$01,$59,$05,$02
3523 3E                   FCB     $3E,$41,$5D,$59,$4E,$7C,$12,$11
352B 12                   FCB     $12,$7C,$7F,$49,$49,$49,$36,$3E
3533 41                   FCB     $41,$41,$41,$22,$7F,$41,$41,$41
353B 3E                   FCB     $3E,$7F,$49,$49,$49,$41,$7F,$09
3543 09                   FCB     $09,$09,$01,$3E,$41,$41,$51,$71
354B 7F                   FCB     $7F,$08,$08,$08,$7F,$00,$41,$7F
3553 41                   FCB     $41,$00,$20,$40,$40,$40,$3F,$7F
355B 08                   FCB     $08,$14,$22,$41,$7F,$40,$40,$40
3563 40                   FCB     $40,$7F,$02,$0C,$02,$7F,$7F,$04
356B 08                   FCB     $08,$10,$7F,$3E,$41,$41,$41,$3E
3573 7F                   FCB     $7F,$09,$09,$06,$00,$3E,$41,$51
357B 21                   FCB     $21,$5E,$7F,$09,$19,$29,$46,$26
3583 49                   FCB     $49,$49,$49,$32,$01,$01,$7F,$01
358B 01                   FCB     $01,$3F,$40,$40,$40,$3F,$1F,$20
3593 40                   FCB     $40,$20,$1F,$7F,$20,$18,$20,$7F
359B 63                   FCB     $63,$14,$08,$14,$63,$03,$04,$78
35A3 04                   FCB     $04,$03,$61,$51,$49,$45,$43,$00
35AB 7F                   FCB     $7F,$41,$41,$41,$01,$04,$08,$10
35B3 20                   FCB     $20,$00,$41,$41,$41,$7F,$08,$04
35BB 02                   FCB     $02,$04,$08,$80,$80,$80,$80,$80
35C3 70                   FCB     $70,$48,$48,$28,$78,$38,$44,$44
35CB 7C                   FCB     $7C,$00,$7F,$44,$44,$38,$00,$38
35D3 44                   FCB     $44,$44,$44,$00,$38,$44,$44,$7F
35DB 00                   FCB     $00,$38,$54,$54,$48,$00,$08,$7E
35E3 09                   FCB     $09,$02,$00,$4C,$92,$92,$7C,$00
35EB 7F                   FCB     $7F,$04,$04,$78,$00,$3A,$40,$40
35F3 20                   FCB     $20,$00,$40,$80,$80,$7A,$00,$7F
35FB 10                   FCB     $10,$28,$44,$00,$3F,$40,$40,$20
3603 00                   FCB     $00,$7C,$04,$18,$04,$7C,$78,$04
360B 04                   FCB     $04,$78,$00,$38,$44,$44,$38,$00
3613 FC                   FCB     $FC,$14,$14,$08,$00,$08,$14,$14
361B F8                   FCB     $F8,$00,$7C,$04,$04,$04,$00,$48
3623 54                   FCB     $54,$54,$24,$00,$04,$3F,$44,$20
362B 00                   FCB     $00,$3C,$40,$40,$3C,$00,$1C,$20
3633 40                   FCB     $40,$20,$1C,$3C,$40,$30,$40,$3C
363B 44                   FCB     $44,$28,$10,$28,$44,$4C,$90,$90
3643 7C                   FCB     $7C,$00,$44,$64,$54,$4C,$00
                          END

NO ERROR(S) DETECTED

SYMBOL TABLE:
```

```
BADDR   E047   BEGTXT 3419   BUFFER 33E7   BYTCNT 33E2   CNTRCH 3306
CNTRL   3200   CONV   32DD   CONV1  32FE   CRTXT  3439   DAT1   33D1
ENDA    33DF   ENDBUF 3419   ENDTXT 3429   ENTER  3212   FRMCNT 33E1
FULL    3278   FULTXT 3477   INADDR 325C   INCH   E1AC   INCH1  3207
LDR     3467   LDRFLG 33E1   NOLDR  3244   NUL1   32C5   NUL150 32C1
NULNOS  32C3   NXT    322B   PCRLF  3257   PDATA1 3203   PIA    802A
PUN     334A   PUN1   3361   PUN2   33B6   PUN2H  33B8   PUN3   32AD
PUN4    32B6   PUN5   33B4   PUNBYT 335E   PUNCH  33CF   PUNDAT 33D4
PUNHL   33C1   PUNHR  33C5   RTS    3206   S      323D   SETUP  32CD
SFLG    33E2   SFORM  336A   SORX   3456   SPUN2  3372   SPUN3  3374
SPUN4   3395   STACK  A040   STTXT  3445   SUBADD 3352   TAB    33E5
TABLE   3483   TITL1  328C   TITL2  329C   TITL3  32A2   TITLE  327F
TW      33DD   X1     3251   XFORM  3307   XLSB   33DC   XMSB   33DB
XPUN2   330D   XPUN3  3329   XTEMP1 33E3
```

The Applesoft Touch

Apple II Re-review

H. N. Dobbs
Newtown School, Waterford, Eire.

