50p

Practical Computing

An ECC Publication. Volume 1 Issue 3

November 1978

Shopping for computers — Buyers' Guide

Teach yourself programming

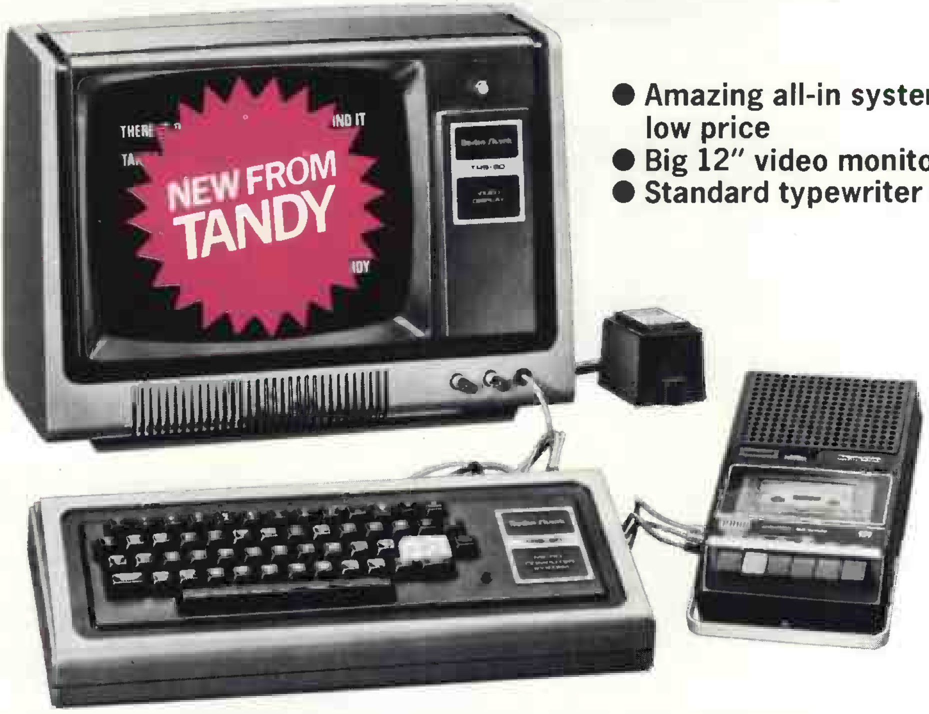
We test the Tandy

Computing in the classroom

Projects for your Kim

TRS-80—THE BIGGEST NAME IN LITTLE COMPUTERS

Complete and Ready to Go NOW!



- Amazing all-in system at a low price
- Big 12" video monitor
- Standard typewriter Keyboard

The TRS-80 computer system is 100% wired, and tested for 240 VAC so you can put it to work immediately! It's ideal for finances, education, accounting, laboratory—even games at home. Below are four complete TRS-80 systems incorporating different combinations of RAM (4K and 16K) and ROM (Level-1 and Level-11 BASIC). Choose the one that's right for you. Expansion is easy due to TRS-80's modular design. All TRS-80 systems below include a 12" video monitor, Realistic CTR-41 battery/AC cassette recorder, power supply, 232-page user's manual, and a 2-game cassette.

Level-1 BASIC system

4K ROM, 4K RAM
Cat.No. 26-1001, 26-1201, 14-841 **£499**

Level-1 with 16K RAM

4K ROM, 16K RAM
Cat.No. 26-1003, 26-1201, 14-841 **£728**

Level-11 BASIC system

12K ROM, 4K RAM
Cat.No. 26-1004, 26-1201, 14-841 **£578**

Level-11 with 16K RAM

12K ROM, 16K RAM
Cat.No. 26-1006, 26-1201, 14-841 **£807**

You can see the TRS-80 at these Tandy Stores and Dealerships

BASINGSTOKE	22 London Street, Basingstoke. Tel: 52795
BIRMINGHAM AREA	Bilston Road, Wednesbury. Tel: (021) 556 6101 528 The Bridge, Bull Ring Shopping Centre, Birmingham. Tel: (021) 643 3876 57-58 Dale End, Birmingham. Tel: (021) 236 4744
BOLTON	5 Nelson Square, Bolton. Tel: 386538
BOURNEMOUTH	134 Commercial Road, Bournemouth. Tel: 293606
BRIGHTON	70 London Road, Brighton. Tel: 693446
BRISTOL	5 Badminton Road, Downend, Bristol. Tel: 561917
BROMSGROVE	6-8 St. John Street, Bromsgrove. Tel: 71800
CAMBRIDGE	1 Emmanuel Street, Cambridge. Tel: 68155
CHESTER	Kwik Save Centre, Sealand Road, Chester. Tel: 375794
COVENTRY	4 Hales Street, Coventry. Tel: 22894
DARLINGTON	15-16 Priestgate, Darlington. Tel: 58676
DERBY	33 Victoria Street, Derby. Tel: 371066
DONCASTER	32-34 Kingsgate, Waterdale Centre, Doncaster. Tel: 21992
GLASGOW	Victor Morris, Glassford Street, Glasgow. Tel: (041) 221 8958
GLOUCESTER	47 Kings Square, Clarence Street, Gloucester. Tel: 31323
LEEDS	72 Merrion Centre, Leeds. Tel: 42520
LIVERPOOL	168 St. John's Centre, Market Way, L'pool. Tel: (051) 708 0161
LONDON AREA	The Colonnades, Porchester Road, Queensway, Bayswater, W.2. Tel: (01) 221 5317 7 Embassy Court, Welling. Tel: (01) 303 5483 124-126 The Broadway, Wimbledon, S.W.19. Tel: (01) 542 6389 6 New Broadway, W.5. Tel: (01) 579 1320 21 Sentinel Square, Hendon, N.W.4. Tel: (01) 202 7331
LUTON	157 Dunstable Road, Luton. Tel: 36159
MANCHESTER	4-8 The Mall Shopping Centre, Hyde. Tel: (061) 368 0268 13-15 The Mall Shopping Centre. Sale. Tel: (061) 973 0371 The Arndale Centre, Stretford. Tel: (061) 865 8214
NEWCASTLE-UPON-TYNE	23 Newgate Centre, Newgate, Newcastle. Tel: 21478
NORTHAMPTON	Weston Favel Shopping Centre, Northampton. Tel: 326354
NOTTINGHAM	126-128 Front Street, Arnold, Nottingham. Tel: 202626
SUTTON	206 High Street, Sutton, Surrey. Tel: (01) 643 8687
SWANSEA	Radio Supplies, 80 Gower Road, Sketty, Swansea. Tel: 24140
WITNEY	Witney Audio, 29 Corn Street, Witney. Tel: 2414
WOLVERHAMPTON	1 Market Street, Wolverhampton. Tel: 21148

Or contact Computer Sales Department, Tandy Corporation, Bilston Road, Wednesbury, West Midlands. WS10 7JN. Tel: (021) 556 6101

TANDY

WORLDWIDE ELECTRONICS

● Circle No. 101

Practical Computing

Managing Editor
Dennis Jarrett

Computabits Editor
Nick Hampshire

Production Editor
Harold Mayes

Advertisement Manager
Erica Gibson

Subscription Manager
Annabel Hunt

Publisher
Wim Hoeksma

Company Secretary
Carole Fancourt

Managing Director
Richard Hease

Editorial, Advertising and subscriptions: 01-278 9517.

Practical Computing is published by ECC as a subsidiary of WHICH COMPUTER? Ltd at its registered office, 2 Duncan Terrace, London, N1, and printed by Bournehall Press Ltd, Welwyn Garden City. Distributed by Moore Harness Ltd, 31 Corsica Street, London, N5.
© Practical Computing 1978
ISSN 0141-5433.

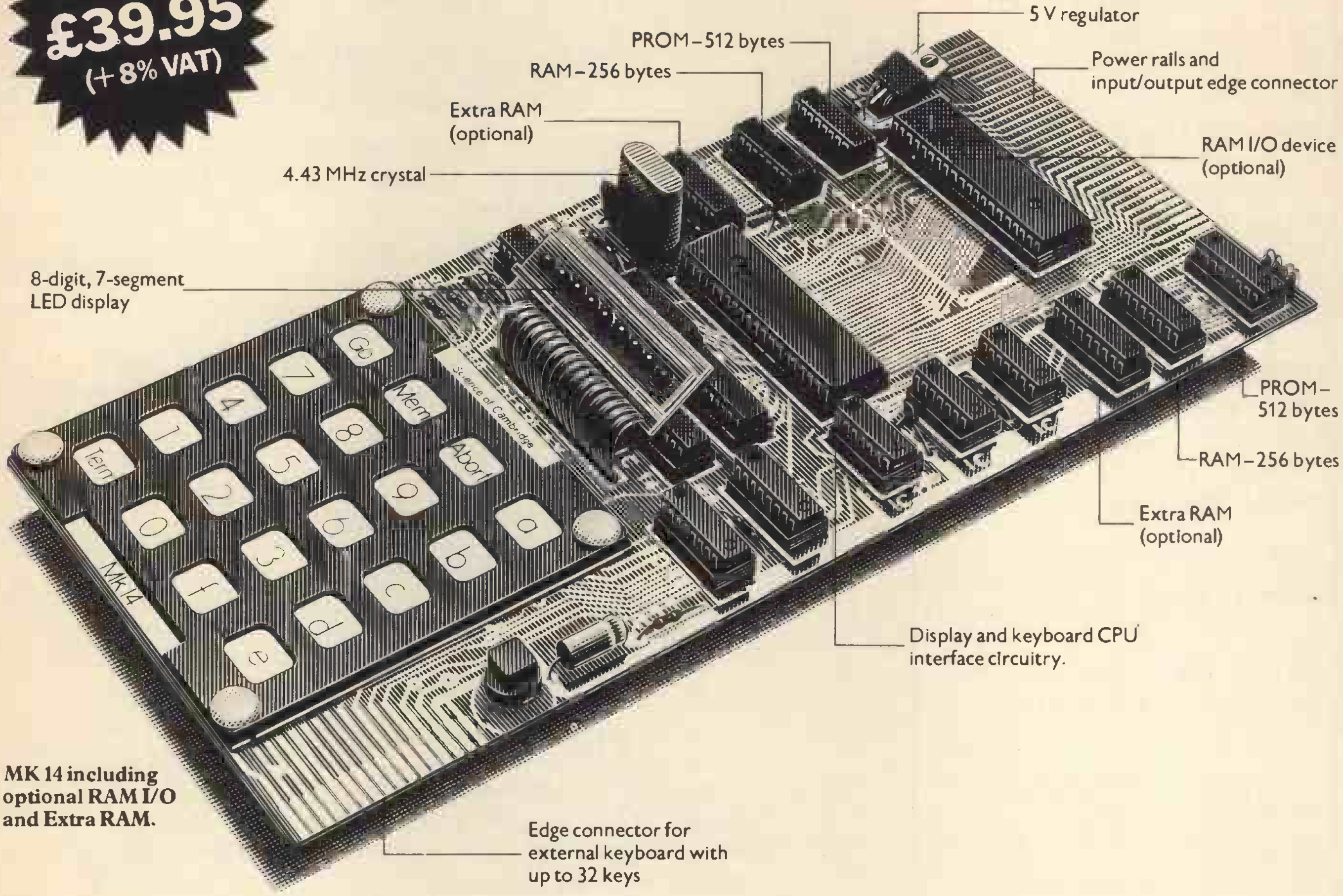
Subscription rates: Single copy: 50p. Subscriptions: U.K., £6 per annum (including postage); overseas, £12 (including airmail postage).

Every effort has been made to ensure accuracy of articles and program listing. Practical Computing cannot, however, accept any responsibility whatsoever for any errors.

Contents

From Science of Cambridge: the new MK 14.

only £39.95 (+8% VAT)


5 V regulator
PROM – 512 bytes
RAM – 256 bytes
Power rails and input/output edge connector
Extra RAM (optional)
RAM I/O device (optional)
4.43 MHz crystal
8-digit, 7-segment LED display
Go
Mem
Abort
Term
7
8
9
4
5
6
1
2
3
0
a
b
c
d
e
f
Mk 14
Science of Cambridge
PROM – 512 bytes
RAM – 256 bytes
Extra RAM (optional)
Display and keyboard CPU interface circuitry.
Edge connector for external keyboard with up to 32 keys

MK 14 including optional RAM I/O and Extra RAM.

MK 14 – a complete computer for £39.95 (+8% VAT)

The MK 14 is a complete microcomputer with a keyboard, a display, 8 x 512-byte pre-programmed PROMs, and a 256-byte RAM programmable through the keyboard.

As such the MK 14 can handle dozens of user-written programs through the hexadecimal keyboard. (20 sample programs are provided in the Manual – which also contains comprehensive building instructions, and instructions on program-writing.)

Yet in kit form (which can be assembled by any fairly experienced kit-builder), the MK 14 costs only £39.95 (+£3.20 VAT, and p&p).

But that's only the start....

The memory capacity of the basic kit is surprisingly powerful – but every computer owner, from a schoolboy to a multi-national corporation, soon feels the need for more memory.

With the MK14, it's yours!

Optional extras include an additional 256-byte RAM, and a 16-line external input/output device (allowed for on the PCB) which give a further 128 bytes of RAM.

And the next step?

The next step is to add your own peripherals!

The first could be a low-cost module which provides an **interface with a standard cassette-recorder.** This means you can use ordinary tape-cassettes for the storage of data and programs.

To get the best from this configuration, you could uprate your system with a **revised monitor** – consisting of 2 replacement PROMs, pre-programmed with sub-routines for the interface, offset calculation and single step, and single-operation data entry.

The second peripheral could be your own **PROM programmer and blank PROMs** to set up your own pre-programmed dedicated applications. (Fusible-link device guarantees program safety.)

All are available now to owners of MK 14 – and remember Science of Cambridge keep you up to date *automatically* with advances in the MK 14 range. A TV interface device is already in the pipeline!

A valuable tool – and a training aid

As a computer, it handles operations of all types – from complex games to digital alarm clock functioning, from basic maths to a pulse delay chain. Programs are in the Manual, together with instructions for creating your own genuinely valuable programs.

And, of course, it's a superb education and training aid – providing an ideal introduction to computer technology.

SPECIFICATIONS

MK 14

∗ Hexadecimal keyboard
∗ 8-digit, 7-segment LED display
∗ 8 x 512 PROM, containing monitor program and interface instructions
∗ 256 bytes of RAM
∗ 4 MHz crystal
∗ 5 V regulator
∗ Single 8 V power supply
∗ Space available for extra 256-byte RAM and 16 port I/O
Edge connector access to all data lines and I/O ports

Optional Extras

∗ Extra RAM – 256 bytes
∗ RAM I/O device

Where to meet other micro users

BY FAR the most popular request to our Feedback columns this month has been how to contact other users of personal computers. In response, therefore, we are publishing a round-up of user groups and interested parties around the country. We shall update this list in forthcoming issues as other user groups notify us.

BEDFORDSHIRE

Texas Instruments 9900
Microprocessor User Group
Simon Garth
67 De Parys Avenue
Bedford

BERKSHIRE

Thames Valley Group
Bob Contis
Pippins
Boulters Lane
Maidenhead SL6 8JT
Berkshire
Tel: Maidenhead 22445

77168 User Group & W. B.I
User Group
Newbear Computing Store
7 Bone Lane
Newbury
Berks
Tel: 0635 49223

BUCKINGHAMSHIRE

I.N.M.C.
User Group for Nasco Lynx
Electronics
92 Broad Street
Chesham

CAMBRIDGESHIRE

Cambridge University
Processor Group
Emrys Williams
Cavendish Laboratory
Downing College
Cambridge

Cambridge University
Processor Group
Tim Hopkins
Magdalene College
Cambridge
CB3 0AG

CHESHIRE

Dr Raizada
Limrose Electronics
241-243 Manchester Road
Northwich
Tel: 0606 41696/7

Manchester User Group
Amateur Computer Club
P Wade
6 Mossgrove Road
Timperley
Cheshire
Tel: (Home) 061-890 2755
(Work) 061-236 9432 ext 211

DEVON

Local East Devon or South
West Region
Amateur Computer Club
D Carne
44 George Street,
Exmouth
Devon EX8 1LQ
Tel: 039-52 74479

South West Group
Amateur Computer Club
G V Barbier
Palmers Mill
Calverleigh
Tiverton

EIRE

Computer Education Society
of Ireland
7 St Kilmacud
Blackrock
Co Dublin
Ireland

ESSEX

Comemco
313 Kingston Road
Ilford
Essex

Amateur Computer Club
Mike Lord
7 Dordells
Basildon
Essex
Note: Membership throughout U.K. with many regional clubs and a newsletter.

GLOUCESTERSHIRE

Cheltenham Local Group
Amateur Computer Club
P Pullin
45 Merestones Drive
The Park
Cheltenham
Tel: Cheltenham 25617

Heath User Group
Gloucester
GL2 6EE

HAMPSHIRE

Southampton University
Amateur Computer Club
Paul Maddison
Students' Union
University Road
Southampton
SO9 5NH.

HERTFORDSHIRE

Scrumpi User Group
(Bywood)
68 Ebberns Road
Hemel Hempstead
Herts
HP3 9QRC
Tel: 0442 62757

KENT

Gillingham Group & Program
Library
A Aylward
194 Balmoral Road
Gillingham

U.K. Pet Users' Club
Commodore Systems
360 Euston Road
N.W1 3BL

LANCASHIRE

Northwest Group
Amateur Computer Club
Ken Horton
50 Lymfield Drive
Worsley
Tel: 061-228 6333 ext 372

LONDON

Z-80 Group
Roger Sinden
18 Percival Road
East Sheen
S.W14 7QE
Tel: 01-878 5374

MIDDLESEX

Harrow Group
Amateur Computer Club
Alan Secker
209 Albury Drive
Pinner
Middx HA5 3RH
Tel: 01-428 0844

MIDLANDS

Midlands User Group
Amateur Computer Club
Roy Diamond
27 Loweswater Road
Coventry CV3 2HJ
Tel: 0203 454061

OXFORD

Research Machines Ltd
PO Box 75
Oxford

SCOTLAND

Linlithgow Local Group
Amateur Computer Club
Stewart Stevenson
Lindisfarne
New Well Wynd
Linlithgow
West Lothian
Tel: 2657

Feedback for you

If you want to know how to put together a system, or find out what is available from where and at what price, simply drop a line to the Feedback column and we shall try and help. If you have any interesting development let others know, too. Write to:

Feedback, Practical Computing,
2 Duncan Terrace, London, N1

Scottish Amateur Computer Society. Meets first Wednesday of each month at The Glencairn Hotel, 21 Royal Circus, Edinburgh at 7.30pm. Details: Harry Sheldrake 031-332 6849.

Anyone interested in forming a Tandy user group should contact Mr Heller at 8 Morris Walk, Newport Pagnell, Bucks. MK16 8QD.

READERS within striking distance of Liverpool can contact a newly-formed Mini/Micro Computer Club.

It is already holding regular monthly meetings with demonstrations of different systems. Members are being encouraged to take as active a part as possible in the club, possibly by talking about and demonstrating their own systems.

Further information from: Merseyside Mini/Micro Computer Club, c/o STEM, School of Education, 19 Abercrombie Square, PO Box 147, Liverpool University, Liverpool, L69 3BX.

Two undergraduates at the University of Sussex are keen to set up a Sussex Personal Computing Society.

They are P. Guile and N. Latchem, who suggest that the club may be affiliated to the university's own computing society and could share facilities.

Anyone interested in forming the club, or even speaking to the society, should contact them at 23 Silverdale Road, Hove, Sussex.

Anyone running a user club or society wishing for reference to its activities or to be included in our list should write to The Editor, Practical Computing, 2 Duncan Terrace, London, N1.

Logical

I READ your article on Pets in the Kitchen in your first issue with interest as Beyts Logic has been writing software for SWTPC users for some time. Beyts Logic was commissioned by Chandru to write this software to his system's specification. The development was done on another SWTPC system, using discs, and then transferred to his, which is cassette-based. Richard Smith, one of our consultants, wrote the system and helped to install it at the restaurant. Chandru has since modified it by cutting-out some routines which were not required.

Tim Beyts
Managing Director
Beyts Logic

Users' Corner

I WRITE to enquire if you have any proposals for a Users' Corner in your magazine. I have developed an advanced program for costing and VAT (recoverable) for a building concern (but readily adaptable) which I would offer in exchange for another apple user's business program. It is 14K in length, crashproof and it works and took over 300 hours to complete. It is on tape and disc.

I have a 48K Apple, discs, voice control *et al* and would welcome meeting dedicated Apple programmers to discuss problems for mutual advantage.

J O Hodgson
30 Carre Street
Sleaford, Lincs.

We are very keen to promote user groups and on our Printout pages we list user groups now in existence. Feel free to write to us if you need publicity for any event.

Tandy packages

CONGRATULATIONS on producing a British computer hobby magazine—at least we can now buy something for 50p instead of the American magazines at £2.

I have recently bought a Tandy computer and I am a little disappointed that there are not many software packages available for business use for the Tandy. Would any of your readers who have developed software packages for the Tandy care to sell or swop some? We could even start a shop for Tandy packages.

Chris Jones
London, N16

The Tandy TRS 80 is reviewed in this issue and Tandy Coporation says that it has a number of software packages under development which should be available in about six months. *Practical Computing* is reviewing a stock recording system produced by a Tandy user which will be published in a future issue. In the meantime, contact Mr Heller of 8 Morris Walk, Newport Pagnell, Bucks, who is interested in forming a Tandy user group.

Exhibitions

CONGRATULATIONS on producing a fine new magazine for hobbyists. Could you tell me if there are any exhibitions or demonstrations of computer equipment where I can see some applications in action?

Martin Jones
Gerrards Cross
Bucks

The next big computer show for hobbyists and small businesses is not until next year. Online, which organised this year's DIY show, is holding another from July 5-7 at the Bloomsbury Centre Hotel. Otherwise you could try Compec, a computer and peripherals exhibition to be held from December 5-7 at Olympia, or Micro Systems from January 31 to February 2 at the West Centre Hotel.

Mailing list

I AM running an estate agents and I should like to set up a mailing list which would keep track of customers looking for particular size and price of house. I have heard that if may be possible to run this on a computer for about £1,000. Can you help?

G. Myles
London SW6

Indeed it is. Many computer firms have mailing programs to run on their machines and we are sending you a selection of names.

Conversion

I HAVE heard that there is a way of converting an electronic typewriter into an input/output device and I am keen to try and build one. Unfortunately, I cannot find an article which details just how to do this. Can you help?

Peter Richards
Leeds 11

We have tried to source an article but without success so far. If there is a reader who has converted an electronic typewriter successfully please drop us a line with details. And we will pay for the article.

Robot

I WOULD like to build a robot but I am afraid of re-inventing the wheel. Is there any association of group of people with a similar interest so that I can correspond? Wishing you success for a great magazine.

Andrew Hunter
Cardiff

New from Texas Instruments. The world's most powerful pocket calculators. For the easiest problem solving ever.

The new Texas Instruments Programmable TI-58 and TI-59 make your problem solving simpler and easier by making the electronics do more work. Now, commonly encountered programmes in maths, science, finance and statistics are set up and accessible at the touch of a key. You need add only the variables.

Solid State Software† is the name of this technological achievement from Texas Instruments. Even the programming is now included in the solid-state electronics. You get complete, pre-written problem solving libraries in convenient plug-in modules. Yet, no prior programming knowledge is necessary.

The programmable TI-58 at £79.95 *

Includes a Master Library *Solid State Software* module packed with 25 useful programmes, all at your command. Or you can key-in your own programmes and store the data — using up to 480 programme steps or up to 60 memories — and employ the Master Library programmes as subroutines.

Optional plug-in library modules are available to convert your calculator into a specialised problem solver in the fields of applied statistics, surveying, aviation, navigation — with many more 5,000-step libraries to come.

The programmable TI-59 at £199.95 *

Includes all the features of the TI-58 — plus more programme steps, more memories, and a magnetic-card capability. Record your own programmes on convenient magnetic cards and store them permanently in your personal problem-solving library.

The Texas Instruments Programmable TI-58 and TI-59 and the compatible PC-100B alphanumeric printer/plotter include a 1-year warranty. See the world's most powerful pocket calculators now.
Or use the coupon to obtain full product information.

Please send me full details on the TI-58, TI-59 and PC-100B.

Name ______________________

Address ______________________

____

____

P.C. 11.78

Simulated Calculator Display.

Giant technology. From the people who made micro-electronic calculators and watches possible.

TEXAS INSTRUMENTS
LIMITED

†Trademark of Texas Instruments
*Suggested retail price, including VAT.

Texas Instruments Ltd, Supply Division, Manton Lane, Bedford. Tel: Bedford (0234) 67466.
Branch offices at: Slough - 0753 33411. Edinburgh - 031 229 5573. Stockport - 061 442 7000. Southampton - 0703 27267

● Circle No. 109

Designing with Apple

As you watch that souffle sink, have you ever wondered if it would not have been better to install the cooker out of the draught from the back door which doesn't close because of some clumsy doing-it-yourself?

Designing a fitted kitchen is an art. People change their minds, manufacturers change their products; the design process necessitates repetitive drawing of plans. This alone can take up to 10 hours' work per kitchen on the part of the designer. With an Apple II microcomputer and some neat software, however, this tedious and time-consuming procedure is eliminated.