I very much enjoyed, and learnt a lot from, the review of Apple II by John Coll and Charles Sweeten in the August PCW. But now we have APPLESOFT] [and the floppy disk system is also available.

One minor point from the review: a program line such as IF 2=3 THEN A=3:A=5 will give A=5 in INTeger BASIC, but will not give A=3 in APPLESOFT. It will of course leave A unchanged, with a default value of 0 unless it has been previously defined.

Integer Basic is Powerful

A major point, though, which I want to take up, is the reference to INTeger BASIC as a "minimal" BASIC. I find it an extremely flexible and powerful language; in some ways even more flexible perhaps than APPLESOFT. There is one amazing omission from it, the absence of READ/DATA.

It does have a number of features which are not found in many other BASICs, even quite high-powered ones. Apart from MOD, PEEK, POKE and CALL, there are all the graphic terms: GR(aphics), COLOR (sic), PLOT(point), VLIN (vertical line), HLIN, AT, SCRN (reads the COLOR of any given spot), TEXT (to get back to normal script); AND, OR, and NOT are both logical and "bitwise" operators in INT as well in APPLESOFT, and can be used to build up complicated conditions; TAB and VTAB (vertical) control the cursor directly and make nonsense of the complaint about lacking the "ability to remain on the same line after an input". In any case BASICs generally use a Return to mark end of INPUT, which leaves one on the next line anyway. In INT one can PEEK to see which line the cursor is on, POKE or VTAB it onto the previous line, TAB it across to the desired place, CALL if desired to delete the input, or simply carry on from there. Alternatively, one can use PEEK and POKE to *simulate* an INPUT.

INT has the possibility of multi-statement lines which are read in the same way as if they were consecutive lines, though this can be used only in program mode (APPLESOFT allows them in direct mode as well). It has instant error recognition (unlike APPLESOFT) and a different LIST format which makes editing easy (the lines LISTed are inset, with the effect that a large number of potential spaces are inserted). Almost any statement can be used in direct mode.

Most importantly, IF .... THEN can be followed by any statement, GOTO and GOSUB can be followed by expressions as well as by numbers, and almost anything (apart from BASIC words, or words containing them) can be used as a variable. *This means that one can write a sort of assembler* BASIC, which is a higher-level language than BASIC itself. It is more easily read and less open to error. For example, here are a few typical lines:

```
10 WAIT = 200 : BOARD = .16384 : CLEAR = .936 : CHOICE = 100 : STROBE = . . . . etc.
100 CALL CLEAR : PRINT "HIT ANY LETTER KEY": GOSUB WAIT : IF KEY > 64 AND KEY < 91 THEN GOTO (KEY * 10 + 50) : PRINT "LETTER KEYS ONLY PLEASE" : GOSUB PAUSE : GOTO CHOICE
200 KEY = PEEK(BOARD) : IF KEY < 128 THEN GOTO WAIT : POKE STROBE, 0 : KEY = KEY AND 127 : RETURN
700 CALL CLEAR : VTAB MIDSCREEN : PRINT "A IS FOR APPLE" : GOSUB PAUSE : GOTO CHOICE
710 . . . . . . . . . . . . . . . . . .: PRINT" B IS FOR BASIC" . . . . etc . . . . . . .
```

Here BOARD and CLEAR and STROBE are *numerical* memory addresses which are easily confused or typed with transposed figures, etc., and not easily recognised by anyone reading the program — whereas the words used for them should be much easier to follow; WAIT, CHOICE, PAUSE (not LISTed), and (KEY * 10 + 50) are program line numbers; and using these means a routine such as CHOICE, or a subroutine such as WAIT (for a key to be pressed) can be relocated simply by changing the line numbers, and changing the initial assignment of value for the word (line 10), without having to go right through the program changing every single reference to that section. Clearly this also gives the effect of ON ..... GOTO ..... , and of ON .... GOSUB, without the trouble of writing out all the line numbers (in this program, KEY * 10 + 50 can have any value from 700 to 950 in steps of 10, which would take up a lot of space and time!). KEY and MIDSCREEN are variables; by the time it emerges from the WAIT subroutine, KEY is the ASCII value of the key pressed, while MIDSCREEN is about 11. Vary according to taste.

Using logical variables, one can even end up with lines such as

```
100 IF TIRED THEN GOTO BED
200 IF HUNGRY THEN GOTO EAT,
```

where TIRED and HUNGRY are variables, IF checks whether they are nonzero, and EAT and BED are line numbers. THEN is, I think, optional in this case, but GOTO is not because THEN expects either a BASIC word or an actual number.

The PEEK and POKE and USR addresses given in the APPLESOFT manual naturally work as PEEK and POKE and CALL addresses for INT as well (with the exception of Hires routines if these have not been loaded). The TRACE is certainly extremely fast, but one can always cut in (ctrl C), read it, and then CONTinue. This doesn't work for LIST — which can be embarrassing if, for example, one accidentally LISTs APPLESOFT. In fact I have been so annoyed by the problems of LISTing long programs that I am writing a short program which can be CHAINed onto a long program in INT and used to break it up into chunks that I can handle. Incidentally the problem is not so serious in APPLESOFT ∫ [, as it has a reserved variable SPEED which can be used to slow down LISTing and general operation as desired — it can produce a print rate of maybe 1 cps, which is slow enough for anyone.

Another facility in INT, which compares favourably with Tandy Level I for instance, is string variable handling. For a start, string variable names are as flexible as numeric variable names. String comparisons are possible (unlike Tandy L. 1). Strings are in fact matrices and must be dimensioned, but this means that individual elements (letters) can be tested or inserted or changed; and one can address a group of elements such as A$(5,7) — in other words the string-variable-as-matrix corresponds to the MID$( ) function available in more extended BASIC such as APPLESOFT.

So much for the "minimal" BASIC!

Applesoft] [

It was annoying (when I met APPLE] [first) that APPLESOFT differed in so many ways from INT, viz. PLTP instead of PLOT, USR instead of CALL, etc. — so that one had to learn two different languages. How nice it would be (to put it mildly) if INT were a subset of APPLESOFT.

Well, APPLESOFT] [is available now, in three versions: on tape, on disk, and in ROM, and I am glad to say that it has taken over the old INT words for graphics, and uses CALL for machine-language subroutines. Furthermore it has new "Hires" graphics words which are formed by prefixing 'H' onto the "Lores" graphics words: HGR, HCOLOR, and HPLOT (. . . , . . .); however, instead of the HLIN and VLIN . . . , . . . AT . . . , which are irrelevant to Hires, it has HPLOT (X,Y) to (A,B) where (X, Y) and (A,B) are two points to be joined by a straight line. This means that the Hires routines are loaded along with APPLESOFT II (in the tape or disk form), and there may be some difficulty in reaching them from INT. I *think* the following sequence works: RUN APPLESOFT, Reset, ctrl B, HIMEM:8192. It may be possible to do it by typing FP, Return, INT, Return, with the disk version or there may even be an easier way. The 'HIMEM' in the first suggested method is required because Reset sets the upper limit of working memory to the maximum available, while Hires requires it to be set at 8K — that is supposing that Hires hasn't changed significantly in the past few months.

The ROM version of APPLESOFT II is on a card which is inserted in one of the I/O Ports in the same way as the UHF modulator card, the Disk Operating card, or the communications card; it is controlled by a hardware on/off switch mounted on the card so that it can be reached through the back panel. Goodness knows whether INT can reach Hires in this version, I doubt it! However anyone who has the old Hires Demo tape can always load the machine-language routines from that.

To anyone who has used the cassette filing 'system', the new floppy disk system must come as an immense relief. It handles at least three types of storage: programs, data files, and binary files. The data files were not working when I had a chance to use the system (mid July), because of a fault in the DOS, but this has since been fixed. Binary files are a pure memory dump, and might contain anything, from machine-language programs to screen graphics. I didn't have time to try these, but I have since seen some extremely impressive disk demos which included one where Hires pictures were re-created from disk storage.

Binary files also offer the advantage, for numeric data, of being able to pack data much more densely than can be done in data files. One of my students (actually I teach him chemistry and maths, not computer studies) is working on a project which involves enormous masses of numbers in the range 0 to 15. These would require two bytes apiece in a data file, at least; or one byte if converted to hex code before storage. In a binary file they can be packed into nibbles — two to each byte. Since he is dealing with something of the order of 640,000 numbers, this improvement in density makes the difference between about fifteen disks (impossible) and about five (possible).