The firm responsible is Templeman Software Services of Stratford-upon-Avon. It was formed originally to write applications for Ventek Datapoint systems and, more particularly, for Computer Automation Syfa equipment; it has been in business six years now. A list of clients includes such impressive household names as Corona, Beechams and BOC. As hardware started to become less important in applications design, TSS began looking around for fresh projects and came across the idea of kitchen design as ripe for adaptation to computer techniques.

Movable feast

The computer in question, an Apple II, is installed on the designers' premises—usually in a prominent position (at the request of the user). Some plan to take it to the customer's home and plug it into the TV.

The designer and client work together in front of the VDU. The dimensions of the kitchen in question are keyed-in and a plan appears on the screen. Doors, windows and permanent features such as a boiler are represented in standard technical drawing notation.

The next step is to choose the units. There are some 10 principal ranges of kitchen furniture and the application program uses a specially-designed coding structure to indicate each unit. Details of a unit are keyed-in and appear on the plan, the dimensions and features shown by a variety of shadings, After each addition the computer asks if you like that particular unit in that location, giving the operator the chance to re-locate or delete it. A deleted unit does not vanish from the screen, but it will not figure in subsequent versions of the plan. The plan can also be seen in a side elevation.

Enthusiastic

At any point in the design process the computer will display (or print) a list of units so far selected. The limited storage facilities and access of tape mean that only average prices can be quoted on a cassette system. If the user opts for floppy discs, however, a far larger catalogue can be maintained and the exact price for a particular item can be immediately available.

When a specific design has been finalised, it would be printed out on a Qume daisy-wheel printer (although the excellent colour graphics of Apple cannot, of course, be reproduced). The order forms and even the contract can be prepared and printed at the same time.

The first kitchen to be designed with the aid of the system was installed at a private house in Stratford on September 11.

Commercial response has been "enthusiastic" and Myles Pollock, who is responsible for the venture, admits surprise at the level of interest aroused. Around 20 systems are on order.

TSS will supply the following package for a basic £2,500.

- All software and subsequent updates, although the usual small charge is made to cover the cost of an update tape and postage;
- Apple II with 48K of memory;
- graphics software;
- games tapes and manuals.

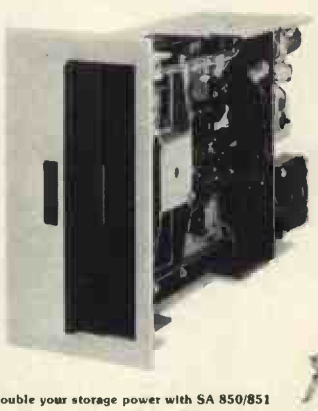
The purchaser must supply a cheap cassette player and any TV set. The printer is an optional extra, as are the mini-floppies: to deal with more conventional functions TSS can also supply an accountancy package.

Advantages to the kitchen buyer are obvious. The system reduces drastically the time taken to design a kitchen; it will also handle the costing and produce the necessary documentation.

Moreover, customers respond to the novelty and immediacy of the experience and the computer is proving to be an excellent sales aid.

The basic program is fairly complex—"it has to be universal but fool-proof." Few kitchen designers have much experience of operating a computer. Nevertheless, users rapidly have become adept at running and even modifying the system.

Templeman Software Services is very pleased with the Apple II and intends to devote some time to developing it further. It is particularly keen to find other applications.



CP/M now available

LP ENTERPRISES is well-known already as a supplier of journals and books on micros, but it is now branching-out into software for microcomputers. The first item offered is the CP/M Operating System for Intel 8080 and Z-80 based micros with 20KB RAMs.

That is a sensible starting place. CP/M was written by Digital Research, in the States, and has been around on 8080s for some three years.

It is the operating system which Rair chose for its Black Box micro-business system, with modifications, largely on the basis that it was a well-proven piece of software.

The CP/M which LP Enterprsies is stocking has passed through the hands of another American software house, the intriguingly-named Lifeboat Associates, being modified in the process to run with North Star disc systems. It costs £99 plus VAT, over the counter.

MAKING THE KIM MORE POWERFUL

WE SEEM to be hearing all the time about the Pet system from Commodore Systems, but Pet's 'kid sister', the Kim 1 microprocessor, is also beginning to make an impact. One company which distributes it, GR Electronics of Newport in Gwent, has announced a range of new features which turn the basic system, which retails at £161, into a sophisticated and powerful machine.

GR is selling a video board at £150 which allows the Kim 1 to be plugged straight into a domestic television set, for use as a visual display unit with a capacity of 16 lines of 64 characters.

A Memory Plus board, costing £199, adds a further 8K bytes of RAM and allows for another 8K of EPROM (that's erasable programmable read-only memory) to be attached.

In addition, GR has launched a Pocket Terminal to act as a sophisticated keyed input device to Kim. The terminal is a hand-held device with 40 dual-purpose keys, giving a full ASCII character set, and costs £240.

On the software front, GR is supplying a wide variety of ready-written software for Kim1. This ranges from basic systems software such as Assembler/Disassembler/Editor, to a number of games programs.

Financial calculator

TEXAS Instruments has introduced another calculator, the TI-31, designed for financial and statistical functions.

It costs £16·95, and copes with such calculations as percentages, simple and compound interest, cash flow, cost control and depreciation.

Included in the TI-31 keyboard are special keys which set in action pre-programmed financial functions, allowing the user to carry out 'what if' exercises simply and quickly.

For statistics, it incorporates facilities like linear regression and trend-line analysis, and has mathematical functions such as logarithms, roots and reciprocals.

It runs normally on replaceable batteries, but an optional accessory kit can be bought to convert it for rechargeable batteries.

Further information: Texas Instruments Ltd, European Consumer Division, Manton Lane, Bedford MK41 7PA.

How to get started

THOSE WHO want a good in-depth knowledge of microprocessors should take note of the regular courses being run at Bradford University.

The three-day course is entitled Getting started in microprocessors, and has been so successful that the University's Postgraduate School of Electrical and Electronic Engineering is now running one every month.

It is based mainly on the Motorola 6800, consisting of introductory lectures and explaining the basic principles of microprocessors. Students also learn how to program and get some hands-on experience with a variety of typical engineering problems.

An optional fourth day is available on the course for those who wish for a deeper grasp of programming. Further details and application forms can be obtained from the Digital Systems Laboratory, Room WB22, School of Electrical and Electronic Engineering, University of Bradford, West Yorkshire BD7 1DP. Tel: Bradford 33466 Ext. 568.

Catalogues

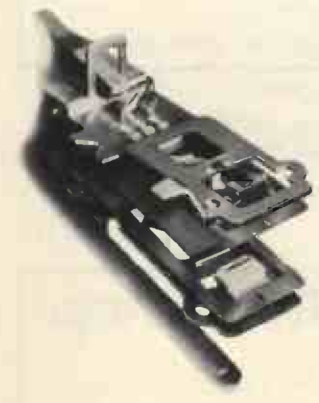
COMART, specialists in S100 microcomputer systems, has its Autumn catalogue available. Products included for the first time are the Cromemco System Three single card computer and the SOL 20/16 terminal computer system.

Catalogues available from Comart Ltd, PO Box 2, St. Neots, Cambridgeshire, PE19 4NY.

U.K. assembly begins

COMMODORE is starting U.K. assembly of the £695 Pet Microcomputer at its Eaglescliffe plant in the North-east of England. Initial plans are for a monthly rate of 500 units, but this will rise to 1,000 units by December and is in addition to current U.S. production.

The Commodore announcement of U.K. production follows a successful launch of the Pet Computer. More than 1,500 units have already been sold in the U.K. and there is a nationwide network of 50 dealers. As well as personal users, the Pet Computer is being sold in universities, schools, research establishments and businesses generally. The Pet Computer is based on the MOS Technology 6500 microprocessor.



Apple agent

TOPMARK Computers has been appointed sales agent for East Anglia and North Home Counties for the Apple II personal computer.

Apple II, reviewed in our July/August issue, features lightweight and colour graphics and is the ultimate personal computer. Applications include process control, data logging, design, statistical analysis, accounting management statistics, automatic testing, laboratory research, teaching, audio-visual display, numerical control and microprocessor development, as well as games and problems.

Topmark Computer is at 77 Wilkinson Close, Eaton Socon, Huntingdon, Cambs, PE19 3HJ. Tel: Huntingdon (0480) 212563.

North Star

COMART, the U.K. distributor for the Micro-disc system from North Star Computers, now has Release 4 of the system software in stock.

This has enhanced versions of both DOS and Basic, and for the first time includes a memory monitor.

It incorporates automatic line numbering for Basic, as well as new error-trapping routines.

Games for all kinds of micros

GOOD NEWS for games enthusiasts.

A company called Mini Micro is compiling catalogues of well-tested and debugged games for micro systems.

Formed in September, the firm is going into business initially through mail order. Games programs are being generated mainly by Mini Micro itself, but any contributions from outside, we understand, are more than welcome.

The first catalogue is being devoted entirely to programs which run on the Pet system. But soon to follow are games written in Z80-machine language, and also for the Research Machine 380Z. Kim 1 will also be covered by mid-1979.

For information about the catalogues, or how to contribute material: Mini Micro, 47 Queens Rd., London N11 2QP, Tel: 889-7615.

New floppy systems

RESEARCH Machines of Oxford has introduced new floppy disc systems for its 380Z computer.

A mini-floppy system is offered with one or two drives, each drive of 70K byte capacity. Also available is a dual, double-sided, standard drive system with a capacity of 500K bytes per drive giving a total of one megabyte of on-line storage; this system can produce IBM3740 compatible discs.

At a later date a new disc controller card will be offered which will allow existing standard drive users to upgrade to IBM standard double density recording, doubling the capacity to one megabyte per drive.

BASF drives are used throughout and the prices of both the mini and standard disc systems include the Digital Research CP/M disc operating system which is rapidly becoming the industry standard for microcomputers.

The price of the 380Z with 32K RAM is £1,158; the dual mini-floppy disc system, MDS-2, is £895; and the dual standard system, FDS-2, is £1,695.

Latest from Intel

WE DON'T know who thinks of the names for Intel computer systems, but we reckon he may need a refresher course.

The latest single-board computer to emerge from that eminent microprocessing company is called the—wait for it—iSBC 80/30.

Still, apart from the name, the system appears to have some interesting and sophisticated features. But it is likely to interest only those with complex computing needs, as its Multibus architecture is designed specifically to allow it to share memory and input/output devices with as many as 15 other processors.

This means that the 16K bytes of read-write memory on the iSBC 80/30 board can belong to the system as whole, to the local processor, or be assigned to both. For extra fast performance, it uses a slave microprocessor for input/output processing.

Software includes a new version of the RMX-80 operating system, PL/M-80, Fortran and the 8085 microassembler. For further information: Intel Corporation (U.K.) Ltd, 4 Between Towns Rd., Cowley, Oxford. Tel: 0865-771431.

And now the great graffiti

GRAFFITI. What does the word mean to you? Hurried scribblings on a lavatory wall? Crude, sometimes illustrated messages poured out to an unsympathetic world? Or witty, memorable aphorisms to brighten an otherwise dull life?

However you view graffiti, there is no escaping the fact that ever since man carved his first mark on a cave wall, it has been a lively and spontaneous mode of expression. For example, archeologists at Pompeii would never have learned half as much as they did had not the Pompeiians been such avid graffitists.

So *Practical Computing* would like to maintain the great tradition by having our readers put on their thinking caps and producing graffiti to express the new trend in home computing.

As you may have noticed, outpourings in recent years have tended to concentrate on sex, football or politics, while surprisingly—or perhaps not surprisingly—the big, wide world of electronics and computers has been ignored entirely.

So *Practical Computing* aims to remedy the omission by inviting readers to submit ideas for electronics graffiti; and we'll give a calculator to the one who sends the best contribution.

The graffiti format gives you plenty of freedom. A sharp, witty remark, a poem—possibly a limerick—or even a slogan could be attempted. We shall be printing a selection of the best efforts in the next issue.

So reach for your felt-tip pens or paint-spray cans, if that's what you use normally, and start graffiiting.

Entries should reach us by December 1 and be addressed to Graffiti Competition, *Practical Computing*, 2 Duncan Terrace, London N1.

To get the ball rolling, we offer a few examples of the kind of thing we're looking for.

If you think they are terrible, why not show us you can do better, and perhaps win a calculator at the same time?

"*I thought a cross assembler was an angry carpenter until I discovered software*".

* * *

"*When in ROM . . . do as you damn well like; it won't make any difference.*'

* * *

"*A bit on the side—computing's answer to lateral thinking.*"

* * *

"*A roll-in program gathers no MOS.*'

* * *

"*I thought a floppy disc was a record which never made the Top Ten until I discovered micros.*'

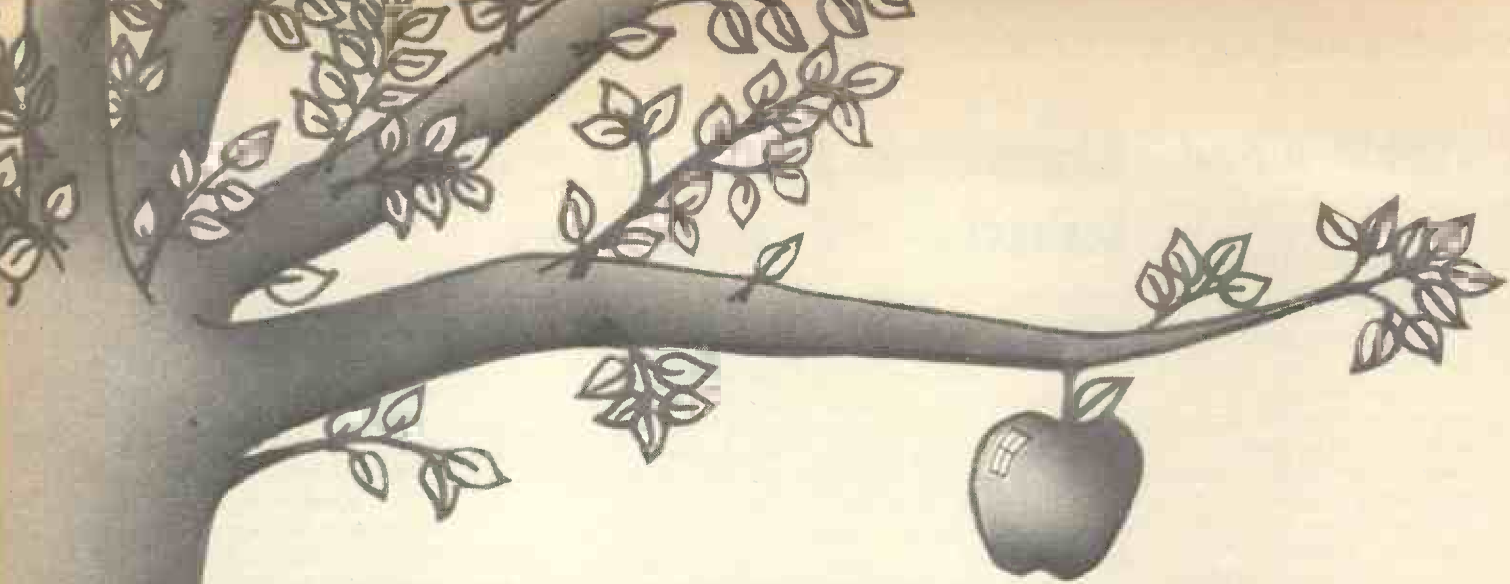
* * *

"*Soldering in the dark—a night at the PROMs.*"

* * *

"*There was a young laddie called White,*
Who invented a 14-bit byte,
You will get so much more,
For your money, I'm sure,
But I doubt if you'll get your sums right."

Now it's your turn.



Keen Computers Branches Out!

Keen Computers are now agents for **South West Technical Products (UK) Ltd** and also stock the **BUS-2**

EVERYTHING YOU NEED FOR YOUR COMPUTER!

A comprehensive range of computer stationery and accessories is also readily available:-

Floppy disk files	£9.98	Suspension binders	from £1.28
Floppy disks	£3.00	C15 cassettes	£0.85

Keen Computers is a main dealer for the **apple II** computer.
Simply a better and more advanced machine for many business purposes
* It uses a 6502 microprocessor – a very updated version of Motorola 6800
* It is a completely built system with colour graphics and comprehensive basic

Apple II Computer 16K	£1050
Additional memory per 16K	£250
Disk with controller	£425
Disk without controller	£375
Applesoft card	£100

Printers

Data 100 fast matrix printer	£1750
DDT DECwriter II	£1050
Centronics 779	£699
Centronics Microprinter	£350

SOFTWARE

As well as our normal consultancy service we also market the following software packages:-

Addressing and mailing program	£50	Incomplete record accounting	£250
Word processor upper and lowercase	£50	Co resident assembler	£20
Personal accounting	£10	Sales growth analysis	£10
Depreciation calcs	£10	A variety of statistical packages	from £25

A large range of games are available
(Prices subject to change without notice)
For a full catalogue please contact:

Dr. T. KEEN, KEEN COMPUTERS LTD., 58 CASTLE BOULEVARD
NOTTINGHAM TEL: 45865/49588

● Circle No. 114

Pet dealers

A further selection of Pet dealers to add to our list published with our Pet review last month.

LONDON
TLC World Trading Ltd,
32 Craven Street,
Charing Cross,
London WC2
Tel. No.: 01-839 3894

Sumlock Bondain Ltd,
Shop Unit 2,
London EC4
Tel. No.: 01-626 0487

Euro-Calc Ltd,
224 Tottenham Court Road,
London W1
Tel. No.: 01-636 8161

HUMBERSIDE
Microprocessor Services Ltd,
1 Nunburnholme Avenue,
North Ferriby,
North Humberside
Tel. No.: (0482) 631015

NORFOLK
Sumlock Bondain
Grosvenor House,
Prince of Wales Road,
Norwich
Tel. No.: (0603) 26259

HOME COUNTIES
R.I.I.C. Business Systems Ltd,
31 Church Road,
Tottenhoe,
Dunstable
Tel. No.: (0582) 608289

Data Efficiency Ltd,
Maxted Road,
Maylands Avenue,
Hemel Hempstead,
Herts HP2 7LE
Tel. No.: (0442) 63561/41191

Syntec Systems Ltd,
Brook House,
173 Millbrook Road,
Southampton SO1 0HR
Tel. No.: (0703) 37731

MIDLANDS
Memec Systems Ltd,
Thame Park Industrial Estate,
Thame,
Oxon OX9 3RS
Tel. No.: (084 421) 3149

H.B. Computers,
22 Newland Street,
Kettering,
Northamptonshire
Tel. No.: (0536) 83922

Davidson-Richards Ltd,
14 Duffield Road,
Derby
Tel. No.: (0332) 366803

Allen Computers,
16 Hainton Avenue,
Grimsby,
South Humberside
Tel. No.: (0472) 40568

NORTH EAST
North Yorkshire Computer Services,
York House, 26 Yarm Road,
Stockton-on-Tees,
Cleveland TS18 3NG
Tel. No.: (0642) 64673/62951

NORTH WEST
D. Kipping Ltd,
Ralli Building,
Stanley Street,
Salford, Manchester
Tel. No.: 061-834 6367/9

SOUTH WALES
Sigma Systems Ltd,
54 Park Place,
Cardiff, CF1 3LP
Tel. No.: 0222 21515

If we missed out your dealership, please let us know.

Newbear Panda

UNVEILED for the first time in September was the Newbear Computer Store answer to Pet and other systems like the Tandy TSR-80, its new contender the Panda.

Based on a Motorola 6800, Panda runs the standard operating system CP/M, and costs around £640. Complete with keyboard and visual display, it also has a cassette tape interface. It has an 8K Basic interpreter, and a disc operating system based on CP/M is promised by next summer. Also from Newbear is a new shop in Manchester. The new store is at 2 Gately Rd., Cheadle, Cheshire. Tel: 061-491 0134.

ALL I WANT FOR CHRISTMAS IS .....

Easier than getting an old man and a reindeer to drop one down the chimney is to enter the **second great Practical Computing** competition.

Practical Computing is giving away some ace equipment — a DEC 16-bit micro — the LSI-11. What's more, the finalists will have the chance to participate in Compec Europe in Brussels next February.

You will be asked to design a total computer system for the home, based on the LSI-11. In the preliminary stages, technical know-how will be kept to a minimum, so you can **all** enter.

Full details in next month's issue, so Come all ye faithful, and reserve **your** copy of the December issue of **Practical Computing**, out on November 15.

..... A COMPUTER!

Dept PC3
313 Kingston Road, Ilford,
Essex, IG1 1PJ, England
01-553 1001

LP ENTERPRISES

From the representatives in Europe . . . for America's leading Micro-computer magazines and books, for the hobbyist, educationist and professional alike, we bring you a little light browsing!

Reading maketh a full man . . . . Francis Bacon (1561–1626)

Tick or indicate quantity ordered

From Adam Osborne Associates	
INTRODUCTION TO MICROCOMPUTERS	
Volume 0: The Beginners Book	£5.95
Volume 1: Basic Concepts	£5.95
Volume 2: Some Real Products (Revised Late 1977)	£11.95
6800 Programming for Logic Design	£5.95
8080 Programming for Logic Design	£5.95
Z80 Programming for Logic Design	£5.95
8080A/8085 Assembly Language Programming	£6.95
6800 Assembler Language Programming	£6.95
Some Common BASIC Programs	£5.95
BUSINESS PROGRAMS IN BASIC	
Payroll with Costing Accounting	£9.95
Accounts Payable & Accounts Receivable (Available from Midsummer 78)	£9.95
General Ledger (Available from late summer 1978)	£9.95
From Scelbi Computer Consulting Inc.	
6800 Software Gourmet Guide & Cookbook	£7.95
8080 Software Gourmet Guide & Cookbook	£7.95
8080 Programmers Pocket Guide	£2.25
8080 Hex Code Card	£2.25
8080 Octal Code Card	£2.25
8080 Guide and One 8080 Code Card	£4.20
8080 Guide and Both Code Cards	£6.00
8080 Standard Monitor	£9.95
Understanding Microcomputers & Small Computer Systems	£7.95
SCELBI 'BYTE' Primer	£9.95
8080 Standard Assembler (In Book Format)	£15.95
8080 Standard Editor (In Book Format)	£9.95
From Peoples Computer Company	
Reference Books of Personal & Home Computing	£4.95
What to Do After You Hit Return	£7.00
Dr. Dobbs Journal Volume 1	£10.00
From Kilobaud/73 Magazine Inc.	
Hobby Computers Are Here	£3.95
New Hobby Computers	£3.95
From Dymax Inc.	
Instant BASIC by Jerald R.Brown	£4.95
Your Home Computer by James White	£4.95
My Computer Like Me . . . When I Speak BASIC by Bob Albrecht	£1.65
Games with a Pocket Calculator by Thiagarajan & Stilovitch	£1.75
Games, Tricks and Puzzles for a Hand Calculator by W.Judd	£2.49
From BYTE Publications Inc.	
Paperbytes:	
Tiny Assembler for 6800 Systems	£5.75
Bar Code Loader for 6800, 8080, Z80 & 6502 Micros	£1.75
Best of BYTE Volume 1	£8.95

Tick or indicate quantity ordered.

	Price UK	Price Overseas If Different
From Creative Computing Press		
Best of Creative Computing Volume 1	£6.95	
Best of Creative Computing Volume 2	£6.95	
101 BASIC Games (Revised & Reprinted Feb.78)	£5.50	
The Colossal Computer Cartoon Book	£3.95	
Computer-Rage (A new Board Game)	£6.95	
Artist and Computer	£3.95	
From Everyone Else		
Magazine Storage boxes (holds 12 minimum)	£1.25	
Sybex: Microprocessors from Chips to Systems by R.Zacs	£7.95	
Sybex: Microprocessors Interfacing Techniques by R.Zacs	£7.95	
Dilithium: Home Computers Volume 1: Hardware	£6.50	
Dilithium: Home Computers Volume 2: Software	£5.95	
Getting involved with your Own Computer	£4.75	
TV Typewriter Cookbook by Don Lancaster	£7.50	
TTL Cookbook	£7.50	
CMOS Cookbook	£7.95	
IC Timer Cookbook	£7.50	
IC OP–AMP Cookbook	£9.50	
RTL Cookbook	£4.25	
Z80 Microcomputer Handbook	£7.50	
Computer Programs that work (In BASIC)	£2.55.	
From Basic Software Library (from Scientific Research Instruments)		
Vol 1: Business and Personal Booking Programs	£17.50	
Vol 2: Maths and Engineering Programs	£17.50	
Vol 3: Advanced Business Programs	£26.95	
Vol 4: General Purpose Programs	£7.95	
Vol 5: Experimenters Programs (General Purpose)	£7.95	
Vol 6: General Ledger Program	£32.50	
Vol 7: Professional Programs	£26.95	
Magazines: Back Issues		
Personal Computing	£1.75	
Interface Age	£2.25	
Dr. Dobbs Journal	£1.75	
Computer Music Journal	£2.50	
Peoples Computers	£1.75	
*BYTE	£2.25	
Creative Computing	£1.75	
Calculators & Computers	£1.75	
ROM	£1.75	
Kilobaud	£2.25	
73	£2.25	
MAGAZINES: Subscriptions		
Personal Computing (Twelve Issues Yearly)	£16.00	£17.00
Interface Age (Twelve Issues Yearly)	£20.00	£20.50
Dr. Dobbs Journal (Ten Issues Yearly)	£13.00	£13.50
Computer Music Journal (Four Issues Yearly)	£8.50	£9.00
Peoples Computers (Six Issues Yearly)	£8.00	£8.50
Kilobaud (Twelve Issues Yearly)	£20.00	£21.00
*BYTE (Twelve Issues Yearly) via USA	£15.00	
BYTE (12 issues yearly) via U.K.	£21.00	
Creative Computing (Six Issues Yearly)	£8.50	£9.00
Calculators & Computers (Seven Issues Yearly)	£10.00	£10.50
73 (Twelve Issues Yearly)	£20.00	£21.00

If you wish to leave your magazine intact, photocopy this page. Letters will be accepted.