Programs are SAVEd by name (e.g. SAVE MONEY) and are listed in the CATALOG with an A for APPLESOFT or an I for INT, and a number which I think is the number of sectors occupied.

CATALOG	
I	043 APPLESOFT
A	007 UP
I	002 WILD
I	001 VANISHING TRICK
B	005 HOUSE PLAN
T	003 USEFUL ADDRESSES
I	003 FORESEEN

Figure 1 : sample CATALOG

```
10  D$ = ""
20  PRINT D$; "DELETE VANISHING
                        TRICK"
30  PRINT D$; "LOAD UP"
40  END

Note: D$ is "ctrl D"
```

Figure 2 : LISTing for VANISHING TRICK

Figure 1 shows a typical CATALOG. HOUSE PLAN is a binary file containing about 1K bytes (judging by the length of APPLESOFT); might well be a Lores graphics picture. USEFUL ADDRESSES is a text or data file. APPLESOFT is one of the programs on the MASTER DISKETTE; someone has LOADed it, changed disks, and SAVEd it on this one for convenience. One can now start from the beginning: 1 . . . power on; 2 . . . Reset (into Monitor); 3 . . . ctrl B, Return(into INT); 4 . . . PR# 6 (activates the Disk Operating System, if the disk connector is plugged into PoRt #6), Return . . . . . and in three more keystrokes one can be in APPLESOFT] [.

APPLESOFT is in fact APPLESOFT] [, but it can't be called that in the CATALOG, because square brackets can only be produced using a PRINT statement and CHR$, which doesn't exist in INT. One can now type LOAD APPLESOFT, Return, wait for a few seconds while it is loaded, and spare a thought for all cassette users who still have 1¼ minutes to wait before typing RUN, Return.

However, we can operate at a higher technological level still. Forget all about APPLESOFT being a program to be LOADed (unless you want to LIST it, of course). Just type FP (for Floating Point), Return. APPLESOFT is LOADed and RUN automatically. More is to follow! If you have been working in INT and you want to LOAD an existing program in FP, such as UP, you simply LOAD UP; if you haven't already RUN APPLESOFT in any way, this is done for you before your own program is LOADed. What happens if you haven't got APPLESOFT on the disk, I don't know. You don't even have to LOAD; for instance you can just RUN WILD (an INT program). This has the combined effect of switching to INT, LOADing your program, and RUNning it. Most impressive.

The disk system is extremely simple; the only connection for the disk drive is a ribbon cable with a connector which plugs into any of the I/O slots in the main board (apart from # 7, which is reserved for the UHF modulator for TV output). PR # 6 (etc.) is required to 'boot' the DOS when starting, and also after a Reset or other careless descent into Monitor. It is possible to get into Monitor without losing the DOS, if desired.

My only objections to the system are purely physical. A minor one is the temptation to leave the disk drive attached when moving APPLE II around, which means that it won't fit into the carrying case. The more serious one is this: to change a disk one raises a flap on the front of the drive box, removes disk, puts in new disk, forgets to close flap. The drive should, but doesn't, refuse to 'work'.

When the CATALOG for a disk gets longer than a 'page' it displays the first page and then waits for any keystroke to display the next page. As the keyboard buffer is cleared in the process, if one is not expecting this it can lead to errors such as UN FORESEEN. To avoid this, hit the space bar before typing anything.

With a bit of jugglery it is possible to SAVE a program with its name in reversed or flashing script; Apple have done this with the first sector on their MASTER DISKETTE, a program entitled, in black-on-white, HELLO (I think that is responsible for the advertisement which appears whenever the M.D. is booted). That is pure and obvious gimmickry; but it is also possible to include non-printing characters such as "ctrl L" in the name of a program; this is infuriating for those who come after, who see in the CATALOG a program, apparently called FORESEEN (but actually FORESEctrLEN), try to LOAD it, and are told by the DOS that there is no such program. This does provide an almost genius-proof security lock, but I have worked out a way to break it (which I'm not telling you!)

Disk commands are usually used in direct mode, but can be used in program mode so long as they are prefixed by a ctrl D. As ctrl D is non-printing, it is useful to define it as a string: D$ = "ctrl D" – which looks exactly like a null string, " ", but behaves differently. Just for fun, see figure 2 for the VANISHING TRICK : if RUN, it disappears from the CATALOG and from program memory, since LOADing a program deletes any previous one.