PC3

HOW TO ORDER

Please note our prices include postage and packing, but not insurance, if wanted add 12p for every £10. of books ordered. Make cheques, PO's etc. payable to:–

L.P. Enterprises.

CREDIT CARDS accepted
BARCLAYCARD VISA / ACCESS
DINERS CLUB / AMERICAN EXPRESS

Phone: 01-553 1001 for Credit Card orders (24–hr service)

Send to address above
Indicate Payment Method:

All Orders must be Prepaid

Total Enclosed £ . . . .

. . . . . . My cheque, P.O., I.M.O. is enclosed in Sterling on U.K. Bank

. . . . . . Charge to Barclaycard/Visa/Access/Diners/American Express

Credit Card No . . . . Expiry Date . . . .

Name . . . .

Address . . . .

. . . . POSTCODE . . . .

Signature . . . .

All publications are published in U.S.A. and shipped air-freight by L.P. Enterprises. In unusual cases, processing may exceed 30 days.
*BYTE subscriptions are processed in U.S.A. and are air-freighted & posted from Amsterdam and will take 3 months to start.
U.K. Subscriptions start within 3 weeks.

TRADE ENQUIRIES WELCOME

TANDY REVIEW

IT WAS with some apprehension that *Practical Computing* took delivery of the Tandy computer. Three boxes containing a display, a keyboard, a cassette recorder and power supply, plus a huge box of manuals, were delivered not long after the simple, single unit Pet computer had been returned to Commodore.

How were we to review this system? As a computer to learn about computing? As a hobby machine? Or as a potential business system.

When we visited our local Tandy shop and asked what the system could be used for, they said all three. So be it.

The Tandy is designed and manufactured in the U.S., where some 68,000 systems have been sold. It is now being sold in all 180 Tandy shops in the U.K. and as such is probably the system which has the widest number of retail outlets in Britain. Since April, Tandy has sold more than 1,000 systems here.

Child's play

There are four versions, each complete with a 12 in. video monitor, a battery/AC cassette recorder, a power supply, a 232-page manual and a two-game cassette.

There are two versions of the system, Level I and Level II. The cheapest is a Level I system with 4K of RAM at £499. Level I with 16K RAM costs £728; Level II with 4K costs £578; and Level II with 16K costs £807.

Our review system was the Level II with 16K RAM. We realised quickly that we should have started with the Level I system.

Connecting the system is child's play, although you are left with a tangle of wires. We found it handy though to be able to move around the keyboard when we wanted to make a few notes.

The keyboard is the familiar QWERTY typewriter layout and it is certainly much easier to use than the Pet.

At the back of the keyboard there is an on-off switch and, hidden behind a smaller flap, a re-set button. Both are somewhat difficult to get to, though you clearly don't need to use them very often.

Switching-on should result in Memory Size being displayed on the screen; occasionally instead you get a random screenful of characters, which means that you have to keep switching on and off until you get the Memory Size display.

The screen has 16 lines of 64 characters and upper-case letters (why no lower-case?), numerics, and a limited number of special characters can be displayed. The screen can also be addressed as a 128 × 48 grid for graphics, each point being displayed as a 2 × 8 dot block.

Frankly, the quality of the display is not too good. It is about the same as can be achieved by an unmodified television set. For the price I would not have expected it to wobble continuously as ours did. Tandy says this is due to a fault on the opto isolator which has now been rectified.

Using the cassette to load and save programs can be very frustrating. The volume is particularly critical and has to be set to the right level, which can vary from tape to tape. In fact, we were unable to load some of the Tandy tapes.

When the tape is being loaded the system is rather uncommunicative. A single asterix appears on the screen to denote that some data has been found, and a second asterisk flashes when data is being read.

Tandy, as mentioned, supplies two versions of the system both of which support different versions of Basic. The reason is apparently to sell the new user Level I to learn about the system, and then to persuade him to upgrade to Level II. The trouble is that Level II programs cannot always be run on Level I; a conversion cassette is provided which will allow you to convert some programs. And Level I programs cannot always be run on Level II. Tandy says the reason is that three statements have to be altered. These are the DIM statement, the Print At statement and that commas need to be changed to semi-colons if they were incorrectly used. An amendment to the manual will be distributed. The Level II manual assumes a level of computing competence and takes you straight into Tandy's version of Basic.

Manual is fun

We would definitely recommend that the novice starts with Level I. This is a rather simple Basic; only two string variables (A$ and B$), and one array (A) are allowed.

Working through the Level I manual can be fun, though,. It's very easy to read and must be a joy for someone without computer knowledge.

"Let's write a simple one-line program to let the TRS-80 introduce itself", chortles the manual. "First be sure the last line on the screen shows a >, which we call a prompt. This is the computer's way

(continued on next page)

Tandy at play: James, aged 11, keys into the Tandy while Tom, aged 13, watches.

"Everyone seems to remember with great clarity what he was doing on November 22, 1963, at the precise moment he heard that President Kennedy was dead."—The Odessa File by Frederick Forsythe.

When I got the TANDY-80 going I was reminded of what I was doing when I heard that President Kennedy was dead.

At 7.30 p.m. on 22 November 1963, my first love and I were alone in an air-conditioned building in down-town Croydon. We were spending every evening together that week, late into the early hours of the morning and I was totally fascinated by her multitude of attractions. Being interrupted in this idyll by the shocking news of the first major political assassination of modern times, emphasised to me how totally entranced and absorbed I had been.

The subject of this entrancement and absorption was of course not a mere human being, but a computer. Following two days of tuition in Fortran programming at IBM in Newman Street, the mysteries of computing had been unlocked for me and I was absolutely head over heels about the wonders I could perform on this magnificent miracle.

The miracle in question was a 12K IBM 1401 with card read/punch, printer and four tape drives. Anything was possible with this huge number cruncher, and I took particular delight in incorporating the sense switches so that when someone visited me at the computer I could punch their name on to a card, bang it into the card reader and have the computer come up with some inane message such as: THE COMPUTER WELCOMES MR ALFRED BLOGGINS INTO ITS PRESENCE.

All this hardware was of course housed in a large air-conditioned room and the joy of it was that I could play with it by myself as this was in the days before intensive shift working; when a computer led a decent 8-hour day and could rest at night apart from the occasional attentions of lunatic enthusiasts like myself.

With the arrival of fancy operating systems, bigger, better and more expensive main frames, front ending, time sharing and all the other 'progress' heaped on us by main-frame manufacturers milking their boom years for all they were worth, the opportunity for good old fashioned, hands-on practical computing disappeared. With it went the fascination.

The TANDY was the first microcomputer I had taken home. I had it for the weekend and on the Saturday morning with the help of my 13 and 11 year old sons we set it up.

In order to breed a little familiarity we had decided to start with the backgammon program that came with the computer. This got us used to loading programs and pressing the buttons of the QWERTY keyboard. It was very impressive how quickly the two boys were completely at home with the equipment. One of the things that most fascinated was the fact that every time it came to one's turn the computer flashed up one's name with some sort of inane message such as: ALFRED BLOGGINS IT IS YOUR TURN.

Tandy at work: Tandy TRS 80 with its 1963 equivalent, the IBM 1401 in the background.

Having played each other for hours we then branched into backgammon matches between imaginary characters such as Niki Lauda versus James Hunt and Paddington versus Muffin (our cat and dog respectively). The old fascination was suddenly returning . . . . . We went on to blackjack and "Quick, Watson!" (a sort of animated Mastermind).

By then the 13 year old had started studying the users manual. This manual has a very attractive can't-put-it-down quality. It doesn't talk down to you and it has a very beguiling sense of humour. It really teaches in a very gentle step by step manner and leads one into the obvious mistakes so that one can learn the only way that anyone ever learns anything well, from one's own mistakes.

It really is an absorbing piece of kit to play with. The great strengths of the TANDY are its superb instruction manual and the convenience of the keyboard. The major drawback is the extraordinary sensitivity of the volume control and it is a great compliment to the fascination of playing with the TANDY that my sons were quite prepared to reload the same program some 20 times over in order to master this problem.

Having said all that, I think that to be able to buy all the power (and fascination!) of that IBM 1401 of 15 years ago for £500, and not to need an air-conditioned hangar to store it, makes a 1978 microcomputer one of the best buys of the century.

I am hooked.

By Wim Hoeksma.

(continued from previous page)

of saying 'Go ahead—do something'. Now type in the following line exactly as shown:

```
10 PRINT "HELLO THERE. I AM YOUR NEW TRS-80
MICROCOMPUTER!"
```

"If everything's OK you can press enter. The prompt will reappear. The computer is telling you: 'Fine—what's next?. Now type RUN and if you have made no mistakes the display will read:

```
HELLO THERE. I AM YOUR NEW TRS-80
MICROCOMPUTER!
```

Chapter by chapter, the manual teaches you how to program in Basic. With the manual there is a set of tapes which you can load on the system which will also teach you Basic interactively. Unfortunately, these cannot be run on Level II. so we could not review them. See what we mean about compatibility, Tandy?

The Level I sets you exercises at the end of each of most chapters to make sure you understand what you have been told. There are also simple programs you can write which demonstrate the facilities and the graphics capabilities of the system.

Level I Basic has all of the commands you would expect to have in a good Basic, including some useful shorthand statements. PRINT, for example, can be cut down to P.; NEW = N.; LIST = L.; GOTO = G. This is really great for cutting down the size of programs. But it is not available in Level II, which is a shame.

For someone who knows nothing about computing, though, and is keen to learn, we suspect that within a few days you could learn how to program the TRS-80 in Basic—it's that simple and the Level I manual is that good.

The Level II manual, on the other hand, assumes you know about computing and takes you straight into its features.

Software packages

At Level II you have a good extended Basic. String, integer, single and double precision variables are allowed with multiple dimension arrays.

One annoying feature of the language is that error messages are not explicit; a missing right track is reported as a syntax error and there is no syntax checking as statements are entered.

On Level II, cassette files may have single-character names and the system will search for a named file Simple, unnamed data files can be written to and read from tape.

One of the best features of Level II is the editor. It allows you to insert, delete or change characters in a line without having to re-type the entire line. In addition, you can search for the *nth* occurence of a particular character, delete all the characters between the current position and a particular character, or list the entire line and return to the start.

The only niggle is that the change com-

(continued on next page)

(*continued from previous page*)

mand does not allow you to change say three to five characters or vice versa; it can be used only to alter the exact number of characters.

At the moment there is not much software available either from Tandy or from other suppliers. With the review system we received a selection of software packages. They include:

Statistical Analysis—a suite of programs to handle data file preparation, data file lists, random samples, descriptive statistics, histogram, T-test, simple correlation and linear regression.

Editor/Assembler, Math I—a suite of programs for teaching maths to first-year students or for those with some form of learning disability. Covers addition, subtraction, division, multiplication, and an evaluation test.

In-Memory Information System—a suite of programs designed for creating, saving, retrieving, modifying and sorting data. Data can consist of names, salaries, stock numbers.

Algebra I—a suite of programs providing fundamental basics of algebra.

Level I Basic instruction course—a suite of programs designed to teach you Basic.

Business programs

There is also a range of single cassette programs for converting data and programs from Level I and Level II; unfortunately they do not work on all of the programs. There is also a T Bug. On the lighter side, there are cassettes for backgammon (hit 99 to take blots off); blackjack (we're told it will take negative bets); a stupid game called Quick Watson; a message centre tape; and a recipe conversion (pints to litres and so on).

Both the Level I and Level II manuals have a variety of games and business-type programs you can set up yourself. These are good for giving hands-on experience. Among the programs are. Space Ship Lander, customer information mailing list/phone list; triangle computation; target practice; ready-aim-fire; a 12-hour clock; Dow Jones' industrial average forecaster; board games; automatic ticket number drawer; home security; loan amortisation.

For the future, Tandy says it is now receiving and developing literally hundred of games and business applications, so there should be a wide range of applications packages available in the next six to nine months.

Technical Specifications

HARDWARE

Microprocessor: Zilog Z80 (8-bit processor).
Keyboard: Integrated ASCII, 53-key QWERTY-type.
Video display: Memory mapped, all graphics and alphanumerics controlled by Basic commands. Cursor control. Automatic scrolling.
Text: 16 lines of 64 characters, also software selectable to 32 characters per line.
Graphics: 128 horizontal by 48 vertical. Graphics and text can be interspersed in any manner by software.
Memory: Includes 4K Read-Only-;emory (ROM), 4K dynamic Read-Write Memory (RAM). Internally expandable in the computer case to 12K ROM and 16K RAM. Total memory capability of 62K.
Input/Output: Computer-controlled cassette interface. Expansion port for additional memory and peripherals. Keyboard built-in.
Electrical: 220/240 v AC, 50 Hz.
Dimensions: 400 × 200 × 90 mm.
Video Display: 30 cm diagonal screen, 420 × 340 × 300 mm.
CTR-41 Data Cassette: Battery or main operation.

SOFTWARE

Level-I Basic in ROM
Level I Features: standard Basic statements; floating point arithmetic; numeric, array, and string variables; video graphics commands; cassette save and load commands.
Commands: NEW, LIST, RUN, CONTinue, REMark, LET, FOR-NEXT-STEP, GOSUB-RETURN, STOP, END, GOTO, IF-THEN, INPUT, ON . . . GOTO, ON . . . GOSUB, PRINT, CSAVE, CLOAD, DATA, READ, RESTORE.
Functions: MEM, TAB, INST, ABS, RND, +, —, *(multiply), /(divide), ⟨, ⟩, =.
Special Commands (Including graphics): CLS clear screen), SET (x,y), RESET (x,y), POINT (x,y), formatted PRINT. Array and string capability. Data storage and retrieval.

Level II Basic features everything in Level I, with the exception of the shorthand. Obviously, to list everything would take the rest of the magazine. So highlights in no particular order:

SHIFT @	Pause in execution; freeze display during list.
BREAK	Stop execution
$	String
%	Integer
!	Single Precision
#	Double-precision
AUTO	Turns on automatic number system
CLEAR	Set numeric variables to zero, strings to null.
CLEAR n	Same as Clear but also sets aside n bytes for strings.
CONT	Continue after break or stop
DELETE, EDIT, LIST, NEW, RUN, SYSTEM, TROFF (turnoff trace) TRON (turn on trace) PRINT, PRINT @n, TAB, DATA, READ, RESTORE.	
PRINT #-1	Output to cassette
INPUT #-1	Input from cassette
LET; END; STOP; GOTO; GOSUB; RETURN; ON exp GOTO line # 1; . . . line # k; smae with GOSUB; NEXT; ERROR; ON ERROR GOTO; RESUME; RANDOM; REM.	
Graphics Statements:	
CLS	Clear screen
RESET	Turn off the graphics block
SET	Turn on the graphics block.
Special Statements:	
POKE	load value into memory location
OUT	send value to port
ERL	Returns line number of current error
MEM	Returns total unused and unprotected bytes in memory
PEEK	Returns value stored in specific memory byte
POINT (x,y)	Checks the graphics block specified by coordinates x and y.
USR	Branches to machine language subroutine.

Prices

All Tandy systems include in the price a 12 in, monitor, a battery/AC cassette recorder. power supply, 232-page manual, and a two-game cassette.

Level I with 4K RAM	£499
Level I with 16K RAM	£728
Level II with 4K RAM	£578
Level II with 16K RAM	£807
Expansion interface, which contains sockets for additional 16K or 32K RAM and a disc controller for up to four mini-floppies (requires Level II Basic)	£229
Mini-disc system (Essential in our opinion if you intend to do business applications. Needs 16K RAM, Level II and expansion interface.	£399
Screen printer (SCI rotary printer)	£479
Centronics printer (60cps)	£999

Configuration of a typical business system: TRS-80 with built-in keyboard, 12 in. video display, 16K RAM, cassette recorder, line printer, Level II Basic, Floppy disc system, expansion interface, game cassette. Price: £2,434.

A list of the retail outlets is available from Tandy, Bilstone Road, Wednesbury West Midlands.

CONCLUSIONS

- **Much depends, of course, on the purpose for which you intend to use the system.**
- **The Tandy arrives in separate parts and you cannot buy just the processor and keyboard, which is a pity as most homes already have a TV and a cassette recorder. Packaged in separate parts may be a disadvantage in educational applications as individual items are more prone to disappearing than a single unit which could be bolted to a desk. It's also awkward to move around.**
- **We found the cassette somewhat erratic and difficult to use. Practice and getting used to the volume settings may overcome this problem. The video display was not really up to the standard we would expect for the price. Ours wobbled a lot. But as mentioned this has been rectified.**
- **The keyboard is, however, attractive. It seems robust enough, and touch-typing is certainly much easier than with the Pet. The on-off and res-set switches are positioned badly and not very easy to operate.**
- **Graphics are not so brilliant. There is no lower-case on the display. The data files are limited on the cassette.**
- **The Basics are incompatible between Level I and Level II and, although there are program and data conversion programs, they do not work for all the programs. Even though you can identify the statements which need changing, we still think it is a headache.**
- **Application packages are in short supply at present, so if you are thinking of running any business applications you will have to develop them yourself for the most part. Tandy, however, says that it has a number of applications under development, either by itself or by users, and as they become available we shall review them.**
- **The most redeeming point is the manuals. For someone who wants to learn about computing or play with a computer, the system is good value for money. For £499 you have a complete system which will teach you all about computing and programming in Basic. And it is, of course, capable of being upgraded.**
- **A further important point in Tandy's favour is that the system is available through its 180 retail outlets. It's easy to buy and if anything goes wrong, there is probably a shop near you.**
- **In summary, therefore, the Tandy represent fair value for money if you want to learn about computers and programming in Basic from scratch.**

PET GOES

This month we look at how the Pet can be used in setting-up a computer laboratory in a school and also examine a Pet Basic program, Storyteller, which shows why so many teachers are excited about the future of personal computing in schools.

by David Smith.

MUCH to my dismay, I discovered that the average student really doesn't care about computers one way or the other. In fact, this may be the great fallacy we have yet to face. Until now, computers in schools have been limited to the special experimental kids, or to bright, gifted students. Like personal computing, computers haven't really reached the average student who does not already have an interest in computers.

Although having a real-life computer in the classroom was much more interesting than a terminal over the telephone, I found that once the novelty wore off, the students lost interest except for the gifted ones; who were bored with the standard fare anyway. And being a single user system didn't help matters. No logistics program in the world can schedule one computer among five classes of 35 to 40 students. But the worst problem of all was that a computer is, frankly, a delicate machine. What was needed was a low cost, compact, indestructable package that could survive the school environment.

PET's biggest asset

The Pet is, in my opinion, typical of such a computer. It features a very good quality monitor and cassette unit built in, so there are no loose parts to be pirated away. The computer is compact and portable, and can easily be secured to a desk-top with a single bolt, making it secure and student proof. Many people have complained about the small calculator keyboard, but none of those have been students. Most students shy away from a typewriter anyway, but calculators don't scare them.

The biggest asset of the PET is its low price and ease of use. At last here was my ultimate student computer.

The key to the problem of computer availability is to buy a complete set of personal computers and equip a computer lab which can be made available to the whole school. At $595, a classroom set of 25 PETs is to $14,875 (U.S. prices). That really isn't very much when you consider how many more students will be reached through the computer lab than from a single expensive S-100 type system. Also, by the time the single user program is built up enough to make it usable, the school will probably spend close to that figure anyway.

So why not go for all the marbles up front and get the kind of computing power which can be made available to all the students, not just the gifted or computer freak types?

If you run into trouble at that figure, you can always back down one PET at a time until you find a figure your principal and school board will swallow. Then later, after you show the success of the computer lab you can increase the number of PETS, which you will have to do once students catch on to the fun of working with their own personal computer.

Instead of trying to schedule students one at a time, teachers can then schedule whole classes for the computer lab, knowing each student will have his own computer to sit at and manipulate. Your third-period class might meet every Friday in the computer lab to explore computer applications in chemistry. Each student would propose a computer simulation project relating to chemistry, and each Friday the class would work individually in the computer lab, simulating some aspect of chemistry.

The projects might run four weeks at a time, with a new project every four weeks. Or they might be open-ended, with the student tailoring his project based on its complexity and the amount of computer time he needs; but the crux of the situation is that each student can work at his own computer and at his own pace. That is the key to providing useful computer experiences in the schools.

Perhaps some students are having a rough time in history. So their history teacher schedules them for computer lab twice a week. Instead of writing their own program as for the chemistry projects, they would make use of canned programs in history prepared by their history teachers, or purchased from outside vendors of software.

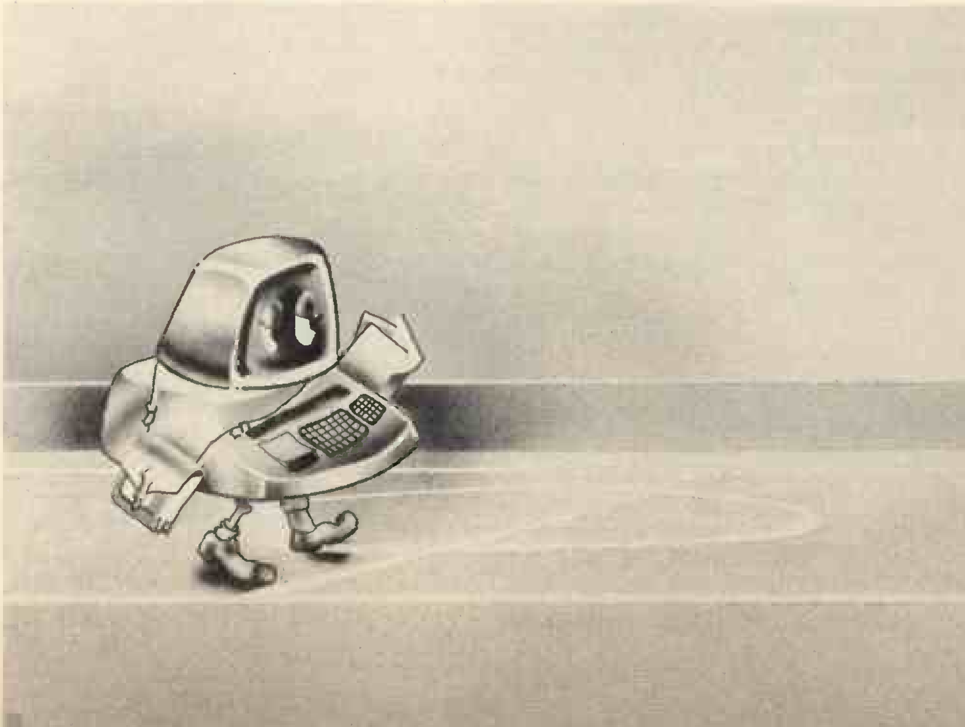
The programs would be like programmed texts, except that an entire term's course in history would be contained on several computer tapes. The student could go through the tape at his own speed, getting feedback from the computer, which would scan his input searching for the correct ideas. Unlike texts, this "book" would be animated and would add a level of interaction to the learning process programmed texts cannot provide.

Move interaction

With individual computers to work from, a single teacher could have much more personal interaction with the students, because he would be free from supervising students from the front of the room, and the computer would keep each student occupied at the interest level appropriate for that student.

We would no longer have to be satisfied with teaching to the average student while losing the slow learner and boring the bright students. The computer lab could take us one step closer to true individual

(continued on next page)



TO SCHOOL

(continued from previous page)

instruction, without having to reduce classroom sizes to 20 students to do it and that could be very important in getting more from fewer teachers. What is needed now is software. The PETs and other low-cost computers are here. What is not here is the canned course library of curriculum materials and application materials and software for using the computer.

Educational software is probably the most neglected area of personal computing. The hardware is here. The systems software and Basic has arrived this year. The next big arrival will be the applications software; and after the applications software will come educational software, which always is the last to arrive.

First of all, we need canned self-instruction programs which help the slow student with computer-assisted instruction tapes paralleling whatever course he may be having trouble with. In short, we need a set of canned CAI programmed texts in computer format which follow the traditional school curriculum. Tapes in history would follow closely a standard history text, but allow interactive learning by discovery and creativity on the part of the student. This would be an animated history text which could come alive on the video screen for the student in response to his inputs—not stale yes-no lists of questions on history, but a dynabook approach taking advantage of the animation and graphics capability of PET-type computers.

Next we need non-traditional curriculum materials for the new wave of computer science-type courses which are slowly becoming approved. The programs would challenge the students' creativity to use each new computer topic in his own programs. A program of this type would be a new topic such as sorted lists, and then input the student's own program on sorting lists and simulate its execution, reporting to the student how successfully he had mastered the ideas of a sorted list.

Gaining insight

The previous two types of software are self-instructional course materials which would allow an algebra student, for example, to complete a term's course in algebra at his own speed in the computer lab. The computer would test the student as he went along and his progress would be printed out on the printer for review by the student in conjunction with the teacher who sent him to the computer lab in the first place.

This type of program is fairly easy. You just follow more or less a standard text on the subject, adding interactive dialogues with the student as you proceed. More difficult is the third type of software needed—applications software to augment subject areas in other courses that are not traditionally orientated towards computers.