It is hopeless trying to keep up with Apple; I am aware that even this re-review is already out of date, as I cannot deal with the ROM version of APPLESOFT][nor can I do justice to the excellent disk demo programs which I have only briefly seen parts of. If I waited long enough for that, Apple would have produced their double-sided double-density half-megabyte disk, and perhaps even APPLESOFT III (I hear they are abandoning Microsoft and writing their own this time). Not to mention speech recognition and simulation, light pen . . . .

The two versions I have worked with, and the third which I have seen demonstrated, have been on loan from Dr Lewis Leith, 17 Knockcullen Park, Templeogue, Dublin (one of the Irish distributors). Dr Leith and others are working to fill the "software gap", and already a range of business and educational programs are available. For example, there is a physics demonstration package (probably available only from him at present) which uses Hires graphics to show things such as beats (interaction between two wave forms), and currents flowing in resonant circuits or through thermionic valves.

I have been working on a program which will generate sequences of numbers (Arithmetic, Geometric, and at least eight other types) and test one's ability to recognise and continue the pattern in each. I think that this type of test may give some indication of an aptitude for maths and/or science. Also I suspect that practice with it could raise one's score on I.Q. tests dramatically, since pattern-recognition is regarded as an important factor in "intelligence".

Now to finish up a little graphic program which may give someone an idea about how those horrible masses of reversed or flashing letters appear when one reverts from graphics to text . . . .

```
10 GR : COLOR = 2
20 FOR A = 1 TO 20
30 HLIN 0, 39 AT A*2 - 1
40 NEXT A
50 COLOR = 4
60 HLIN 8, 20 AT 15
70 HLIN 8, 20 AT 21
80 HLIN 8, 20 AT 23
90 COLOR = 5
100    HLIN 8,20 AT 17
110    HLIN 8,20 AT 19
120    HLIN 8,20 AT 22
130    COLOR = 12
140    HLIN 8,20 AT 20
150    PRINT: PRINT "HIT ANY KEY
       WHEN READY":PRINT:PRINT
160    IF PEEK (-16384) < 128 THEN
                                160
170    POKE - 16368,0
180    TEXT
190    END
```

PCW The Apple II was reviewed in Vol 1 No 4 **PCW**

THE SORCERER HAS ARRIVED

Introducing the personal computer you've waited for. The Exidy Sorcerer.

I didn't buy my personal computer until I found the one that had all the features I was looking for.
The Exidy Sorcerer does everything I wanted to do and a few things I never dreamed of.
It isn't magic. Exidy started with the best features of other computers, added some tricks of their own, and put it all together with more flexibility than ever before available. Presto! My reasons for waiting just disappeared.

I wanted pre-packed programs. Software on inexpensive cassette tapes for the Sorcerer is available from Exidy and many other software makers.

I wanted user programmability The Sorcerer's unique plug-in ROM PAC™ Cartridges contain programming languages such as Standard (Altair 8k*) BASIC, Assembler and Editor (so I can develop system software), operating systems such as DOS (so I can also use FORTRAN and COBOL) and applications packages such as Word Processor.

*Altair is a trademark of Pertec Computer Corp.

I wanted Graphics, and the Sorcerer is super. Its 256 character set – more than any other personal computer – includes 128 graphic symbols that I can define.

I wanted high resolution video. With 122,880 points in a 512 x 240 format, I get the most detailed illustrations.

I wanted to display more information. The Sorcerer displays 1920 characters in 30 lines of 64 characters – equal to a double-spaced typed page.

I wanted a full, professional keyboard. The Sorcerer's 79-key data processing keyboard provides designated graphics, the complete ASCII character set in upper and lower case, and a 16-key numeric pad.

I wanted memory. The 12k of ROM holds a Power-On Monitor and Standard BASIC; 32k of RAM is supplied on board.

I wanted expandability. Serial and parallel I/Os are built in, and the optional 6-slot S-100 expansion unit lets my system grow.

I wanted a computer that's easy enough for children to use. I just connect my Sorcerer to a video display and a cassette tape recorder, and if I have any questions the easy-to-understand Operation and BASIC Programming manuals have the answers.

I wanted to buy from an experienced Manufacturer. In five years Exidy has become the third largest producer of microprocessor-based video arcade games.

I wanted to spend less than £1,000. (This is where COMP. does a little magic). My Sorcerer cost me £950!.
Now, what are you waiting for?
Call COMP. on 01-441 2922 or write to

COMP COMPUTER COMPONENTS
14 Station Road,
New Barnet, Herts. EN5 1QW.

(Price shown ex. VAT)

DEALER ENQUIRIES INVITED, A LIMITED NUMBER OF DEALERSHIPS STILL AVAILABLE.