Nearly every course could benefit from a computer if we could think of a clever way to apply a computer to that subject. In this area, we are like the early automobile, looking for a road to go somewhere. Let us see if we can think of some ways computers could be used in non-mathematical courses.

Biology would be an ideal course for computer lab time. Students could investigate the interaction of life and nature in such simulations as the great whale simulation which portrays the hazards of a herd of whales as they try to migrate north. Decisions by the students will determine whether the herd makes it or not and those decisions must be based on the student's grasp of the ecology of marine life.

Other simulations involving genetics and the ever-present population controversy could further spark a students visit to the computer lab for further study. Teachers do not have time to create all this software themselves. It needs to be provided in quantity and quality.

With the advent of voice synthesisers and speech recognition units for the S-100 bus, the future for the language lab at your local high school may really be a disguised computer lab. Students could speak into the microphone with the computer printing out or repeating what they said along with the correct pronunciation in the student's headphone. Again this would be interactive, with the student creating a story in English and then read ing it in Spanish or some other language into the computer, with the computer repeating the correct pronunciation.

It may be some years before this type of capability reaches school level, but until then the computer could manage pronunciation lists of words for drill and other mundane tasks needed in learning language.

Physics is an easy one, since it is simply applied mathematics. Wonderful simulations could be worked out by the computer which would combine maths with science. For example, one I use in my classes is for the students to determine when a golf ball will land in the hole, in a bunker, or over the green, given the club used and the initial velocity applied to the ball.

Making some assumptions relating the club type to the angle of trajectory, the problem becomes an exercise in analysing the quadratic equation, which governs the parabolic flight of the ball.

Students gain insight into the geometry of the parabola, the algebra of quadratic equations, and the physics of motion in two dimensions. All of those ideas can be simulated on the computer, which could ask for the student's response to the problem and reply with a graphic animation showing the flight of the golf ball based on the equation the students gave the computer.

Graphic capability

This is possible on the PET and other computers with graphic capability, but the software orientated towards helping the student learn the physics involved, is no available yet.

Contrary to what some might think, English offers a host of reasons to visits the computer lab. Interactive creative writing could be done with the computer providing prompts to help simulate the student's creativity. After the essay has been typed-in by the student, the computer would go into edit mode, allowing easy correction

(continued on next page)

(continued from previous page)

of the paper by the English student, and finally, printing out the paper for the student.

While this might not teach penmanship, it could help teach creativity and expression, especially if the program could scan the paper for grammatical and spelling errors. As an example of this application, I have written a program for the PET called Storywriter which illustrates some ideas of mine on programming for human consumption.

We can look at this program and see what will be involved in creating the kind of educational application software we have been discussing.

The trouble with programming is that it is done by programmers. If programmers were forced to use the junk they might begin to see things from the user's point of view and begin to design programs as if people mattered. It is amazing what users and programmers alike accept in industry. If you bought a TV as poorly designed as most software, you would demand your money back before you left the store. But computer users merrily accept whatever the computer company provides with hardly a whisper. It is amazing. A much higher level of product must be developed if computers are to penetrate the schools.

Hardest aspect

The number one factor in all software design must be ease of use by the student. Second to that, the program itself must be intelligible with ample documentation to allow customising to fit individual classes and students. In this area, the microcomputer market has been especially bad. Manufacturers have provided little software and almost no documentation for what they have provided. It's sink or swim for the buyer.

Most microcomputer users are so paranoid about memory space that programming among microcomputer users has probably regressed several years in terms of modern efforts to standardise programming techniques. Believe me, any Basic program can become quite unintelligible to even an advanced machine language programmer if it is unstructured and compacted enough.

This is probably the hardest aspect of programming—intelligibility. As more and more companies produce appliance-type computers, look for longer delays between product design and market date. The added delay will be because of the higher level of software support the public in general will demand before laying out $500 for a computer. This is probably why Texas Instruments has yet to announce the long-awaited home computer. It is hiring programmers like mad to create the software library before introducing the product, but unheard of in the microcomputer business. So let's try to design a program that will be intelligible in its design and coding as well as use.

Children love to create funny stories. Storyteller provides a framework for kids to expand on the traditional *Goldilocs and the Three Bears*, by making it into their own story. By placing the story text in data statements we can create a framework adaptable to any story setting.

Let's look at the first piece of documentation any program should have. A high-level top-down structured flowchart which tells what the program does in plain English. In figure 1 we see what Storyteller does. The key to the program is the menu selection routine. This module puts a menu on the screen for the user to select what he wants to do.

Storyteller program

In figure 2 we see the PET display immediately upon loading Storyteller into the computer.

This menu meets rule number one—a program must be easy to use. By use of the GET command, PET will ignore any key pressed except 1, 2, 3, 4, 5, or the stop key. Any other keys may be pressed all day, but the program will continue to display the menu. In addition, after executing each major function in the menu, the individual modules return to the menu routine as shown in the high-level flowchart.

Thus there is no ambiguity about how to use the program and the user is protected from his own foolishness. This is prime pre-requisite for programs used by students.

Another feature of the GET command is that it eliminates the need to press the return key all the time. This is very useful because teaching the use of the return key is a major obstacle in letting a novice sit down and use a computer.

One feature of top-down modular design is that it leads to top-down testing. Once the menu selection routine is written, the Storyteller program can be run and tested using dummy modules for each of the five major functions the menu selects.

In this way, the major branching functions of the program can be tested immediately, and the lower level modules can be built up as time permits. This provides immediate feedback on the success of the program while it is being designed and means faster program development, but it takes more space and more careful attention to simplicity to preserve a top-down flow from one module to the next.

In figure 3, we see the high-level flowchart for the first module on the second level. This module creates the new story, and in turn, calls subroutines, which represent the third level of program development. Each of the five menu function modules has a similar flowchart for it.

In turn, the subroutines called by the main function modules have flowcharts representing level three and so on until we get down to the details of text formatting and the alike.

Fits user needs

This top-down coding process may cost more time in proper design but it more than compensates for it in producing a well-documented, easily-followed program which can be understood six months after it was written; and clarity and simplicity are a must for educational software.

Because the PET normally displays

(continued on next page)

(continued from previous page)

upper case and you must press shift to get lower case, some key processing must be done on each key to reverse this. Again this is designing for human consumption; making the computer fit the needs of the user.

This is done by using a keystroke processor routine. The ASCII values of the upper case letters are 65 =A to 90 =Z. The lower case letters are 193 =a to 218 =z.

By using the GET command, each keystroke can be examined and converted by adding 128 to its ASCII value if it is upper-case, and subtracting 128 if it is lower case. In this way, we reverse the normal keyboard operation so that a shifted key is upper-case and a non-shifted key is lower-case. A routine to print each character as it is entered is also necessary, since the GET command takes characters on the fly without halting the CPU and does not echo them.

We can include a false cursor also, if we remember to backspace and erase the false cursor when we stop GETting keystrokes. To escape the entry mode, we will test the keystroke for an ASCII 13 or carriage return. Then we can gather all the keys which were input and store them in a string matrix called Text$(I) so our responses will be available when we want to print our story.

This article is based on a talk given by U.S. teacher David Smith at NCC '78, a U.S. personal computing show and conference. Since giving that talk at NCC, Smith has bought an Apple II and is now implementing the ideas expressed in the article, including a complete course as a series of 40 lessons or programs chained together on the Apple floppy disc to provide a complete high-school algebra course.

The courseware features extensive use of the Apple's high resolution graphics to display mathematics symbols and plots student's functions so they can see how algebra relates to geometry as they study the subject. The entire course is on a single Apple floopy disc and will be marketed under the trade name Compu-Tutor *sometime soon.*

Smith would enjoy corresponding with anyone in England who is interested in developing subject courseware for microcomputers at the high-school level and would also be happy to write to anyone who might like more information on microcomputers and education.

He is head of the local PET-Apple Users' Group for the North Orange County Computer Club, whose activities include publications of a useful-sounding 500-page PET Notebook of PET documentation.

If you wish to correspond please send your letters to Practical Computing *and we will be pleased to forward them.*

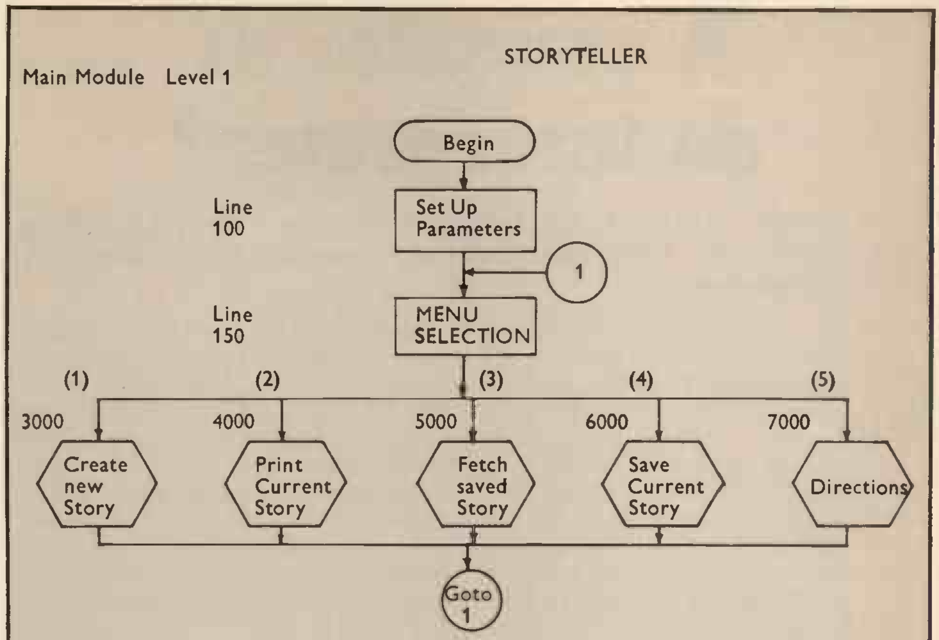


Figure 1. Main Flowchart—"Pet goes to school"

PET DISPLAY FOR STORY TELLER

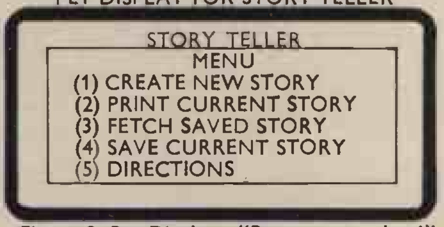

STORY TELLER
MENU
(1) CREATE NEW STORY
(2) PRINT CURRENT STORY
(3) FETCH SAVED STORY
(4) SAVE CURRENT STORY
(5) DIRECTIONS

Figure 2. Pet Display—"Pet goes to school"

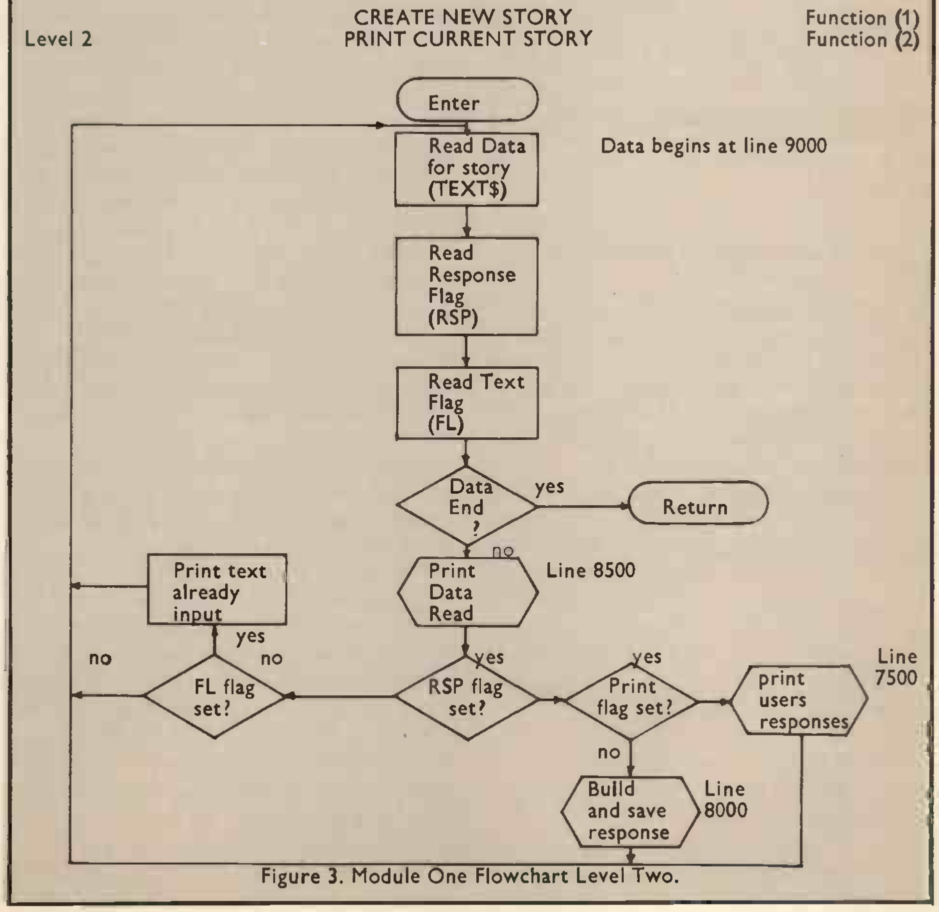


Figure 3. Module One Flowchart Level Two.

A compiler or an interpreter?

This month we examine the difference between a compiler and an interpreter by studying what Software Dynamics has to offer with its Basic compiler.

IN ALL STAGES in the development of computers, software has lagged behind hardware, and the microcomputer world is no exception. With the notable exception of Altair, most systems a year ago had no disc operating systems, and software was restricted to an editor, assembler, and simple Basic interpreter. Much of this early software was rather amateurish. Basic became popular not only because it is a simple language in which to write simple programs, but because it is much easier, and therefore cheaper, to develop an interpreter than a compiler.

As a sign of the growth of the microcomputer world, there are now some excellent softwate products available and included among them is the Software Dynamics Basic compiler.

Software Dynamics Basic has all the normal features of Extended Basic including:

—string variables
—output formatting with PRINT USING statement
—ON ERROR processing
—IF/THEN/ELSE statements.

It also has some very unusual features, however. Statements need only be numbered where they are referenced from another statement, for example GOTO or GOSUB. The numbers do not define the order in which the statements are to be processed.

Numeric variables

Variable names may be at least five characters long; the actual length is defined by the assembler used to process the output from the compiler. Those two points taken together mean that source programs are easier to understand than normal Basic programs.

Numeric variables occupy six bytes and the compiler decides whether they need to be held in integer or floating point form. Where numeric vairables are held as integers processing is much faster. For example, in the following two loops (a) will execute more than five times faster than (b)

```
(a) FOR I = 1 to 1,000
        NEXT I
(b) FOR I = 1·0 to 1,000·0
        NEXT I
```

String variables can be up to 65,534 bytes long and substrings are defined either as B$ (3,2), which is the string composed of the third and fourth bytes in B$, or using the MID$, LEFT$ and RIGHT$ functions. The notation B$ (3) represents the value of the third byte in B£; this is used in place of the CHR$ function which is found in other Basics to put control characters in a string.

Subroutine handling users the GOSUB and RETURN statements and additionally there is GOSUB POP, which allows one or more return addresses to be removed from the stack without transferring control. GOSUB POP 0 empties the return stack.

Manuals very good

The file handling is dependent to some extent on the operating system being used with the compiler. Files may be ASCII or binary and can be processed sequentially or randomly. Statements are included for opening, closing, deleting and re-naming files.

The technique used for accessing files randomly is very neat. Most Basics which allow random file access do so in terms of fixed length records; for example, Digital Basic-Plus works with 512-byte records, and it is the programmer's responsibility to handle records shorter or longer than this standard size. Software Dynamics Basic treats a file as a continuous string associated with which there is a pointer to the part which is being processed. This pointer can be moved to any position with the RESTORE statement. The byte position can be defined by an expression or a function, for example the statement

```
RESTORE      2, RECORD (10)
```

moves the pointer to the file open on channel 2 to the position defined by RECORD (10). RECORD would be a function defined using a DEF statement. This allows logical records of any size to be processed without the programmer being concerned with the physical disc block size.

One other unusual capability of the language is the CHAIN statement which allows control to be transferred from one program to another. This in itself is not an unusual feature of disc Basics but the compiler allows data and program origins to be specified, so that data can be passed in memory from one program to another; thus chaining is more akin to overlaying than simply passing control from one statement to another.

Our experience of using the compiler has been on an SWTP 6800 with mini-floppy discs and the Smoke Signal Broadcasting System. The documentation supplied was Software Dynamics Compiler Manual, Software Dynamics Assembler Manual, Notes on Using the Compiler with the SSB operating system, and A listing of the I/O package.

Both Software Dynamics manuals are very good. The compiler manual includes plenty of supporting information such as data formats, the I/O package interface, timing tests, and sample programs.

A separate editor is required to enter the source program and we have used the TSC Editor which is very powerful. Once a program has been entered it must be compiled, assembled and run with the run-time and I/O packages.

A sample dialogue is shown below with input underlined.

```
& RUN I :SDBAS
SOFTWARE DYNAMICS BASIC VERSION 1·2
INPUT FILE = ICFMZ
OUTPUT FILE = JUNK I
COMPILATION COMPLETE

& RUN I :SDASM
MAL/6800, VERSION 1·2/37E4
SOURCE FILE = JUNK I
LISTING FILE = CR
BINARY FILE = ICFM.BIN
  CR
* * * NO ERRORS
& I :SDRUN ICFM.BIN
```

This becomes fairly tedious, especially when all that is required to run a normal interpretive Basic program is to type RUN. Soon there will be a new Software Dynamics operating system, SDOS, which will greatly simplify this process.

The run-time package provides debugging aids which allow a program to be interrupted, stop at a specific line number, execute one statement at a time and print a trace of the statement number being executed. There is no facility for examining the values of variables and this must be done using PRINT statements inserted before the program is compiled.

CONCLUSIONS

- **The main advantages of using a compiler instead of an interpreter are that programs execute more quickly and require less space.**
- **If you are developing applications software packages, your source code can be protected as only compiled programs need to be supplied to users.**
- **The Software Dynamics Basic is one of the best extended Basics available.**
- **Against those advantages must be set the fact that the inexperienced programmer will find program development and debugging more difficult with a compiler.**

Hop on a Nasbus to 32K of memory now

The Nascom – 1 is designed with expansion in mind. This is made possible by using the best products available. The Z80 microprocessor incorporated in the basic system is so powerful it can support 64K bytes of memory and 256 ports. To utilize this capability, we have designed the buffered 77 – way Nasbus.

With this arrangement, the way is clear for considerable expansion, starting with our new memory expansion board. It has 16 memory sockets and two EPROM sockets. Therefore, you can fill it with 4K dynamic RAM up to a maximum 8K or with 16K dynamic RAM up to a maximum of 32K. A 2K Tiny BASIC in EPROM has been developed for the board.

To go with the board, we have produced a very flexible I.O board with three PIOs each giving two, 8bit ports, plus a UART for serial interface.

As you start building up your Nascom system you will need a convenient means of storing boards. Our new, custom-designed, Vero frame will allow for a Nascom – 1 to link through a buffer board to a 77-way Motherboard. There is then the option of eight or more expansion boards. To power this capability there is a new 8.5 amp power supply especially designed for the frame.

No other system offers so much at such a low cost. And it all starts with the basic Nascom – 1 kit which for just £197.50 offers an intelligently usable system with video and cassette interface, a full alpha-numeric keyboard and a mighty CPU chip. So if you want the best – make it a Nascom system.

Nascom-1 Kit still only £197.50 +VAT

UK National Distributors

Camera Centre,
Barrow-in-Furness, Cumbria
Crystal Electronics,
Torquay, Devon
Electrovalue,
Egham & Manchester
Eley Electronics,
Glenfield, Leicester
Henry's Radio,
London W2

Lock Distribution,
Oldham, Lancs
Lynx Electronics,
Chesham, Bucks
Microdigital,
Liverpool L2
Teleplay,
New Barnet, Herts

Stop press...

Microcomputer Seminar
Nascom Microcomputer's highly successful seminar is coming to Bristol. The programme will be similar to London and Manchester, both of which were sold out. The day includes five lectures, demonstrations and an open forum. Venue is the Dragonara Hotel, Bristol, Saturday, October 14th, 09.50 to 17.30.

Admission: £4.50 (inc. VAT). Lunch will be available at £4.00 (inc. VAT) per head if there is sufficient demand.

Please send me
.... tickets to your seminar at £4.50 each and further details on Nascom-1 expansion products/Nascom-1 Kit/Int. Nascom Microcomputer Club.*
*delete as applicable

Name ________________________________

Address ________________________________

____

Tel. No ________________________________

Nascom Microcomputers
92 Broad Street, Chesham, Bucks.
Tel: (02405) 75151

Cut out coupon and post to Nascom Microcomputers. Cheques and PO's should be made payable to Nascom Microcomputers.

PC1178

● Circle No. 115

Seed's System

The Seed MSI 6800 System starts at less than £500. It looks a promising system for the small business user, writes Martin Collins.

STRUMECH Engineering Ltd is a company of structural and mechanical engineers whose main business is the manufacture of telescopic towers and general fabrication work. Its involvement in microcomputers derives from one of its directors, whose hobby is electronics.

He bought an SWTP 6800 system as a kit and assembled it. From the interest it generated with the company, Strumech decided to set up an electronics division to market small systems—and SEED—Strumech Engineering Electronics Division—was born.

SEED was sown in August, 1977. The systems it sells today are all based on the SS-50 bus. The principal manufacturers whose equipment SEED offers are SWTP and MSI for processors, Smoke Signal Broadcasting for floppy discs, and peripherals from Soroc, Tally and Centronics.

Assembled or in kit form

There is no space in this article to argue the merits of one bus design against another but the SS-50 has the advantage of being cheaper and, in theory at least, more reliable than the S100. It has disadvantages—the more exotic peripherals such as colour graphics and voice input are not available; but the SS-50 bus provides a good base for a straightforward commercial or educational system.

SEED has sold about 100 systems so far. The majority are MSI 6800s, but it also supplies desk-mounted systems using the SWTP 6800 and Smoke Signal mini-floppy discs. SEED's biggest market so far has been in education systems and customers include Birmingham Educational Advisory Sector, which has 10 systems and polytechnics in Manchester, Bristol and Leeds.

Systems can be supplied either assembled or in kit form, though SEED makes no secret of the fact that it prefers to sell assembled systems—the cost of subsequent support involved in fixing systems poorly assembled on the kitchen table eats into the low profit margins on the kits.

MSI is Midwest Scientific Instruments. It was set up about 10 years ago to develop and market medical laboratory instruments; its first micro product was the FD-8 floppy disc, a system which was required for MSI laboratory work. It was designed to plug into any micro system via a parallel interface.

Along with the disc, MSI sold an operating system and a Basic. They were written for the Motorola 6800 and designed to be used with the SWTP 6800. Having co-operated closely for some time, MSI and SWTP are now direct competitors. Last year MSI introduced its own system, the MSI 6800. This also uses the M6800 micro, of course. Since then competition has intensified, with SWTP introducing mini and full-size floppy disc systems.

A more robust system

The MSI 6800 computer system is in a free-standing cabinet which holds the processor board, power supply, motherboard, fan, up to 56K bytes of memory, and up to eight interfaces to peripherals like terminals and printers.

The MSI 6800 appears to be a much more robust system than the SWTP 6800. Specific advantages of the MSI 6800 include: the use of gold-plated connectors on the motherboard and interface adaptor board; the motherboard itself is thicker, better supported, and does not flex when boards are inserted into it; the front panel includes buttons to generate the non-maskable interrupt (NMI) and the interrupt request (IRQ) as well as RESET and power on/off; the cabinet is more robust and can include a fan.

On the CPU board there is a monitor program in prom which takes the place of a traditional front panel with lights and switches. The MSIBUG monitor supplied with the system is a great improvement on the MIKBUG monitor developed by Motorola for use with the 6800. Specific advantages are: only memory and interface addresses above 56K are used, allowing the memory space below that to be allocated as required; the control terminal is interfaced via an ACIA, allowing any type of terminal to be used; improved commands to examine/change memory, start program, and dump register.

As much of 6800 software uses routines in MIKBUG, MSIBUG has these routines with the same addresses. In addition to the MSIBUG prom the CPU board has sockets for up to 4K of EPROM memory.

Serial and parallel interfaces are plugged into the interface adaptor board, which is plugged into the mother board. Memory options for the system are 8K RAM, 8K EPROM, 4K PROM/RAM, and 16K RAM.

With the exception of the 16K RAM board, all are manufactured by MSI. They are available in kit form or assembled and with or without sockets. The 16K RAM board is manufactured by Smoke Signal Broadcasting and is available only assembled without sockets.

SEED can supply two VDUs for use with the MSI 6800, the ACT-1 and the more sophisticated SOROC IQ 120. The SOROC units offer such facilities as cursor control, separate numeric pad, and speeds from 75 to 19,200 buad.

Good software products

Discs available are the FD-8 full-size floppy disc and the Smoke Signal Broadcasting mini-floppy; the FD-8 allows up to four drives to be interfaced, each of which can hold 300K bytes.

There is a also cassette tape interface which operates at 300 baud to the Kansas City standard with a normal audio tape recorder. SEED can supply Centronics or Tally matrix printers.

Software is all disc-based. Systems with the FD-8 use FDOS: this supports the Motorola CORES Editor and Assembler, Basic in interpreter and compiler versions, the TSC text editor, and a variety of utilities.

FDOS is a fairly simple operating system. Files are allocated as contiguous disc space and to recover gaps which have been left by deleted files, the discs must be packed—an operation which SEED admits is somewhat error-prone.

The interpretative Basic is the same as SWTP Version 2 Basic with the addition of disc files. This is reliable but rather slow; it does, however, include some useful features such as nine-digit floating point.

The Software Dynamics Basic compiler and the TSC Editor are both extremely good software products. The compiler offers a greatly-enhanced version of Basic, including: string, integer, and floating point variables; variable names of up to six characters; random and sequential disc files; chaining and the ability to have common data areas between chained programs.

The compiler is more difficult to use than normal interpreters but for experienced programmers this disadvantage is greatly out-weighed by the improved language. Compiled programs will normally occupy less space than interpretative ones and run very much more quickly. For example the loop—FOR I = 1 TO 1000 NEXT I will take about two seconds with the fastest interpreters and less than 0·2 seconds with the compiler.

The Software Dynamics Basic is re-

(continued on next page)

(continued from previous page)

viewed at length on page 28 of this issue.

The documentation for the system includes very detailed descriptions of the hardware, assembly instructions—even if you buy an assembled system—theory of operation, circuit diagrams, and check-out and trouble-shooting instructions.

Very competitively priced

The software documentation however, is more of a mixed bag. FDOS is described in three pages; the information on CORES is limited to how to use it under FDOS. There is no user manual. The Software Dynamics compiler has a very good user manual but the documentation on its use under FDOS is rather scrappy.

The system is priced very competitively, whether compared to SWTP 6800 or S-100 systems.

	Kit	*Assembled*
MSI 6800 with 8K and serial interface	£ 375	£ 565
8K RAM	141·75	211
16K RAM	—	350
Serial Interface	36·50	55
Parallel Interface	31·20	41
FD-8 Dual Drive with controller	—	1640
BASIC interpreter	—	48
BASIC compiler	—	245

SEED has released details of new disc and software products which will be available later this year. They are:

FD-8A, an improved double-density-version of the FD-8. Two drives and controller will cost £1,450.

FD-8Q, with double-sided double-density floppy discs giving 2·4 megabytes on two drives. The price for two drives and controller will be £1,875.

HD-8R, a 10-megabyte disc drive (five fixed and five removable); the drive and controller will be £4,250.

SDOS, a new operating system, developed by Software Dynamics: it will support any combination of the new discs listed.

There will also be a packaged version of the system, the MSI System 12, which has in a single desk unit, the MSI 6800, with one or two floppies and the rigid disc.

MSI intends to use the products to become established in the small business system market. SEED already has a working HD-8R in the U.K. but the new operating system has not arrived yet.

- **Once the new disc and software products arrive in the autumn, the MSI 6800 will be the only system here which will start at less than £500 (the basic kit including processor, interface, and 8K) and expand to a full 10MB rigid-disc system. That 10MB configuration will also be priced extremely competitively.**
- **With the exception of the mini-floppy disc the complete system is from a single manufacturer. There should be no compatibility problems.**
- **If the new SDOS operating system lives up to the high standard of the Software Dynamics Basic, it is going to be good. Certainly the initial specification looks good.**
- **With the current mix of distinctly variable software documentation, the system probably is not the best on the market for the first-time user.**
- **The ability of SEED to support the hardware and software is likely to become stretched if sales continue at the present level. To overcome this it is looking for dealers in various parts of the country.**
- **Maintenance is available on complete systems from CFM, and once some accounting packages have been developed the system will probably be a good bet for the small business user.**

HEATH'S KIT

LOOKING at Heath's modest one shop in London's Tottenham Court Road, you would not imagine Heath to be such a large operation in the United States. In fact, the company is the third largest home computer firm in the U.S.A. behind Commodore and Radio Shack (better known to us as Tandy).

Heath has sales of around £45 million per annum of which computer product sales comprise 10–15%. Heath's main line is in consumer electronics, especially hi-fi kits.

The two main computer products are the H8 'Mainframe' based on the Intel 8080 and the H11 based on the Dec LSI-11 Micro Processors. As yet, U.K. sales for computer equipment have not been startling.

So far Heath has reported only ten sales of the H8 in the U.K. Six were sold from its shop; the other four were from the head office in Gloucester, which is a mail order operation. In the United States Heath has 52 retail outlets controlled from its head office at the twin cities of St. Joseph/Benton Harbor in Michigan. Sales of the H8 are in the region of 4,000 since its introduction about a year ago and the H11 was introduced a couple of months later. Sales for this model have reached the 2,500 mark. In the U.K. no H11's have been sold yet. Bill Schiffbauer, Sales Co-ordinator for computer products at Heath corporate headquarters told *Practical Computing* "Heath entered the analogue computers business in 1957 and was well received in the educational market, but we decided to drop this line around 1970. However, the advent of the mini computer and micro computer made us realise that there was a new trend in computing: so we decided to introduce the H8 and followed soon after with the H11".

Marketing philosophy

The H11 is very different from the H8: internally the boards are laid horizontally, whereas on the H8 they are laid out vertically. The H11 is based on the earlier board but according to Schiffbauer "we should be offering the H11 with the LSI-11/2 board by the end of this year or early next year.

"On October 15 this year we introduced a full-sized floppy disc drive for the H11 which is totally hardware compatible with Digital's equipment. It will also be software-compatible, as far as Digital software is concerned, on the H11. Software developed on the H11 will not necessarily run on the LSI-11, however."

Owners of the H11 are entitled to join DECUS, the official Digital user group, and also the Heath User Group (HUG published a quarterly user magazine called REMar. There is also an unofficial user magazine which gives users tips over and above those given by Heath.

As the name implies a Heathkit computer is supplied in kit form and assembled by the user, in common with most Heathkit products. One exception is the minifloppy disc drive offered as an add-on to the H8; but according to Kelvin Higham, who runs the Heath shop in London, the drive will be available in kit form towards the end of the year.

Heathkit computers are very much orientated towards electronics hobbyists who are looking for complete systems to take them straight into computing *per se*, rather than building customised systems from each component.

This marketing philosophy is very much that of the hi-fi hobbyist, someone who goes in to a shop and buys a kit

(continued on next page)

(continued from previous page)

which includes the turntable, amplifier and speakers. In the same manner Heath offers compatible peripherals.

The H8 and H11, look much the same except for their front panels.

The H8 front panel has an octal keyboard and a nine-digit light-emitting diode displays (LEDs); the H11 front panel is blank, apart from labels and two switches. The reason is obvious: if one is going to spend the money it costs to buy the pretty sophistic H11, one will hardly be satisfied by pressing numerous key combinations to run Noddy programs which in the end only display a numerical result that has to be converted into decimal.

Limited expansion

There are, of course subroutines available with which to convert the results to decimal and to display them as such; but unless these subroutines are permanently stored in ROM (or, if 'slow' CMOS memory is used, kept in memory with battery power) they have to be keyed in. So there is a fair chance that once the processor has been assembled, be it the H8 or H11, there will be little by way of amusement from the front panel only.

Once one has built an amplifier, one goes out to buy a turntable, right? Disc drives are inevitably quite expensive so domestic audio cassettes may be used. A true hi-fi does not use a cheap audio cassette system to plug into an expensive amplifier, however: and in the same way a true computer will not normally go for serial storage with an audio cassette.

To make the point clearer, and again using the hi-fi anology, one can get the equivalent of a complete 'music centre'—a computer such as the Commodore Pet (reviewed *Practical Computing*) which in its present form has limited expansion facilities. 'You get what you pay for' is a phrase which applies as much to computing as to any other market. The plain fact is that you can spend considerably more money on a comparable Heathkit configuration; but there is more room for flexibility.

The H8 is based on the Intel 8080 microprocessor but the bus is Heath-designed, and for computer hobbyists in the know this means that it is not S–100 compatible; in fact it is called the Benton Harbor Bus and its non-compatibility is not a subject to which we will attempt to contribute. The H11, on the other hand, is completely DEC-compatible—but we will come to that later.

On the H8, a ROM-based monitor allows for the inspection of all registers and functions through the keyboard and LEDs. The kit costs £299 and includes system software in audio cassette form but excludes memory.

System software comprises the HASL-8 two-pass assembler, the TED-8 line-orientated text editor, and the BUG-8 debug program. BUG-8 only requires 3K memory but the others need at least 8K.

8K memory cards cost £120 each and up to four of them may be slotted into the backplane. This contains eight vertical slots, two of which are occupied by the front panel card and the processor card. "At a later stage we will introduce 16K cards, which will give the H8 full 64K addressing capability", says Kelvin Higham.

The remaining slots may be used to plug in other cards available from Heath such as the H8·2 parallel interface which may be connected to a paper tape reader/punch or a line printer. There is also the H8-5 Serial I/O and cassette interfacc to which may be attached Heath's own H9 video terminal, or a DECwriter; the controller card for the minifloppy disc drive also takes one card slot.

The serial interface features jumper-selectable data transfer rates from 110 to 9600 baud, 20mA current loop and RS232C connections, the CUTS 'Kansas City' cassette recording format, and control lines for remote start and stop of two cassette units (allowing separate recording and playback for easy program or file editing).

BASIC is an extra cost, and is supplied either in 1200 baud audio cassette form or as fan-fold paper tape. Heath calls it Extended Benton Harbor BASIC and it requires a minimum of 12K memory in which to run: however "16KB is preferred if full use is to be made of its capabilities".

No applications

The other systems software is also available in paper tape form and this too costs extra. Otherwise Heath does not dabble in applications software (except for games)—it leaves that to the user group (HUG) and its magazine REMark. Incidentally, there is no HUG in the U.K. at present but Bill Schiffbauer says that Heath is interested in starting user groups anywhere there is a demand for it.

Manuals are included in the purchase price. Heath also sells its manuals separately at £21.60 for both the H8 and H11—this is refunded once an order is placed.

The H11 is virtually a Heath-packaged

(continued on next page)

(*continued from previous page*)

LSI-11 with cabinet, front panel, power supply and fan. The LSI-11 was reviewed in our sister magazine WHICH COMPUTER? in July 1978 in its packaged versions, the PDP-11/03, PDP-11V03 and PDP-11103.

The 'bare' LSI-11 comes with 4K words of on-card memory (RAM) and when packaged as the H11, costs £1,100. This would at first glance suggest a competitive price against the PDP-11/03—which also happens to be a packaged LSI-11 costing £1,310. The difference however is in memory; the PDP-11/03 contains 8K words so to give the H11 the same amount of memory an extra H11-1 4K word card costing £199 has to be slotted in. Furthermore, on the PDP-11/03 the serial interface comes as standard; the H11-5 serial interface costs an extra £92·10.

However Heath supplies a good deal of systems software is available at no extra cost—though it all requires a minimum of 8K memory. The ED11 editor assists in the creation and modification of ASCII source tapes, and is also used to write assembly language programs and for general text editing or word processing functions.

Powerful version

PAL-11S is a relocatable assembler which converts ASCII source tapes into binary modules: this allows you to create programs in small, modular segments for easier coding and debugging. These serve as inputs to the LINK-11S linkage editor, which links modules into a load module ready for execution once it has been got into the H11 via the Absolute Loader.

The ODT-11X debug program allows for modifying and controlling of program execution "on the fly" for quick, efficient debugging. The IOX executive program permits I/O programming without the need to develop device-driving programs and links to programs using the LINK-11S (for paper tape punch or line printer). DUMP-AB and DUMP-R dumps absolute contents of memory to paper tape.

DEC's relatively powerful version of the standard Dartmouth BASIC interpreter includes string capability. Also provided is FOCAL, DEC's own interpretive language—ideal for most scientific, engineering and mathematic applications and available in 4K and 8K versions.

All H11 purchasers are required to sign a DEC licence agreement, incidentally.

With software like that who would want to twiddle around with a numeric keypad and get results from LEDs!

The LSI-11 processor card measures a standard 8·5 × 10 inches, and is accommodated in one of the backplane's four horizontal slots. Other cards, including the LSI-11/2, are half-cards (measuring only 5 × 8·5 inches). As well as the memory and serial interface cards, an H11-2 parallel interface is available; so is an H11-6 Extended Arithmetic Chip which fits on the LSI-11 card. The latter provides fixed-point multiply, divide, and extended shifts, plus full floating-point add, substract, multiply and divide.

The Heathkit H9 video terminal comes with a full ASCII 67-key QWERTY keyboard and a 12 line by 80 character display. "The 'unofficial user magazine once carried an article on how to tweak the terminal in order to get 24 lines on the display", confides Kelvin. Other features of the H9 include a plot mode for graphs, curves and simple figures, an audible alarm, auto scrolling, and basic up-down-left-right-home cursor control. Erase mode permits automatic full page erase or erase to end of line starting at cursor position. A 'transmit page' function allows a full page to be formatted, edited and modified, and then transmitted as a block of continuous data.

PRICES

	H8	*H11*
	£	£
Processor	299·00	1100·00
4K Memory	120·00	199·00
4K Memory expansion (H8 only)	90·00	—
Parallel interface	130·00	86.42
Serial Interface	108·00	92·10
BASIC (free on H11)		
—cassette	17·04	—
—paper tape	16·33	—
Paper tape system software—(free on H11)	27·40	—
Extended Arithmetic Chip (H11 only)	—	170·00
H9 Video terminal		499·00
H10 Paper tape reader/punch		307·53
Papertape (3 rolls)		16·21
Papertape (3 boxes—fanfold)		16·14
ECP-3801 Cassette drive		48·34
ECP-3801 Cassette tape		4·73
LA 36 DEC writer		1252·80
Paper (3450 sheets)		28·51
RS 232C interface		65·00
Microprocessor training course		248·70
Digital techniques course		63·46
Multimeter		40·18
Oscilloscope		164·32
Navigation Computer		130·32
Chess Challenger		210·00

The H10 paper tape punch/reader uses standard inch-wide roll or fanfold eight-level paper tape. The punch operates at a maximum speed of 10 characters per second and the reader reads at a maximum rate of 50 cps. A copy control on the rear panel permits tape being read to be duplicated by the punch for efficient and accurate tape copying.

If you are going to attach a cassette recorder for storage, it is advisable to buy it from Heath: according to the literature, "proper operation of the H8-5 and H8 software is assured only when the ECP-3801 cassette recorder and ECP-3802 tape is used. Heath does not assume responsibility for improper operation resulting from the use of any other cassette units". Sound a little like IBM?

Heath also offers the well-liked LA36 DECwriter keyboard printer terminal (also reviewed in our sister publication WHICH COMPUTER? in November, 1977). "We can supply the DECwriter if an order is put in for it but we don't actually hold it in stock, says Kelvin. "Frankly, one can buy it cheaper elsewhere". This is hardly surprising considering there has been a price war going on between the various suppliers of DECwriters since the beginning of this year bringing the average price down to well below £1,000; from Heath it costs £1,252. Heath just cannot compete with its low-volume orders.

Manuals crisp

A 5¼ inch minifloppy disc drive is now available on the H8 for £617; a dual-drive system cost £850. A full-size eight-inch floppy disc drive for the H11 is coming at the end of the year. This will mean that a Heath buyer has the full range of computing from the hobbyist through to the small business system level to choose from; he may start at any point too in the knowledge that any expediture on peripherals will not be wasted should he switch from the H8 to the H11.

The manuals are worthy of mention too; they are well produced and laid out with crisp, lucid diagrams. This is common to all electronics products from Heath. Technical jargon is kept to a minimum and a separate section at the back of each manual contains the necessary electronics theory to explain how each particular unit works.

The hobbyist would be well advised to drop in to the shop and look around—it is sure to be a treat. It *is* a low profile operation—no exhibitions, no shows, no advertising—but Heath is a name to watch out for. In its approach to computers it gets full marks for flexibility and quality.

How to play Mastermind

Bulls and Cows is a traditional game of logical deduction which recently has become well-known in the form of the coloured-peg version called Mastermind, marketed by Invicta Plastics.

In the pencil-and paper version of the game, two players begin by each thinking of a code consisting of a string of a certain number of digits. The players then take turns to make inspired guesses at the other player's code string. The guesser is given the following information about this guess:

The number of Bulls—i.e., digits correct and in the correct position in the string (bullseyes);

The number of Cows—i.e., digits correct but not in the right position.

The first player to guess the other player's code string is the winner.

Ideal game

Bulls and Cows is an ideal game to program on a computer because although good play can be obtained from a fairly simple algorithm, a good human player has the edge over even the most sophisticated computer strategies.

This article describes the essential parts of a Bulls and Cows playing program and concludes with a full listing for a program to play the game on a KIM 6502-based microprocessor system.

The program not only thinks of a code string which you are to guess but tries simultaneously to guess a code string you have thought of. Many possible variations of the game exist, depending on the number of digits in the codes and the possible values each digit may take; in the version played by the program, the codes consist of four digits, each digit being from 0 to 7.

In what follows the numbers of Bulls and Cows between two numbers will be given as two digits, Bulls followed by Cows. For example, if the code string were '6502', the replies to '4040', '6100', and '2560' would be '01', '20', and '13' respectively.

Sample description

Operation of the program is best illustrated with a description of a sample game. To start the program, the address 0200 is entered at the keyboard and 'GO' pressed, whereupon a line of dashes will be displayed indicating that the machine is choosing its code string. Pressing 'GO' again displays:

'– – – – r l'

The four dashes indicate that the program is waiting for input, and 'r l' indicates that it is the start of round one in the battle. A four-digit guess at the machine's string is then entered at the keyboard, and it will be displayed on the four leftmost displays. Pressing 'GO' instructs the machine to reveal the numbers of Bulls and Cows between this guess and its code, and they are displayed in the two right-most display positions, Bulls followed by Cows:

'0 1 2 3 0 2'

Pressing 'GO' then allows the machine first guess at your code; for example:

'2 1 0 2 – –'

The reply is entered in the same format as the machine's, and appears in the right-most two display positions. Pressing 'GO' again starts a new round:

'– – – – r 2'

The games continues until either you guess the machine's code, in which case it reluctantly displays 'I LOSE', or it guesses your code, when it displays 'Ho-ho', but allows you to continue trying to guess its code.

If you despair of ever guessing the machine's code, pressing 'DA' (display answer) will reveal its code and start a new game. If at any time an incorrect number entry is made, '+' will erase it from the display and allow you to make a fresh entry.

The subroutine which finds the number of Bulls and Cows between two strings deserves explanation. One obvious way of finding the numbers of Bulls and Cows is to make four comparisons between digits in corresponding positions to get the number of Bulls, and then make the remaining 12 comparisons between digits in different positions to get the number of Cows.[1]

Digits contributing to Bulls or Cows at any stage must be deleted so that they cannot be counted again. The execution time with this method is clearly proportional to the square of the number of digits.

The program presented at the end of this article uses a more sophisticated algorithm which differs in the way it finds the number of Cows, and its execution time is proportional only to the number of digits itself.[2] The method is to keep counters for the eight possible digits. Digits in one of the strings being compared are counted at +1, and in the other string as —1. The counters initially are set to zero. Digits in the two strings are then compared pair by pair.

Replies received

If a pair matches, a Bull is recorded and the program proceeds directly to the next pair. If, however, they do not match, the counter for the digit in one string is incremental, and the counter for the digit in the other string is decremented.

If incrementing a digit counter gives a negative or zero result, a Cow is recorded because the counter must at some earlier stage have been decremented due to an occurrence of that digit in the other string. Similarly, if decrementing a digit counter gives a positive (or zero) result, a Cow is recorded for the same reason. After the four corresponding

(continued on next page)

(*continued from previous page*)

pairs of digits have been compared in this way, the routine returns with the numbers of Bulls and Cows.

As an example, assume that the two strings are '5432' and '3552'; the diagram shows the contents of the eight digit counters after each of the four comparisons have been made, working from the ends of the strings:

better than 5·74 guesses per game? The answer is certainly yes, but at the expense of longer execution times. As it stands, the program chooses at random from the set of answers possible at each stage. Consider a game in which the guesses and replies so far are:

Guess:	Reply:
2057	02
0236	12
0736	20
0462	11

Figure 1

Compared:		Digit Counters: 0	1	2	3	4	5	6	7	Result: BULLS	COWS
2	2	0	0	0	0	0	0	0	0	1	0
3	5	0	0	0	+1	0	—1	0	0	1	0
4	5	0	0	0	+1	+1	—2	0	0	1	0
5	3	0	0	0	0	+1	—10		0	1	2

	0123	0323	5263	Expected number of guesses
0123	40	30	11	(0 + 1 + 11/3 = 2/3)
0323	30	40	11	(1 + 0 + 11/3 = 2/3)
5263	11	11	40	(1 + 2 + 01/3 = 1)

The algorithm used by the machine to guess its human opponent's code is simple but effective, and in a sample run of 1,000 randomly-generated strings, the program averaged 5·74 guesses and the distribution is shown in Fig. 1.

All the strings were guessed in fewer than 10 attempts. The program achieves this by keeping a list of all its previous guesses to the human's code string, together with all the replies it received.

Correct answer

At every stage in the game it chooses a guess which is compatible with all the previous guesses, and therefore a candidate for the correct answer.

Since every guess reduces the number of possibilities, eventually the algorithm arrives at the correct string. If the program has received inconsistent replies to its guesses it will, at some stage, be unable to find any guess which is suitable; in that event it displays 'Error' and claims a victory.

The program reacts disturbingly quickly, taking at most a second or two to respond and then only at the end of a game when the number of possibilities is limited. A similar program written in an interpretative language, such as Basic, would be very slow; one published recently apparently took up to several minutes per move[3].

Could this program be improved so that it averaged

The possible answer at this stage are (0123', '0323', and '5263'. The program would select one of these at random but a better choise would be the string which distinguishes best among the remaining possibilities; in other words, which sorts them into as many reply groups as possible. Comparing each of these strings with every other we obtain the following table of replies:

(*See bottom of figure 1*)

Obviously '5263' is not a good choice as a guess, since you may need two more guesses before reaching the correct answer. At the start of the game there are far too many possibilities for an analysis of this depth, but near the end this modification to the strategy would lead to an improved average and it would be interesting to incorporate it in the program.

NOTES

1. Subroutines for the 6800 to calculate the numbers of Bulls and Cows between two codes using this method were published in Mircro-Bus, *Practical Electronics*, October 1977, p. 102.

2. I am grateful to David Deutsch for this algorithm.

3. A Basic program to play Mastermind appears in *Byte*, October 1977, pp. 168–171. A Pascal version appears in *Byte*, August 1978, pp. 168–176.

```
                    ;       VARIABLES IN PAGE O
                    ;
0000                FIRST  *=*+$DO             FOR LIST OF GUESSES
00D0                RAN    *=*+2 CURRENT RANDOM NUMBER
00D2                LRAN   *=*+2 LAST RANDOM NUMBER
00D4                TEMPA  *=*+1
00D5                NEXTIN =*    FOR DISBUF SUBROUTINE
00D5                BULLS  *=*+1 COUNTER FOR HITS
00D6                COUNT  =*    FOR DISBUF SUBROUTINE
00D6                COWS   *=*+1 COUNTER FOR CORRECT MISSES
00D7                SPECT  *=*+8 SPECTRUM OF GUESS
00DF                KEY    *=*+6 FOR NUMBER BEING TESTED
00E5                MYTRY  *=*+6 FOR MY NUMBER
00EB                TRY    *=*+2 ADDRESS OF NUMBER BEING COMPARED
00ED                PBUF   *=*+2 ADDRESS OF DISPLAY BUFFER
00EF                PLAST  *=*+2 ADDRESS OF END OF LIST OF GUESSES
00F1                WONU   *=*+1 YOU'VE WON FLAG
00F2                WONI   *=*+1 I'VE WON FLAG
                    ;
                    ;       EQUATES - REFER TO KIM MONITOR
                    ;
                    TEMP   =$00FC              USED BY CONVD2
                    PADD   =$1741              6530 A DATA DIRECTION
                    SBD    =$1742              6530 B DATA
                    AK     =$1EFE              DETERMINE IF KEY PRESSED
                    CONVD  =$1F48              CONVERT HEX TO SEGMENTS
                    CONVD2 =$1F4E              DISPLAY SEGMENT CODE
                    GETKEY =$1F6A              GET KEY - A = VALUE
                    ;
                    ;       START EXECUTION HERE
                    ;
00F3                       *=$0200
0200  4C 48 02      ENTER  JMP   START         JUMP ROUND SUBROUTINES
                    ;
                    ;       BULCOW
                    ;       ******
                    ;
                    ;       CALCULATES NUMBER OF BULLS (HITS) AND
                    ;       COWS (CORRECT MISSES) BETWEEN 2 4-DIGIT
                    ;       OCTAL NUMBERS AT (TRY) AND KEY, AND
                    ;       RETURNS THEM IN A AND X RESPECTIVELY.
                    ;
0203  A9 00         BULCOW LDA   £0
0205  A2 09                LDX   £9
0207  95 D5         CLEAR  STA   BULLS,X       ZERO SPECT COWS & BULLS
0209  CA                   DEX
020A  10 FB                BPL   CLEAR
020C  A0 03                LDY   £3            COMPARE 4 DIGITS
020E  B1 EB         BULL2  LDA   (TRY),Y       GET DIGIT
0210  D9 DF 00             CMP   KEY,Y         ARE THEY EQUAL?
0213  D0 04                BNE   NOBULL
0215  E6 D5                INC   BULLS         IF SO - COUNT A BULL
0217  10 11                BPL   NOCOWS        BRANCH ALWAYS
0219  AA            NOBULL TAX                 USE (TRY) DIGIT AS INDEX
021A  F6 D7                INC   SPECT,X       COUNT DIGITS
021C  F0 02                BEQ   COW
021E  10 02                BPL   NOCOW
0220  E6 D6         COW    INC   COWS          COUNT A COW
0222  B6 DF         NOCOW  LDX   KEY,Y         USE KEY DIGIT AS INDEX
0224  D6 D7                DEC   SPECT,X
0226  30 02                BMI   NOCOWS
0228  E6 D6                INC   COWS          COUNT A COW
022A  88            NOCOWS DEY                 POINT TO NEXT DIGITS
022B  10 E1                BPL   BULL2         NOT FINISHED
022D  A5 D5                LDA   BULLS         GET BULL COUNT
022F  A6 D6                LDX   COWS          GET COW COUNT
0231  60                   RTS
                    ;
                    ;       INCRAN
                    ;       ******
                    ;       INCREMENTS 2-BYTE NUMBER RAN.
                    ;
0232  E6 D1         INCRAN INC   RAN+1
0234  D0 08                BNE   NOCARY        NO CARRY
0236  E6 D0                INC   RAN           M. S. BYTE
0238  A5 D0                LDA   RAN
023A  29 0F                AND   £$0F          KNOCK OFF 4 BITS
023C  85 D0                STA   RAN
023E  60            NOCARY RTS
                    ;
                    ;       DISBUF
                    ;       ******
                    ;       DISPLAY AND KEYBOARD READ ROUTINE
                    ;       READS A DIGITS FROM KEYBOARD
                    ;       STORES THEM STARTING AT (PBUF)+X
                    ;       DISPLAYS 6 DIGITS STARTING AT (PBUF)
                    ;       CONTROL KEYS: 'DA' DISPLAYS ANSWER
                    ;                     '+'  CLEARS ENTRY
                    ;                     'GO' INTERPRETS ENTRY
                    ;
023F  AA            DISBUF TAX                 SET A=X
0240  85 D6         DISP   STA   COUNT         DIGIT COUNTER
0242  86 D5                STX   NEXTIN        POINTER FOR DIGITS
0244  20 32 02      DISP2  JSR   INCRAN        SPIN RANDOMIZER
0247  A9 7F                LDA   £$7F
0249  8D 41 17             STA   PADD          6530 A DATA DIRECTION
024C  A2 09                LDX   £9            FOR DISPLAY LINES
024E  A0 00                LDY   £0            FIRST DIGIT
0250  B1 ED         NEXT   LDA   (PBUF),Y      GET DIGIT
0252  30 05                BMI   DASH          IF MINUS DISPLAY SEGMENTS
0254  20 48 1F             JSR   CONVD         ELSE DISPLAY HEX DIGIT
0257  10 05                BPL   CONT          ALWAYS
0259  84 FC         DASH   STY   TEMP          FOR DISPLAY
025B  20 4E 1F             JSR   CONVD2        LIGHT SEGMENTS
025E  C8            CONT   INY                 NEXT DIGIT
025F  C0 06                CPY   £6            ALL DONE?
0261  D0 ED                BNE   NEXT
0263  8E 42 17             STX   SBD           6530 B DATA
0266  A9 00                LDA   £0
0268  8D 41 17             STA   PADD          ALL INPUTS
026B  20 FE 1E             JSR   AK            LOOK FOR KEY PRESS
026E  F0 D4                BEQ   DISP2         NOTHING SO LOOP
0270  20 6A 1F      WAIT   JSR   GETKEY        KEY VALUE
0273  C9 15                CMP   £$15          NO KEY?
0275  F0 05                BEQ   FINE          KEY HAS BEEN RELEASED
0277  85 D4                STA   TEMPA         KEY VALUE
0279  4C 70 02             JMP   WAIT          WAIT FOR RELEASE
```

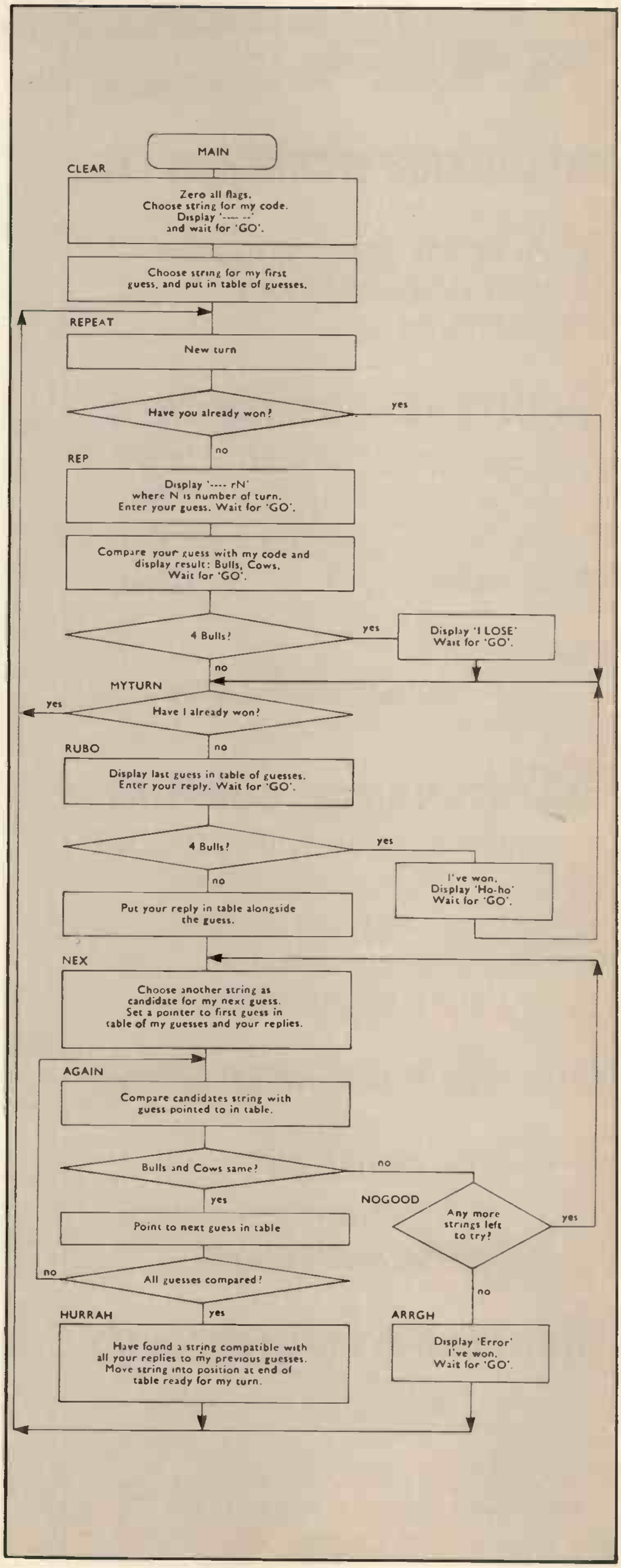
(*continued on next page*)

(continued from previous page)

```
027C  A5 D4     FINE   LDA   TEMPA
027E  C9 08            CMP   £$08        CONTROL KEY?
0280  10 0E            BPL   CONTRL
0282  A4 D5     DIGIT  LDY   NEXTIN      DIGIT KEY 7 OR LESS
0284  C4 D6            CPY   COUNT       ALL DIGITS IN?
0286  F0 BC            BEQ   DISP2       IF SO, IGNORE MORE
0288  91 ED            STA   (PBUF),Y    ELSE STORE IT
028A  C8               INY               POINT TO NEXT LOCATION
028B  84 D5            STY   NEXTIN
028D  4C 44 02         JMP   DISP2       REFRESH DISPLAY
0290  C9 11     CONTRL CMP   £$11        'DA' KEY
0292  30 B0            BMI   DISP2
0294  F0 0E            BEQ   DAKEY
0296  C9 12            CMP   £$12        '+' KEY
0298  F0 08            BEQ   RUBOUT
029A  A5 D5     GOKEY  LDA   NEXTIN      'GO' KEY
029C  C5 D6            CMP   COUNT       ALL IN YET?
029E  D0 A4            BNE   DISP2       BACK FOR MORE
02A0  18               CLC
02A1  60               RTS
02A2  38        RUBOUT SEC               CARRY MEANS RUBOUT
02A3  60               RTS
02A4  A9 E5     DAKEY  LDA   £MYTRY      MY NUMBER
02A6  85 ED            STA   PBUF        TO DISPLAY ANSWER
02A8  20 3F 02  START  JSR   DISBUF
02AB  4C E8 02         JMP   MAIN        FOR NEW GAME
                ;
                ;      CHOOSE
                ;      ******
                ;      UNPACKS 2-BYTE NUMBER IN RAN INTO
                ;      FOUR OCTAL DIGITS AND PUTS THEM
                ;      STARTING AT KEY.
                ;
02AE  A2 03     CHOOSE LDX   £3          DIGIT COUNTER
02B0  A5 D1            LDA   RAN+1
02B2  A4 D0            LDY   RAN
02B4  84 D4            STY   TEMPA
02B6  48        NEXT3  PHA
02B7  29 07            AND   £7          GET LOWER 3 BITS
02B9  95 DF            STA   KEY,X       STORE DIGIT
02BB  68               PLA
02BC  CA               DEX               POINT TO NEXT DIGIT
02BD  30 0C            BMI   RET         ALL DONE
02BF  46 D4            LSR   TEMPA       3-BIT SHIFT:
02C1  6A               ROR   A
02C2  46 D4            LSR   TEMPA
02C4  6A               ROR   A
02C5  46 D4            LSR   TEMPA
02C7  6A               ROR   A
02C8  4C B6 02         JMP   NEXT3       NEXT DIGIT
02CB  60        RET    RTS
                ;
                ;      MOVE
                ;      ****
                ;      MOVES 4 BYTES FROM KEY TO (PLAST).
                ;
02CC  20 AE 02  CMOVE  JSR   CHOOSE      CHOOSE AND MOVE
                ;
02CF  A0 03     MOVE   LDY   £3          BYTE COUNT
02D1  B9 DF 00  MOVE2  LDA   KEY,Y
02D4  91 EF            STA   (PLAST),Y
02D6  88               DEY
02D7  10 F8            BPL   MOVE2
02D9  60               RTS
                ;
                ;
                ;      WORD
                ;      ****
                ;      TRANSFERS MESSAGE OF 6 SEGMENT PATTERNS
                ;      FROM PROMPT+X TO DISPLAY BUFFER AT (PBUF).
                ;
02DA  A0 00     WORD   LDY   £0          START WITH OFFSET 0
02DC  BD BC 03  WORD2  LDA   PROMPT,X    GET LETTER
02DF  91 ED            STA   (PBUF),Y    TO DISPLAY BUFFER
02E1  E8               INX
02E2  C8               INY
02E3  C0 06            CPY   £6          ALL TRANSFERRED?
02E5  D0 F5            BNE   WORD2
02E7  60               RTS
                ;
                ;
                ;      MAIN PROGRAM
                ;      **** *******
                ;
02E8  A9 00     MAIN   LDA   £0
02EA  A2 0D            LDX   £WONI-MYTRY
02EC  95 E5     CLEAR  STA   MYTRY,X     ZERO FLAGS & ADDRESSES
02EE  CA               DEX
02EF  10 FB            BPL   CLEAR
02F1  9A               TXS               STACK POINTER = $FF
02F2  8D C3 03         STA   TURN
02F5  A9 80            LDA   £$80        BLANK
02F7  85 E9            STA   MYTRY+4
02F9  85 EA            STA   MYTRY+5
02FB  A9 E5            LDA   £MYTRY
02FD  85 EF            STA   PLAST       FOR MY NUMBER
02FF  20 CC 02  BEGIN  JSR   CMOVE       THINK OF A NUMBER
0302  A2 00            LDX   £PROMPT-DASHES  '---- --'
0304  20 DA 02         JSR   WORD        TRANSFER DASHES
0307  A9 00            LDA   £0
0309  20 3F 02         JSR   DISBUF      DISPLAY DASHES
030C  A9 00            LDA   £FIRST
030E  85 EF            STA   PLAST       START OF STORED REPLIES
0310  20 CC 02         JSR   CMOVE       PREPARE FIRST GUESS
                ;
                ;      START OF NEW ROUND
                :
0313  EE C3 03  REPEAT INC   TURN        NUMBER OF ROUND
0316  F0 D0            BEQ   MAIN        AVOID A LOOP
0318  A5 F1            LDA   WONU
031A  D0 2D            BNE   MYTURN      YOU'VE ALREADY WON
031C  A9 DF            LDA   £KEY
031E  85 ED            STA   PBUF        DISPLAY KEY
0320  A2 02     REP    LDX   £PROMPT-ROUND  '---- rN'
0322  20 DA 02         JSR   WORD
0325  A2 00            LDX   £0          ENTER GUESS FROM 0
```

(continued on next page)

Figure 2. *Simplified flowchart of the bulls and cows program. The labels in capital letters correspond to labels used in the assembler listing of the program.*


MAIN
CLEAR
Zero all flags. Choose string for my code. Display '---- --' and wait for 'GO'.
Choose string for my first guess, and put in table of guesses.
REPEAT
New turn
Have you already won?
yes
no
REP
Display '---- rN' where N is number of turn. Enter your guess. Wait for 'GO'.
Compare your guess with my code and display result: Bulls, Cows. Wait for 'GO'.
4 Bulls?
yes
Display 'I LOSE' Wait for 'GO'.
no
MYTURN
yes
Have I already won?
RUBO
no
Display last guess in table of guesses. Enter your reply. Wait for 'GO'.
4 Bulls?
yes
no
I've won. Display 'Ho-ho' Wait for 'GO'.
Put your reply in table alongside the guess.
NEX
Choose another string as candidate for my next guess. Set a pointer to first guess in table of my guesses and your replies.
AGAIN
Compare candidates string with guess pointed to in table.
Bulls and Cows same?
no
yes
NOGOOD
Any more strings left to try?
yes
Point to next guess in table
no
All guesses compared?
no
yes
HURRAH
ARRGH
Have found a string compatible with all your replies to my previous guesses. Move string into position at end of table ready for my turn.
Display 'Error' I've won. Wait for 'GO'.

(continued from previous page)

```
0327  A9 04             LDA   £4           ..UP TO 4
0329  20 40 02          JSR   DISP         DISPLAY & READ GUESS
032C  BO F2             BCS   REP          RUBOUT?
032E  A9 E5             LDA   £MYTRY
0330  85 EB             STA   TRY          TO BE COMPARED
0332  20 03 02          JSR   BULCOW
0335  85 E3             STA   KEY+4        BULLS
0337  86 E4             STX   KEY+5        COWS
0339  05 F2             ORA   WONI
033B  C9 04             CMP   £4           4 BULLS?
033D  DO 07             BNE   NOWIN        NO
033F  C6 F1       WIN   DEC   WONU         WONU = $FF
0341  A2 OE             LDX   £PROMPT-UWIN  'I LOSE'
0343  20 DA 02    WORDS JSR   WORD         DEFEAT
0346  20 3F 02    NOWIN JSR   DISBUF       DISPLAY IT
                  ;
                  ;     MY TURN TO GUESS
                  ;
0349  A5 F2       MYTURN LDA  WONI         HAVE I ALREADY WON?
034B  DO C6             BNE   REPEAT
034D  A5 EF             LDA   PLAST        MY GUESS
034F  85 ED             STA   PBUF         FOR DISPLAY
0351  AO 04       RUBO  LDY   £4           2 DIGITS, 4 TO 6
0353  A2 02             LDX   £PROMPT-ROUND  '--'
0355  20 DC 02          JSR   WORD2
0358  A2 04             LDX   £4           ENTER REPLY FROM 4
035A  A9 06             LDA   £6           ..UP TO 6
035C  20 40 02          JSR   DISP         DISPLAY & READ REPLY
035F  BO FO             BCS   RUBO         RUBOUT?
0361  A9 04             LDA   £4
0363  A8                TAY
0364  D1 ED             CMP   (PBUF),Y     4 BULLS?
0366  DO 07             BNE   NOWINS       NO
0368  A2 14             LDX   £PROMPT-IWIN  'Ho-ho'
036A  C6 F2             DEC   WONI         WONI = $FF
036C  4C 43 03          JMP   WORDS
036F  D8          NOWINS CLD               CLEAR DECIMAL MODE
0370  18                CLC
0371  A5 EF             LDA   PLAST
0373  69 06             ADC   £6           MAKE ROOM FOR REPLY
0375  85 EF             STA   PLAST
                  ;
                  ;     THINK UP NEXT GUESS
                  ;
0377  A5 DO             LDA   RAN
0379  85 D2             STA   LRAN
037B  A5 D1             LDA   RAN+1
037D  85 D3             STA   LRAN+1       SAVE RAN
037F  20 32 02    NEX   JSR   INCRAN
0382  20 AE 02          JSR   CHOOSE       NEW ATTEMPT
0365  A9 00             LDA   £FIRST
0387  85 EB             STA   TRY          FIRST REPLY
0389  20 03 02    AGAIN JSR   BULCOW       COMPARE
038C  AO 04             LDY   £4           BULLS
038E  D1 EB             CMP   (TRY),Y      SAME?
0390  DO 17             BNE   NOGOOD
0392  8A                TXA                GET COWS REPLY
0393  C8                INY
0394  D1 EB             CMP   (TRY),Y      SAME?
0396  DO 11             BNE   NOGOOD
0398  18                CLC                SO FAR SO GOOD
0399  A5 EB             LDA   TRY
039B  69 06             ADC   £6           POINT TO NEXT GUESS
039D  85 EB             STA   TRY
039F  C5 EF             CMP   PLAST        DONE THEM ALL?
03A1  DO E6             BNE   AGAIN        IF NOT
                  ;
                  ;     FOUND GUESS COMPATIBLE WITH ALL
                  ;     PREVIOUS REPLIES TO GUESSES.
                  ;
03A3  20 CF 02    HURRAH JSR  MOVE         PUT GUESS READY FOR TURN
03A6  4C 13 03          JMP   REPEAT       NEW ROUND
                  ;
                  ;     GUESS NO GOOD - TRY ANOTHER
                  ;
03A9  A5 DO       NOGOOD LDA  RAN
03AB  C5 D2             CMP   LRAN
03AD  DO DO             BNE   NEX
03AF  A5 D1             LDA   RAN+1
03B1  C5 D3             CMP   LRAN+1
03B3  DO CA             BNE   NEX          TRY ANOTHER
03B5  A2 08       ARRGH LDX   £PROMPT-ERROR  'ERROR'
03B7  C6 F2       DECI  DEC   WONI         I CLAIM A WIN
03B9  4C 43 03          JMP   WORDS        DISPLAY ERROR
                  ;
                  ;
                  ;     SEGMENT PATTERNS FOR MESSAGES
                  ;
03BC              PROMPT =*
03BC  CO          DASHES .BYTE $CO,$CO
03BD  CO
03BE  CO          ROUND  .BYTE $CO,$CO,$CO,$CO,$DO,$80
03BF  CO
03C0  CO
03C1  CO
03C2  DO
03C3  80
03C4  80          ERROR  .BYTE $80,$F9,$DO,$DO,$DC,$DO
03C5  F9
03C6  DO
03C7  DO
03C8  DC
03C9  DO
03CA  86          UWIN   .BYTE $86,$80,$B8,$BF,$ED,$F9
03CB  80
03CC  B8
03CD  BF
03CE  ED
03CF  F9
03D0  80          IWIN   .BYTE $80,$F6,$DC,$CO,$F4,$DC
03D1  F6
03D2  DC
03D3  CO
03D4  F4
03D5  DC

                        .END
```

THE TOTAL SOLUTION FROM

ALMARC

OF COURSE!

Now Almarc & Vector Graphic offer the complete solution to your computing needs for £2300.00* . . The Vector MZ needs only the addition of a V.D.U. and it's ready to go. Completely assembled and fully tested the Vector MZ Offers the following features as standard:-

* S-100 bus
* 4 MHz Z80A processor
* 158 instructions
* two quad density Micropolis floppies—over 630k bytes on line
* serial port
* two parallel ports
* 32K static ram
* 12K prom/ram board with extended monitor
* Extended disc Basic

Simply connect your peripherals (Elbit V.D.Us & Centronics printers are available from Almarc) and your up and running and, because the MZ uses the S-100 bus, you can plug in a massive range of add on units.

Ring or Write for a demonstration to:-

Almarc Data Systems Ltd.,
29 Chesterfield Drive,
Burton Joyce,
Nottingham. Tel: 0602 248565.

* Discount terms available.

● Circle No. 124

AIRAMCO LTD — MICROCOMPUTER PRODUCTS

Distributors for JADE COMPUTER PRODUCT5
All products brand new with full industrial specification

S100 COMPUTER CARDS KIT		ASSEMBLED	BARE BOARD
S100 Mother Board Kit 13 slots	£71	£82.50	£26.25
Jade 8080A CPU BOARD	£75.00	£122.50	£22.50
Jade Z80 CPU BOARD 2MHz	£95.00	£138.75	£26.25
Jade Z80 CPU BOARD 4MHz	£99.95	£149.95	£26.25
Jade Serial/Parallel I/O BOARD	£93.70	£134.29	£22.50
Jade 2708/2716 EPROM BOARD	£44.95	£82.46	£22.50
(16×2708 or 2716) WITH 8 2708s	£97.46	£134.96	£22.50
Problem Solvers 16K STATIC RAM 450nS		£270.00	—
16K STATIC RAM 250nS		£296.00	—
S.D. 5ales "EXPANDORAM" Dynamic			£19.95
Memory 375nS access time 8K	£133.25	—	
Memory 375nS access time 16K	£189.00		—
Memory 375nS access time 32K	£297.00	—	—
Memory 37SnS access time 64K	£513.00	—	—
VERSAFLOPPY DISK CONTROLLER (up to			—
4 drives 5¼ or 8"	£111.75	£165.00	—
SHUGART 8" Drive 801R	—	£385.00	—
PERTEC 5¼" Drive FP200	—	£225.00	—
KB756 Keyboard	£45.00	£56.00	—
S100 Desk Top Rack	£199.00	£230.00	£149.00

Components

2708 1024×8 EPROM	£6.99	4116 16K×1 DRAM	£14.99
2716 2048×8 EPROM	£29.90	8 for	£108.00
(intel)		6821P PIA	£5.90
TEXTOOL 24 pin		AY51013 UART	£4.50
ZERO FORCE SKIS	£5.60	AY51014 UART (5V)	£6.50
(equivalent of above) T.I.		AY53600 ENCODER	£9.99
1702 256×8 EPROM	£4.85	8080A CPU	£8.99
2104 4906×1 DRAM	£3.50	21L02-1 450nS	£1.20
8 for	£26.00	8 for	£7.92
21078-4 4096×1 DRAM	£3.50	21L02 250nS	£1.40
8 for	£26.00	8 for	£9.60
Note New Low Prices 16K Memory		2112-1 256×4 (450)	£2.25
		4044 4K×1 (450)	£7.45
		4045 1K×4 (450)	£8.25

All Prices EXCLUDE VAT @ 8%.
Trade discounts on Quantity
Please add £1.00 P&P for S100 items then add VAT @ 8%.

24-hr Ansaphone order service with ACCESS or BARCLAYCARD.
MAIL ORDER ONLY

For components please add 40 P&P, then add VAT @ 8%.

AIRAMCO LTD
30 WITCHES LINN
ARDROSSAN
AYRSHIRE
KA22 88R
TEL. 0294 65530
TELEX 779808

Semiconductor prices are always changing and the trend is generally downwards. So ring for latest up-to-date details.

● Circle No. 125

THE NEWBEAR COMPUTING STORE

The Bear announces its new store at:–
2 Gatley Road,
Cheadle,
Cheshire
Tel: 061-491 0134
callers welcome, mail order to Newbury.

PETITEVID VDU KIT £85

110 to 1,200 baud V24, 64 ch × 16 line scrolling, all on 8in. × 4in. PCB. Needs TV set, UHF Modulator and ASCII keyboard.

BOOKS		P&P
Computer Designs 77-68 a 6800 Microcomputer	£7·50	50
WB-1 a TTL Microcomputer	£6·50	50
Spare Diagram Set for WB-1	£1·00	50
Zilog Z-80 Technical Manual	£4·00	50
Z-80 PIO Manual	£2·50	50
Z-80 Programming Manual	£4·00	75
Motorola: Understanding Microprocessors	£2·75	30
M6800 Microprocessors Programming Manual	£4·50	50
M6800 Applications Manual	£9·50	1 00
6800 Assembly Language Programming	£6·95	50
Understanding Microcomputers	£7·56	50
Z-80 Programming for Logic Design	£5·95	50
How to Buy and Use Microcomputers	£6·95	75
Microcomputer Primer	£5·55	75
Adam Osbourne Introduction to Microcomputers		
Vol. 0 Beginners Book	£5·25	50
Vol. 1 Basic Concepts	£5·95	50
Vol. 2 Some Real Products	£11·95	1 00
8080 Programming for Logic Design	£5·95	50
6800 Programming for Logic Design	£5·95	50
Some Common Basic Programs	£8·95	50
Payroll with Cost Accounting in Basic	£9 95	1.00
Sybex Microprocessors	£8·00	50
Microprocessor Interfacing Techniques C20	£8 00	50
Scelbi 8080 Software Gourmet Guide Cook Book	£7·95	50
6800 Software Gourmet Guide Cook Book	£7·95	50
The Scelbi Byte Primer	£9·95	1·00
What to do after you hit return	£7·00	75
PCC Reference Book on Home Computers	£4·95	50
Instant Basic	£4·95	75
My Computer Likes Me	£1·65	30
Games with a Pocket Calculator	£1·75	30
Games Tricks & Puzzles for a Hand Calculator	£2·45	30
Best of Creative Computing Vol. 1	£6·95	75
Best of Creative Computing Vol. 2	£6·95	75
Hobby Computers are Here	£3·99	50
Best of Byte	£8·95	75
101 Basic Computer Games	£5·50	75
The Home Computer Revolution	£3·45	50
Computer Lib	£5·95	50
First Book of Kim	£7·00	50
Dr. Dobbs Journal (Vol. 1)	£10 00	75
Z-80 Microcomputer Handbook	£7·25	50

Hardware Components Section

MEMORIES

2102-1	£1·25	4027 (250ns)	£4·50
2102L-1	£1·35	2114 (450ns)	£8·25
2112	£3·04	4116 (250ns)	£15·00
2513	£8·50	2708 (450ns)	£6·99
		2716 (J1)	£22·50

SPECIAL MOTOROLA MICROCOMPUTING ICs

MC6800P	£9·26	SWATBUG	£16 00
MC6820P	£4·63	MC6830L7	£11·33
MC6850P	£8·00	MC8602P	£2 88
MC6810AP	£3·61	MC14538P	£3·69
		MC3459	£2·00

ZILOG MICROPROCESSORS

Z-80 CPU 2·5MHz	£15·50	Z-80A CPU	£20 50
Z-80 P10 PS	£10 00	Z-80A P10	£11 55
Z-80 CTC PS	£10 00	Z-80A CTC	£11·55

MICROPROCESSORS

SC/MP Mk II £10·30 6502 £14.93 8080 £6 00

INTERFACING ICs

		VDU	BUFFERS	
SFF96364	£16·20			
MC1488P	£1 40		81LS95	£1·30
MC1489P	£1·40		81LS96	£1·30
75150P	£1·30	For V24	81LS97	£1·30
75150N	£1·20	RS232C	81LS98	£1·30
75154	£2·50		8726	£1·84
4N33	£1·95		8T85	£1·60
AY-5-1013	£5·40		8T97	£1·60
8402	£4·50		74367	£1·30

8080A Bugbook	£6·95	75
TTL Cookbook	£6·30	50
How to Program Microcomputers (Barden)	£6·30	50
Microprocessors Systems	£5·70	50
Take a Chance with your Calculator	£4·95	50
Home computers Questions and Answers		
Vol. 1 Hardware	£5·95	50
Vol. 2 Software	£5·95	50
Data Acquisition Handbook	£2·50	50
SC/MP Applications Manual	£3 00	50
Getting Involved with your own Computer	£4·75	50
Your Home Computer	£4·95	50
Using the 6800 Microprocessor	£6 25	50
Basic Programming in Real Time	£5 95	50
IC Timer Cookbook	£6·95	75
Active Filter Cookbook	£10·45	75
IC OP-AMP Cookbook	£9·05	75
Kim-1 User Manual	£7 00	75
6500 Programming Manual	£7 00	75
6500 Hardware Manual	£7 00	75
The New Hobby Computers	£3 95	30
Artist and Computer	£3·95	30
Computer programs that work	£2·40	30

SYM-1
(formally VIM-1)

THE NEW 6502 BASED MICRO FROM SYNERTEK. FULLY ASSEMBLED AND TESTED £199·00 PLUS 8% VAT. CARRIAGE £1 00. SEND FOR DATA SHEET.

BEAR BAGS (Kits)

1	77-68 CPU P.C.B. Components and Edge Connectors	£49=50
2	77-78 LED's and Switches	£14·95
3	77-68 Power Supply *	£17·95
4	77-68 19" Bu Rack and Backplane *	£27·70
5	77-68 4K RAM P.C.B. and Components	£75·00
6	77-68 Mon 1 P.C.B. and Components	£50·70
7	4K Ram Exorcisor P.C.B. and Components	£71·50
8	8K Ram Exorcisor P.C.B. and Components *	£160·00
9	Petitevid V.D.U. Kit	£85·00
10	Kansas City Cassette Interface	£18 95
11	U.H.F. Modulator	£4·50
12	77-68 VDU P.C.B. and Components	£69·50
13	77-68 Mon 2 P.C.B. and Components	t.b.a.
14	Prom Programmer P.C.B. and Components	£35 00
15	Promverter	£8 50

Plus VAT 8%
50p Postage & Packing
* £1·00 Postage and Packing

PCB and COMPONENTS
77·66 PRINTED CIRCUIT BOARDS

77-68 CPU P.C.B.	£10 00
77-68 4K Ram P.C.B.	£10 00
77-68 Mon 1 P.C.B.	£10 00
77-68 V.D.U. P.C.B.	£10 00
S50 bus 4K Ram P.C.B.	£12 00
Exorcisor Compatible Motherboards (8 slot)	£23·50
4K Ram P.C.B.	£10 00
Veroboard Backplane 8" × 17"	£4·30
77-68 Extender card	£4·10
ASCII Keyboard assembled and tested	£86 00
Keyboard Kit	£56 00
Keyboard Case	£20 33
U.V. Prom Eraser	£56·00
S100 bus 9 slot motherboard	£19·50

Goods are normally shipped within 24 hours, subject to availability. Barclaycard & Access. VAT at 8% for Hardware Components. 30p postage and packing unless otherwise stated. Cheques to be made out to The Newbear Computing Store. Send for an up-to-date catalogue to The Newbear Computing Store, 7 Bone Lane, Newbury 0635-49223. Callers are welcome Monday to Saturday 9.00a.m.-5.30p.m., but please ring to let us know when you are calling.

● Circle No. 126

ILLUSTRATING BASIC

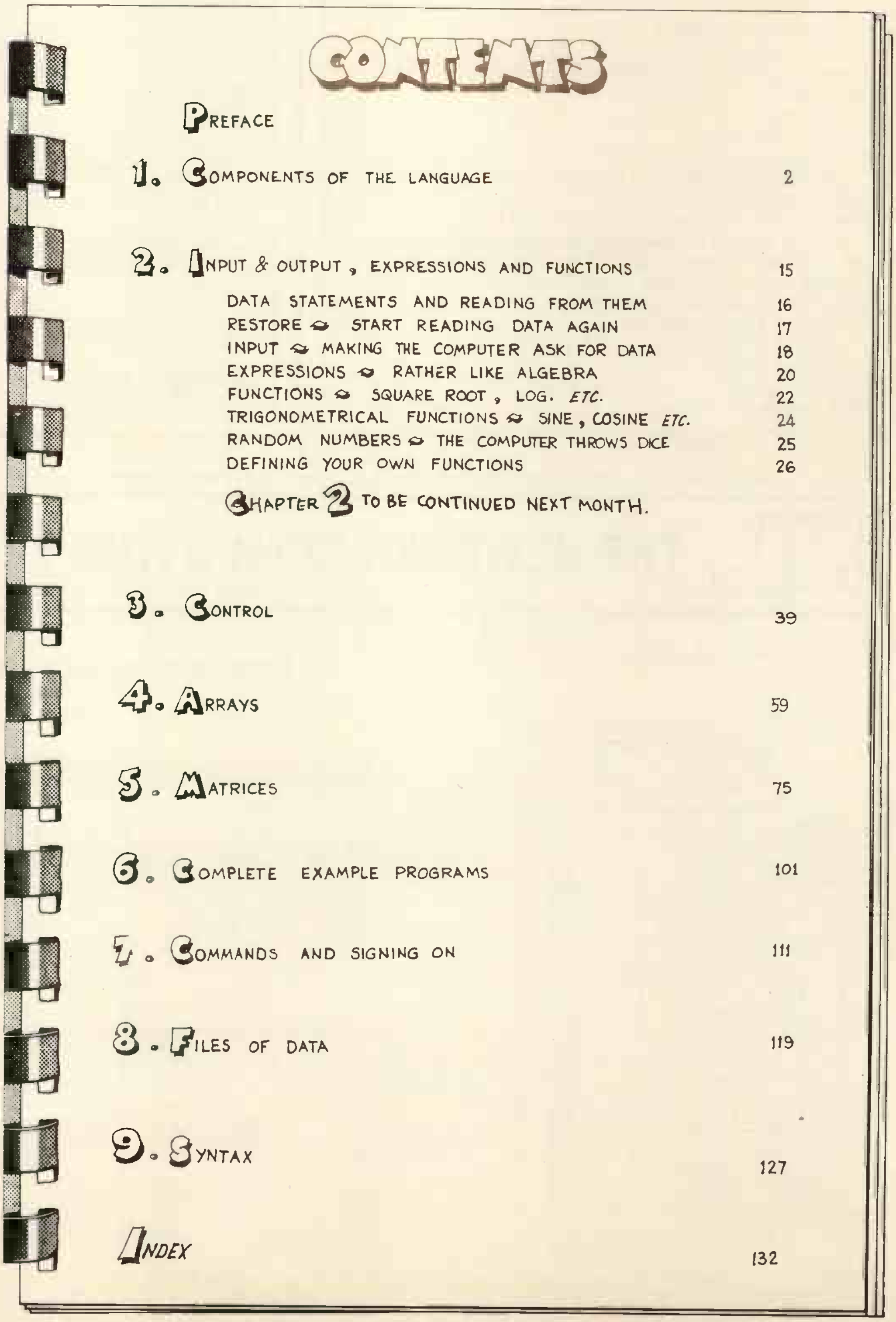
(A SIMPLE PROGRAMMING LANGUAGE)

WE CONTINUE our series of articles on how to program in Basic, probably the most widely-used programming language for small computers.
For the series, which began last month, we have obtained the serialisation rights for one of the best books on the subject, *Illustrating Basic* by Donald Alcock.

Each month, we are publishing a part of the book, so by the end of the series you will have the complete book.
It is written with a distinct informality and has a rather unusual presentation; but it is this style, we believe, which makes it one of the most easy to read tutorials.

Alcock *Illustrating Basic.* Chapter 2. c Cambridge University Press. Reprinted by permission.

Copies of *Illustrating Basic* can be obtained from *Practical Computing.* See enquiry card in this issue for details.



CONTENTS

CHAPTER 2 TO BE CONTINUED NEXT MONTH.

2 INPUT & OUTPUT, EXPRESSIONS AND FUNCTIONS

DATA+READ

THE DATA STATEMENT PROVIDES RAW NUMBERS AND TEXTS FOR READ STATEMENTS TO FEED ON. YOU MAY HAVE ANY NUMBER OF DATA STATEMENTS ANYWHERE IN A PROGRAM. BEFORE STARTING WORK THE PROGRAM JOINS THEM TOGETHER TO FORM ONE LONG "QUEUE". THERE IS NOTHING TO *OBEY* IN A DATA STATEMENT, BUT AS THE COMPUTER OBEYS A READ STATEMENT IT PICKS UP AS MANY ITEMS AS IT NEEDS FROM THE FRONT OF THIS QUEUE LEAVING THE REMAINDER FOR SUBSEQUENT READ STATEMENTS.

THE FOLLOWING TWO PROGRAMS BOTH PRODUCE THE RESULT SHOWN AT THE END OF THE SECOND PROGRAM.

```
10 REM      UNBROKEN  QUEUE
20 DATA   "ODDS", 7, -23, -17, "EVENS", -12, 36
30 READ     Q$,   B,   C,  D,    E$,   E,  F
40 PRINT    Q$;   B;   C;  D;    E$;   E;  F
50 END
```

```
10   REM      FRAGMENTED QUEUE
20   DATA   "ODDS", 7
30   READ  Q$,B,C,D,E$,E,F
40   DATA    -23, -17
50   PRINT  Q$;B;C;D;E$;E;F
60   DATA   "EVENS", -12, 36
70   END
RUN
ODDS 7  -23  -17 EVENS-12   36
```

IT IS UP TO YOU TO PUT NUMBERS AND TEXTS WHERE THEY WILL BE READ BY VARIABLES OF THE RIGHT KIND. THE FOLLOWING PROGRAM WOULD FAIL WHILE READING N. (ONE OR OTHER PAIR SHOULD BE REVERSED.)

```
10   DATA     "ANNO", 1977
20   READ      N, ✱  C$
30   PRINT     N;    C$
40   END
```

SOME *BASICS*, HOWEVER, CREATE SEPARATE QUEUES FOR NUMBERS AND TEXTS ☞ THIS EXAMPLE WOULD WORK. BUT SEPARATE QUEUES ARE UNUSUAL.

IF THERE ARE MORE "READS" THAN ITEMS THE COMPUTER PRINTS A MESSAGE SAYING IT HAS RUN OUT OF DATA AND STOPS WORK.

```
10 DATA  1.5, 2.5, -3.5, 4
20 READ  A, B
30 READ  C, D, E
40 PRINT  A;B;C;D;E
50 END
RUN
* OUT  OF DATA AT LINE  30 *
```

RESTORE

THE RESTORE STATEMENT BRINGS YOU BACK TO THE FRONT OF THE QUEUE OF ITEMS FORMED FROM ALL THE DATA STATEMENTS.

WHEN THE COMPUTER OBEYS A **READ** STATEMENT IT READS AS MANY ITEMS AS IT NEEDS FROM THE FRONT OF THE QUEUE AND LEAVES THE REMAINDER FOR SUBSEQUENT **READ** STATEMENTS. BUT IT DOESN'T *DESTROY* THE QUEUE. INDEED ON MEETING A **RESTORE** STATEMENT THE COMPUTER JUMPS BACK TO THE VERY FRONT OF THE QUEUE AND OBEYS THE NEXT **READ** STATEMENT FROM THERE.

```
10  DATA  1,2,3,4,5,6,7,8,9
20  READ  A,B,C
30  PRINT  A;B;C
40  READ  A,B,C
50  PRINT  A;B;C
60  RESTORE
70  READ  A,B,C
80  PRINT  A;B;C
90  END
RUN
1  2  3
4  5  6
1  2  3
```

THESE NEVER GET READ AT ALL

N.C.C. "STANDARD BASIC" USES THE WORD "RESET"

ITEMS

ITEMS IN DATA STATEMENTS MAY BE *NUMBERS* (WITH OR WITHOUT + & − SIGNS), OR *TEXTS*, OR BOTH; BUT NOT *EXPRESSIONS* (SEE PAGE 20).

```
10  REM   NUMBERS AND TEXTS ALLOWED
20  DATA  1.5, +16.28, -13.47, "YES", "NO"
30  REM   EXPRESSIONS  FORBIDDEN
40  DATA      3.14*6.5↑2/4
```

ITEMS MUST BE SEPARATED FROM EACH OTHER BY COMMAS. THERE IS NO COMMA AT THE END OF EACH LIST.

IF A TEXT IS TOO LONG FOR THE VARIABLE THEN THE EXTRA CHARACTERS ARE LOST:

```
10  DATA  "FAR TOO LONG FOR THE VARIABLE"
20  READ  V$
```

THE LAST CHARACTER STORED IN V$

SEE PAGE 13

IN THE ORIGINAL *DARTMOUTH BASIC* THE ONLY WAY TO GET DATA INTO A PROGRAM WAS BY THE **DATA** STATEMENT. MOST *BASICS* TODAY HAVE THE MORE POPULAR **INPUT** STATEMENT AS WELL. THE **INPUT** STATEMENT IS EXPLAINED OVERLEAF.

INPUT

YOU CAN MAKE A *BASIC* PROGRAM ASK FOR DATA: IT RESTS IN SUSPENSE WHILST YOU TYPE A LINE.

```
10    REM     ILLUSTRATING INPUT
20    PRINT   "TYPE ME A NUMBER"
30    INPUT    X
40    PRINT   "YOU JUST TYPED"; X
50    END
RUN
TYPE ME A NUMBER
?
```

THE PROGRAM IS RESTING AT STATEMENT 30 WAITING FOR DATA. IT WILL WAIT INDEFINITELY UNTIL YOU TYPE SOME DATA AND PRESS THE RETURN KEY. (SOME *BASICS* MAY SHUT DOWN AFTER A WHILE IF THEY GET NO RESPONSE.)

```
? 6.5
YOU JUST TYPED 6.5
```

THE INPUT STATEMENT MAY NEED MORE THAN ONE NUMBER:

```
30 INPUT X, Y, Z         ⇨   ? 6.5, 7.5, 8.5
```

IT CAN INPUT TEXTS AS WELL:

```
40 INPUT  A$, B$         ⇨   ? "REVELATIONS", "THE APOCRYPHA"
```

AND A MIXTURE OF NUMBERS & TEXTS:

```
30 INPUT  T$, C, V       ⇨   ?  "ROMANS", 15 , 4.2
```

BUT WHATEVER THE INPUT STATEMENT NEEDS, THE MANNER OF ASKING (CALLED THE PROMPT) IS THE SAME: IN SOME *BASICS* JUST A RING ON THE BELL; IN OTHERS A QUESTION MARK; IN OTHERS AN ARROW OR "GREATER THAN" SIGN.

```
♫                  ?                  >
```

SO TO HELP THE USER OF A PROGRAM (WHO MAY NOT BE ITS AUTHOR) YOU SHOULD PRECEDE EVERY INPUT STATEMENT WITH AN EXPLANATORY PRINT STATEMENT:

```
25    PRINT  "TYPE A TEXT & A NUMBER"
30    INPUT   T$, C
```

[WHEN VARIABLES HAVE SUBSCRIPTS (SEE PAGE 64) YOU MAY INPUT & USE THEIR VALUES FROM LEFT TO RIGHT: *e.g.* INPUT I,J,A(I,J) .]

WHAT HAPPENS IF YOU TYPE SOMETHING WRONG IN RESPONSE TO A PROMPT?
UNFORTUNATELY THERE ARE DIFFERENT ACTIONS BY DIFFERENT *BASICS*.

TOO MUCH DATA: YOU ARE ASKED TO RETYPE THE WHOLE LINE.

```
10 INPUT A,B,C
```

```
? 10, 20, 30, 40, 50 ✱
TOO MUCH DATA; RETYPE LINE
?
```

BUT SOME *BASICS* JUST *IGNORE* THE EXTRA NUMBERS; OTHERS *SAVE* THEM FOR THE NEXT INPUT STATEMENT ⇄ OR DO SO ONLY IF THE LINE OF DATA ENDS WITH A COMMA.

NOT ENOUGH DATA: YOU ARE ASKED TO RETYPE THE WHOLE LINE.

```
20 INPUT A,B,C,D,E
```

```
? 10 ✱
NOT ENOUGH DATA; RETYPE LINE
?
```

BUT SOME *BASICS* SIMPLY CONTINUE PROMPTING UNTIL THEY HAVE ENOUGH DATA.

MISMATCHING: YOU ARE ASKED TO RETYPE THE WHOLE LINE.

```
30 INPUT T$,C,V
```

```
? "ROMANS", 15, "SIX"
WRONG KIND OF DATA; RETYPE LINE
?
```

BUT SOME *BASICS* WOULD ACCEPT THE FIRST TWO ITEMS ABOVE AND RE-PROMPT ONLY FOR THE THIRD; SOME *BASICS* WOULD STOP THE PROGRAM ALTOGETHER.

TEXTS: IF A TEXT HAS NO SPACES IN IT THEN YOU ARE ALLOWED TO LEAVE OUT THE QUOTATION MARKS:

```
40 INPUT T$, C
```

```
? ROMANS, 15
```

BUT SOME *BASICS* DO NOT ALLOW THIS; SOME WOULD EVEN STOP THE RUN.
IF A TEXT IS TOO LONG FOR THE VARIABLE THEN YOU ARE ASKED TO RETYPE THE WHOLE LINE:

```
50 INPUT T$, C
```

```
? "THE GOSPEL ACCORDING TO ST.JOHN",2 ✱
TEXT TOO LONG; RETYPE LINE
?
```

BUT SOME *BASICS* WOULD SIMPLY TRUNCATE THE OVER-LONG TEXT.
SOME *BASICS* ALLOW TEXTS ENCLOSED WITHIN APOSTROPHES; SEE PAGE 12.

COMMAS & SPACES: THE TYPED LIST SHOULD HAVE COMMAS BETWEEN ITEMS AND NO SPACES INSIDE ITEMS. SOME *BASICS* PERMIT COMMAS TO BE OMITTED BUT INSIST ON PROPER SPACING (SEE PAGE 6); OTHERS DISREGARD SPACES (EXCEPT IN TEXTS) BUT INSIST ON THE COMMAS.

CHAPTER 2

EXPRESSIONS

YOU CAN WRITE EXPRESSIONS OF GREAT COMPLEXITY IN *BASIC*.

HERE IS AN EXAMPLE OF AN EXPRESSION ⇒ THE PAINTED AREA OF THE WATER TANK ON PAGE 2.

3.14 * D↑2/4 + 3.14 *D *H

GIVEN D=6.5, H=27, IF YOU WERE ASKED TO EVALUATE THIS EXPRESSION YOU WOULD CERTAINLY *NOT* START BY WORKING OUT 4+3.14=7.14 NOR 2/4 = 0.5 THEN 0.5+3.14 = 3.64
WHY NOT ? BECAUSE OF CONVENTIONS WE ALL ACCEPT:

☆ EXPONENTIATION COMES FIRST : DO 6.5^2 = 42.25

☆ THEN MULTIPLICATIONS & DIVISIONS : DO 3.14×42.25 = 132.665
THEN 132.665 ÷ 4 = 33.1663
THEN 3.14 ×6.5 = 20.4100
THEN 20.4100×27 = 551.07

☆ FINALLY ADDITIONS & SUBTRACTIONS : DO 33.1663+551.07=584.236

AND THAT IS PRECISELY HOW BASIC DOES IT.

HERE IS THE SAME FORMULA EXPRESSED DIFFERENTLY:

(3.14 * D * (D/4 +H))

YOU MAY "NEST" BRACKETS TO ANY REASONABLE DEPTH. THE COMPUTER EVALUATES THE EXPRESSION IN THE INNERMOST PAIR OF BRACKETS AND WORKS OUTWARDS ⇒ PRODUCING THE RESULT YOU WOULD EXPECT BY THE RULES OF ALGEBRA.

THERE IS NO IMPLIED MULTIPLICATION ; YOU CAN'T DO THIS:

3.14 D(D/4+H)

AND YOU MAY NOT PUT TWO OF THESE THINGS ⇒ ↑ / * − +
NEXT TO ONE ANOTHER WITHOUT A BRACKET INTERVENING:

3.14* +D* (+D/+4 +H)

O.K. A BRACKET INTERVENES

HOW DO YOU EXPRESS 10^{-2} ? NOT AS 10↑-2, BUT AS 10↑(-2) OR 1/(10↑2).

ON TYPING EXPRESSIONS YOU MAY SPACE OUT OR BUNCH UP AS YOU PLEASE:

```
10 LET   A   =  3.14  *  D↑2 / 4  +3.14*D*H
```

RULES FORBIDDING SPACES IN KEYWORDS & LINE NUMBERS ARE GIVEN ON PAGE 6.

CLARIFY EXPRESSIONS BY USING BRACKETS EVEN WHEN NOT STRICTLY NECESSARY.

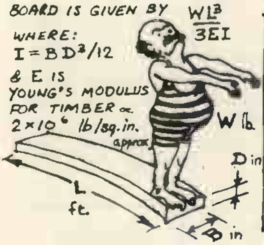
WHAT IS A/B/C ? MOST *BASICS* TREAT IT AS (A/B)/C BUT SOME AS A/(B/C). (TRY 8/4/2 BOTH WAYS AND SEE THE DIFFERENCE.) ALWAYS USE BRACKETS TO MAKE SURE THE COMPUTER EVALUATES AN AMBIGUOUS EXPRESSION THE WAY YOU INTEND.

A↑B↑C IS ANOTHER AMBIGUOUS EXPRESSION: (A↑B)↑C OR A↑(B↑C) ? AND A^{2M} SHOULD BE EXPRESSED A↑(2*M), NOT A↑2*M.

HERE ARE TWO PROGRAMS ILLUSTRATING EXPRESSIONS.

THE DEFLECTION OF A DIVING BOARD IS GIVEN BY $\frac{WL^3}{3EI}$

WHERE:
I = BD³/12
& E IS YOUNG'S MODULUS FOR TIMBER ≈ 2×10⁶ lb/sq.in.



```
10  PRINT "TYPE: W(LB), L(FT), B(IN), D(IN)"
20  INPUT          W,    L,     B,    D
30  LET   E = 2000000
40  LET   I = B * D↑3/12
50  LET   V = W*(L*12)↑3/(3*E*I)
60  PRINT "BOARD SAGS"; V; "INCHES"
70  END
```

```
10  PRINT "TYPE:  SUM , YEARS, INTEREST%"
20  INPUT            S,      N,      P
30  LET    R = P/100
40  LET    M = S*R*(1+R)↑N/(12*((1+R)↑N-1))
50  PRINT  "MONTHLY BURDEN = $" ; M
60  PRINT  "YOU LOSE $"; M*(12*N)-S
70  END
```

THE MONTHLY REPAYMENT

ON A MORTGAGE LOAN OF $S

AT P%

OVER N YEARS IS:

$$\frac{SR(1+R)^N}{12[(1+R)^N-1]}$$

WHERE: R = P÷100.

SOME *POINTS TO WATCH* :

IF YOU WRITE "LET A = B/C" AND "C" TURNS OUT TO BE ZERO (OR SOMETHING VERY VERY SMALL) THE COMPUTER PRINTS A MESSAGE TO SAY "DIVISION BY ZERO" (OR "NUMBER TOO BIG"). IN SUCH CASES SOME *BASICS* THEN SUPPLY A RESULT FOR "A"; THE RESULT BEING ±10^{38} (OR BIGGER IF THE COMPUTER CAN STORE IT), THE *SIGN* BEING THAT OF THE NUMERATOR OF THE EXPRESSION CAUSING THE TROUBLE. ON THE OTHER HAND SOME *BASICS* STOP THE PROGRAM. OTHER *BASICS* SUPPLY THE BIG VALUE AS DESCRIBED ABOVE BUT DON'T REPORT THE TROUBLE.

–A↑B IN MOST *BASICS* IS TREATED AS –(A↑B). BUT IT IS BEST TO INCLUDE THE BRACKETS TO MAKE SURE.

A↑B IS TAKEN AS 1.0 IF B=0 AND A>0. THE PROGRAM STOPS IF A<0 ALTHOUGH SOME *BASICS* ALLOW IT WHEN B IS A WHOLE NUMBER.

A↑B GIVES 10^{38} (SEE ABOVE) IF A=0 AND B<0.

FUNCTIONS

IN ADDITION TO THE OPERATORS * / + − ↑ *BASIC* PROVIDES *FUNCTIONS* SUCH AS *SQUARE ROOT*.

YOU HAVE ALREADY SEEN AN EXAMPLE OF A FUNCTION ON PAGE 3:

```
8  LET   R = INT(G+1)
```

WHERE R ENDS UP AS THE INTEGRAL PART OF THE EXPRESSION INSIDE THE BRACKETS. THE EXPRESSION G+1 COULD BE MORE COMPLICATED: HERE IS THE WATER TANK PROGRAM AGAIN:

```
10  PRINT  "TYPE: DIAMETER, HEIGHT, COVERAGE"
20  INPUT          D,        H,        C
30  LET   T = INT(1+(3.14*D*(D/4+H))/C)
40  PRINT  "YOU NEED"; T; "POTS"
50  END
```

THE EXPRESSION MAY ITSELF CONTAIN FUNCTIONS: EVEN THE SAME ONE:

```
10  INPUT  X
20  PRINT  SQR(SQR(ABS(X)))
30  END
```

WHICH IS A PROGRAM TO PRINT THE FOURTH ROOT OF THE POSITIVE VALUE OF A NUMBER.

A FUNCTION IN AN EXPRESSION MAY BE TREATED IN EXACTLY THE SAME WAY AS A NUMERICAL VARIABLE ≈ WHEREVER YOU MAY TYPE X YOU MAY ALSO TYPE INT(X). YOU SHOULD NOT TYPE ANY SPACES IN THE NAME OF THE FUNCTION NOR BETWEEN THE NAME AND OPENING BRACKET: HOWEVER, SOME *BASICS* DO NOT OBJECT TO SUCH SPACES. THE EXPRESSION INSIDE THE BRACKETS MAY BE SPACED OUT OR BUNCHED UP.

MANY *BASICS* OFFER A GREAT VARIETY OF FUNCTIONS BUT ALL SHOULD OFFER AT LEAST THE STANDARD ELEVEN DESCRIBED HERE. THEY ARE CALLED *INTRINSIC* FUNCTIONS BECAUSE THEY ARE PART OF *BASIC* ITSELF. YOU MAY INVENT OTHER FUNCTIONS WHICH ARE NOT INTRINSIC AS EXPLAINED ON PAGE 26.

FUNCTIONS SGN(X), ABS(X), SQR(X), INT(X), LOG(X), EXP(X) ARE DESCRIBED OPPOSITE: THE TRIGONOMETRICAL FUNCTIONS SIN(X), COS(X), TAN(X) ARE DESCRIBED ON PAGE 24; PAGE 25 IS DEVOTED TO THE FUNCTION RND.

SGN(X)

= +1 IF $X > 0$
= 0 IF $X = 0$
= −1 IF $X < 0$

"THE SIGN OF"

```
10 PRINT SGN(7.2); SGN(0); SGN(-.2)
```

```
1 0 -1
```

ABS(X)

= +X IF $X \geqslant 0$
= −X IF $X < 0$

"THE ABSOLUTE VALUE OF"

```
20 PRINT ABS(7.2); ABS(0); ABS(-.2)
```

```
7.2  0  0.2
```

SQR(X)

= $\sqrt{X}$ IF $X \geqslant 0$
= ERROR IF $X < 0$ (BUT SOME BASICS PRODUCE $\sqrt{-X}$)

"THE SQUARE ROOT OF"

```
30 PRINT SQR(16); SQR(0); SQR(-16)
```

```
4  0
ERROR: SQR(NEGATIVE)
```

INT(X)

"THE INTEGRAL PART OF"
(THE HIGHEST INTEGER LESS THAN OR EQUAL TO X)
(WHERE 0 IS "HIGHER" THAN −1)

```
40 PRINT INT(3.4);INT(-3.4);INT(3)
```

```
3  -4  3
```

N.B.

LOG(X)

= $\log_e(X)$ IF $X > 0$
= ERROR IF $X = 0$
= ERROR IF $X < 0$ (BUT SOME BASICS PRODUCE $\log_e(-X)$)

"THE NATURAL LOGARITHM OF"

```
50 PRINT LOG(1); LOG(10); LOG(2.71828)
```

```
0  2.30258  1
```

EXP(X)

= ε^X
(where $\varepsilon \doteq 2.71828...$)

"THE NATURAL ANTILOGARITHM OF"
"THE EXPONENTIAL OF"

```
60 PRINT EXP(0); EXP(2.30258);EXP(1)
```

```
1  10  2.71828
```

HERE IS A LITTLE PROGRAM TO ILLUSTRATE SGN, ABS & INT.

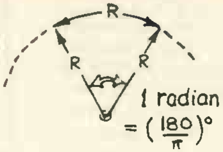
```
10 PRINT "TYPE A SUM OF MONEY; + OR- "
20 INPUT L
30 LET S=INT(ABS(L)*100+.5)/100
40 LET D=INT(S)*SGN(L)
50 LET C=INT((S-ABS(D))*100)
60 PRINT D;"DOLLARS AND";C;"CENTS"
70 END
RUN
TYPE A SUM OF MONEY;+OR-
? -123.456
-123 DOLLARS AND 46 CENTS
```

ABS(L) IS POSITIVE WHATEVER THE SIGN OF L; *100 CONVERTS TO CENTS; +.5 ADDS HALF A CENT; INT( ) ROUNDS TO NEAREST CENT; /100 CONVERTS THE FORM TO d.cc FROM THE ORIGINAL ±d.cccc....

D IS THE DOLLARS WITH THE SIGN RESTORED

C IS THE CENTS & IS ALWAYS POSITIVE

TRIG. FUNCTIONS



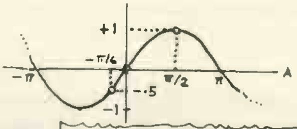
ALL ANGLES ARE MEASURED IN *RADIANS*.

IN THE EXAMPLES BELOW P IS SET EQUAL TO π.

```
70 LET P= 3.141592
```

SIN(A)

"THE SINE OF"

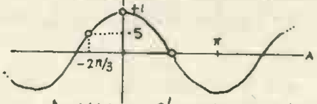


```
80 PRINT SIN(-P/6); SIN(0); SIN(P/2)
```

```
-.5  0  1
```

COS(A)

"THE COSINE OF"

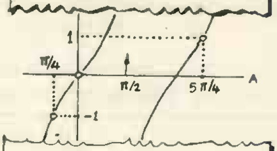


```
90 PRINT COS(-2*P/3); COS(0); COS(P/2)
```

```
.5  1  0
```

TAN(A)

"THE TANGENT OF"



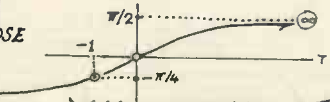
```
100 PRINT TAN(-P/4); TAN(0); TAN(5*P/4)
110 PRINT TAN(P/2)
```

$\tan(\frac{\pi}{2}+n\pi)=\infty$

```
-1  0  1
ERROR: TANGENT TOO BIG
```

ATN(T)

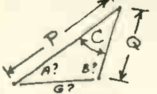
"THE ANGLE WHOSE TANGENT IS" *(or $\tan^{-1}(T)$)*



```
120 PRINT ATN(1E38); ATN(0); ATN(-1)
```

```
1.57079  0  -.78539
```

THE FOLLOWING EXAMPLE PROGRAM USES TRIGONOMETRICAL FUNCTIONS:



GIVEN P, Q & ANGLE C

- ★ *THE AREA IS:* $\frac{1}{2}PQ \sin C$
- ★ *ANGLE A IS:* $\tan^{-1}(Q\sin C/(P-Q\cos C))$
- ★ *ANGLE B IS:* $\tan^{-1}(P\sin C/(Q-P\cos C))$
- ★ *SIDE G IS:* $\sqrt{P^2+Q^2-2PQ\cos C}$

```
10  PRINT 'TYPE LENGTHS (P&Q) & INCL. ANGLE (DEGR.)'
20  LET K = 3.141592/180
30  INPUT P, Q, D
40  LET C= D*K
50  LET S= 0.5*P*Q*SIN(C)
60  LET A= ATN(Q*SIN(C)/(P-Q*COS(C)))/K
70  LET B= ATN(P*SIN(C)/(Q-P*COS(C)))/K
80  LET G= SQR(P*P+Q*Q-2*P*Q*COS(C))
90  PRINT "AREA";S; "OPPOSITE SIDE";G
100 PRINT 'BASE ANGLES"; A ; "AND' ; B
110 END
```

DEGREES TO RADIANS

RADIANS TO DEGREES

RND

BASIC CAN "THROW DICE" TO GENERATE RANDOM NUMBERS ⇄ *PSEUDO* RANDOM NUMBERS TO BE MORE PRECISE.

```
10  REM   TO ILLUSTRATE  RND
20  LET  X = RND
30  LET  Y = RND
40  LET  Z = RND
50  PRINT  X;Y;Z
60  END
RUN
·240643   ·417191  ·192204
```

RND IS A FUNCTION WITHOUT AN ARGUMENT

THE SAME RESULT EVERY RUN

NOTE: THE ORIGINAL DARTMOUTH BASIC, & SEVERAL EXISTING ONES, REQUIRE AN ARGUMENT:

LET X = RND(A).

SOME BASICS IGNORE THIS ARGUMENT; OTHERS USE IT IN SUBTLE WAYS SUCH AS TO DENOTE DIFFERENT "STREAMS" OF NUMBERS.

IN AN EXPRESSION "RND" MAY BE TREATED IN EXACTLY THE SAME WAY AS A NUMERICAL VARIABLE; WHEREVER YOU MAY TYPE X YOU MAY ALSO TYPE RND. STATEMENTS 20 TO 50 ABOVE COULD BE COMBINED INTO ONE STATEMENT:

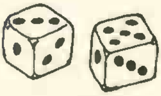
```
20  PRINT  RND; RND; RND
```

WHEN A PROGRAM IS RUNNING, EVERY TIME *BASIC* MEETS "RND" IT SUPPLIES A RANDOM NUMBER, n, WHERE $0 \le n < 1$ (NB. SOMETIMES 0, NEVER 1). IT IS NOT REALLY RANDOM, IT IS CALLED *PSEUDO RANDOM*. ON MEETING RND THE COMPUTER GENERATES AND PROVIDES THE NEXT IN A FIXED CYCLE OF NUMBERS. *BASICS* DIFFER, BUT ONE CYCLE COMMONLY USED HAS A MILLION SIX-DIGIT FRACTIONS FROM 0.000000 TO 0.999999. IN EVERY CYCLE EACH FRACTION OCCURS EXACTLY ONCE. THE CYCLE STARTS AFRESH IN EVERY RUN; IF A PROGRAM USES ALL THE NUMBERS IN THE CYCLE THEN THE CYCLE BEGINS AGAIN. THE PROGRAM ABOVE WOULD GIVE THE SAME RESULT IN EVERY RUN ⇄ BUT A DIFFERENT VERSION OF *BASIC* MIGHT PRODUCE THREE COMPLETELY DIFFERENT NUMBERS. HOWEVER ...

RANDOMIZE

THIS STATEMENT MAKES *BASIC* START THE CYCLE AT AN UNPREDICTABLE PLACE ON EACH ENCOUNTER.

THIS PROGRAM THROWS A PAIR OF DICE:



```
10  RANDOMIZE
20  LET X=INT(1+6*RND)
30  LET Y=INT(1+6*RND)
40  PRINT "THROW:"; X ; "AND";Y
50  END
RUN
THROW:  3  AND 5
```

"RANDOMIZE" ENSURES AN UNPREDICTABLE THROW ON EACH RUN

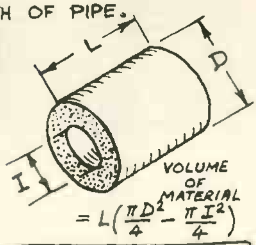
SOME *BASICS* DON'T HAVE RANDOMIZE; INSTEAD THEY USE AN ARGUMENT, RND(A). THE WAY "A" IS USED DIFFERS FUNDAMENTALLY FROM *BASIC* TO *BASIC*; YOU HAVE TO CONSULT THE SPECIFIC MANUAL.

DEF

IN *BASIC* YOU CAN *DEFINE* UP TO 26 OF YOUR OWN FUNCTIONS USING THE DEF STATEMENT. THESE FUNCTIONS ARE NAMED FNA, FNB, FNC *etc.*

COMPARE THESE TWO PROGRAMS, BOTH OF WHICH CALCULATE AND PRINT THE VOLUME OF MATERIAL IN A LENGTH OF PIPE.

```
10   PRINT  "TYPE: L,D & I"
20   INPUT           L,D,  I
30   LET A1 = 3.141592*D↑2/4
40   LET A2 = 3.141592*I↑2/4
50   LET  V = L*(A1-A2)
60   PRINT "VOLUME IS";V
70   END
```



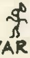
"DEF" IS SHORT FOR "DEFINE"

```
10   DEF  FNC(X)=3.141592*X↑2/4
20   PRINT "TYPE: L,D & I"
30   INPUT         L,D,  I
40   LET  V = L*(FNC(D) - FNC(I))
50   PRINT "VOLUME IS";V
60   END
```

ONCE YOU HAVE DEFINED A FUNCTION (AS IN LINE 10 ABOVE) YOU MAY USE IT EXACTLY AS THOUGH IT WERE AN *INTRINSIC* FUNCTION.

WHAT HAPPENED TO THE VARIABLE X ? NOTHING: IT IS CALLED A *DUMMY ARGUMENT*. THIS "DEF" STATEMENT SAYS: "COMPUTE THE VALUE OF WHAT LIES IN BRACKETS; SQUARE IT; MULTIPLY THE SQUARE BY 3.141592; DIVIDE THIS PRODUCT BY 4 AND GIVE THE RESULT TO FNC( )". IT WOULD BE JUST AS LOGICAL TO WRITE:

10 DEF FNC( )=3.141592* ↑2/4

BUT THERE IS NO SUCH CHARACTER AS  ON THE KEYBOARD SO *BASIC* INSISTS ON THE NAME OF A VARIABLE TO BE USED AS A DUMMY. THE VARIABLE X IS USED ABOVE BUT *BASIC* WILL NOT CHANGE THE VALUE OF X AT STATEMENT 40.

UNFORTUNATELY THERE ARE *BASICS* THAT *WOULD* CHANGE THE VALUE OF X AT STATEMENT 40 ABOVE. YOU CAN TEST YOUR VERSION BY RUNNING THE FOLLOWING PROGRAM.

```
10   REM   TEST  FOR DUMMY DUMMIES
20   DEF  FNA(T) = T+2
30   LET  T=0
40   LET  A= FNA(3)
50   PRINT  T; "SHOULD STAY ZERO"
60   END
```

IF YOUR VERSION PRINTS A 5 THEN YOU WILL HAVE TO KEEP VARIABLES DISTINCT FROM ARGUMENTS: SEE OPPOSITE

YOU MAY ALSO DEFINE FUNCTIONS *WITHOUT* ARGUMENTS ⇒ SEE OPPOSITE.

HERE ARE SOME EXAMPLES OF FUNCTIONS DEFINED:

```
10  REM   FUNCTIONS WITH NO ARGUMENT
20  DEF  FNI = INT(2+11*RND)
30  DEF  FNR = 3.141592/180
40  REM
50  REM  FUNCTIONS WITH DUMMY ARGUMENT
60  DEF  FNC(X) = 3.141592*X*X/4
70  DEF  FNM(X) = X * SGN(1+SGN(X))
```

RANDOM INTEGER IN THE RANGE 2 TO 12

MULTIPLY DEGREES BY FNR TO GET RADIANS

AREA OF A CIRCLE OF DIAMETER "DUMMY X"

"MACAULAY" OR "DIRAC" FUNCTION OF X
=X IF X≥0
=0 IF X<0

YOU MAY ALSO HAVE "ARGUMENTS" THAT ARE NOT DUMMIES; THE FUNCTION USES VALUES OF VARIABLES IN YOUR PROGRAM.

```
80  REM  FUNCTIONS USING PROGRAM VARIABLES
90  DEF  FNH = SQR(A*A + B*B)
100 DEF  FNA(X)= 0.5 * A* B* SIN(X)
110 DEF  FNL = P - INT(P/Q)* Q
120 DEF  FNT(X) = INT(X*10↑D+·5)/10↑D
130 DEF  FNG = INT(A+(B-1)*RND)
```

HYPOTENUSE OF RIGHT ANGLED TRIANGLE WITH NORMAL SIDES A&B

AREA OF A TRIANGLE WITH SIDES A&B AND INCLUDED ANGLE "DUMMY X"

FNL IS THE REMAINDER WHEN P IS DIVIDED BY Q

ROUND "DUMMY X" TO D DECIMAL PLACES

FNG IS A RANDOM INTEGER IN THE RANGE A TO B

BASICS DIFFER. IF YOU WANT TO WRITE "PORTABLE" PROGRAMS OBSERVE THE FOLLOWING RULES:

★ MAKE SURE THE DEF STATEMENT IS ON A LOWER-NUMBERED LINE THAN THE FIRST USAGE OF THE FUNCTION.

★ MAKE SURE YOUR PROGRAM *OBEYS* THE DEF STATEMENT ONCE. (DON'T JUMP ROUND IT WITH "GO TO" ⇒ SEE P.40.)

★ EVEN IF YOUR VERSION ALLOWS IT, DON'T HAVE MORE THAN ONE DUMMY ARGUMENT. *e.g.* DON'T HAVE `DEF FNA(A,B,C)=A+B+C`

★ DON'T MAKE FUNCTIONS MORE THAN ONE LINE LONG. SOME *BASICS* ALLOW THIS; YOU SIGNIFY THE END OF A MULTI-LINE DEFINITION WITH THE STATEMENT "FNEND".

★ ALWAYS PUT A DUMMY ARGUMENT EVEN IF NOT USED. *e.g.* WRITE LINE 30 ABOVE AS DEF FNR(0)= 3.141592/180. SOME *BASICS* DON'T ALLOW FUNCTIONS WITHOUT AN ARGUMENT.

★ FOR SAFETY USE LETTER O (ALSO O1, O2, *etc.*) EXCLUSIVELY AS DUMMY ARGUMENTS ⇒ NEVER AS VARIABLES! SOME *BASICS* DO NOT TREAT ARGUMENTS AS TRUE DUMMIES. (SEE OPPOSITE FOR THIS TEST.)

Why Pilot can save you time and effort

One of the best languages for writing computer-assisted instruction programs is now available on a micro. It is a simple language which anyone can use and it is cheap.

by CHARLES SWEETEN

AT 10 o'clock one evening, someone rang me and said "I've just had something from the States—you had better come and see it". It proved to be the latest implementation of a language called PILOT, which has been under development since 1971.

Pilot is an author language which has been designed for writing Computer Assisted Instruction (CAI) programs. It has often been assumed that a great deal of time and effort is needed to write CAI programs and anyone who has tried has become aware rapidly of the shortcomings of languages such as Fortran and Basic for this purpose.

Those languages are not designed to deal with text in a flexible, interactive way, and projects using them for CAI programs have needed heavy investment to make any progress.

Those languages with text processing features, like PL/1, are not commonly available to the teachers who need to write CAI programs, and they are troublesome to learn. As a result, most CAI programs have been written by professional programmers on large machines, and this creates barriers—the programs are difficult to implement elsewhere, the individual teacher is unable to influence the shape of the program in the direction he wants it, and professionals always want to write large programs which take time to write and test. This makes the use of CAI expensive and inflexible.

Pilot is cheap

What is needed is a simple language anybody can use which can be implemented on a wide variety of machines. Pilot is such a language. It is cheap, it is easy-to-use, and it works.

In this version, Pilot is a standardisation of a number of extended versions which were developed at a number of institutions in the U.S. These in turn were developed from CORE PILOT (or PILOT 73) which originated at the San Francisco Medical Centre.

The real significance of this version is that it is implemented on the South West Technical Products M6800-based microcomputer. Previously, the only versions available here have been on mainframes and minicomputers. There is, for example, an earlier version (PILOT 73) on the Open University HP2000F system in section 801 Mathematics (Education).

Tutorial programs

There is also a series of tutorial programs about the language. In other sections there are a few programs written in Pilot—P-HYP, P-KEPL, P-MAG. The ILEA service on the City of London Polytechnic HP2000E has PILOTE which again is PILOT 73. There are also the tutorials and P-HYP, and Aston University has PILOT F. Data General has CO-PILOT for running on Nova series computers. So you can see that Pilot is around in its older versions.

Pilot in this version on the SWTPC system has been produced at Western Washington University by Larry Kheriaty, George Gerhold and SWTPC. It is a text-orientated language which is built round the process of examining a text string input and matching any part of that in various ways. Conditional branching is then made on the basis of the last matching attempt or any relational expression.

Calculation is done on a calculator chip and, due to the high price of the faster chip, is at present very slow. Program length is limited only by disc space, and up to four users can use the system—even the same program if they wish—at the same time.

The hardware required for a single user system is a SWTPC computer, 16K of memory from 0 to $3FFF, memory for the disc operating system, a calculator board, a serial I/O board, and a disc system. Pilot requires only one disc drive though it can access others.

MEMORY MAP

0000 – 0FFF	multi-user board
1000 – 2FFF	PILOT
3000 – 3FFF	terminal I scratch pad
4000 – 4FFF	terminal 2 scratch pad
5000 – 5FFF	terminal 3 scratch pad
6000 – 6FFF	terminal 4 scratch pad
7000 – 7FFF	FLEX for mini-floppy disks
A000 – BFFF	FLEX for 8 in. floppy disks

A Pilot program consists of a set of ordered but not numbered statements, any of which may have a special label. Each statement consists of an optional label, operation letter, various optional modifying letters, and an argument or text field. The operations available are:

PR:	set options
R:	remark (not processed)
T:	type text
:	continuation of last type
A:	accept from keyboard
M:	match input buffer
J:	jump
U:	jump subroutine
E:	return from subroutine
C:	compute
D:	dimension
XI:	execute instruction in string

For example, possible statements are:

```
* TEXT T : This line of text will be typed
T (B >0) : so will this if B >0
J  (B <0) : TEXT
```

Computation follows common rules but has, needless to say, its own combination of syntax rules. Type will print almost any combination of text and variables.

Accept will store any alphnumeric string in an input buffer. Various options are possible, such as removal of all spaces, or conversion to upper- or lowercase, or assignment to a string variable, or assigning the first numeric sequence in the buffer to a numeric variable.

Match carries the above options a stage further and is really the most

(continued on next page)

(continued from previous page)

important operation. The program searches the input buffer for a sequence which matches, or nearly matches, a given sequence. The result of this match sets a flag. The given sequence may request a search for both of two strings (or more), or a search for either of two strings (or more). Spaces and arbitrary characters may be indicated, and spelling mistakes which include two characters back to front and one wrong character may be allowed for.

The use of the other operation codes is what you would expect in each case.

Backus Naur Form Definition of PILOT
SWTPC August 16 1978 Charles Sweeten

```
<program> : : -( <statement> )<E statement>
<statement> : : - <PR statement> ! <T statement> !
                  <A statement> ! <M statement>!
                  <J statement> ! <E statement> !
                  <C statement> ! <R statement> !
                  <U statement> ! <D statement> !
                  <XI statement>
<PR statement> : : -( * <label> ) PR:( <option
                    letter>)
<T statement> : : -( * <label> ) T ( <modifier> )
                    ( <conditioner> ) ( <relational
                    expression> ) : <space> (
                    <text> ) ( <variable> <space>
                    ) ( <text> ) )
<A statement> : : -( * <label> ) A ( <modifier> ( )
                   <conditioner> ) ( <relational
                   expression> )  : <space> (
                   <variable> )
<M statement> : : -( * <label> ) <modifier> ) (
                    <conditioner> ) ( <relational
                    expression> ) : <space>
                    <string> ( , <string> )
<J statement> : : -( * <label> ) J ( <conditioner> ) (
                   <relational expression> ) :
                   <space> ( <label> )
<E statement> : : -( * <label> ) E ( <conditioner> ) (
                 <relational expression> ) :
                 <space> ( <label> )
<C statement> : : -( * <label> ) C ( <conditioner>
                   ) ( <relational expression> ) :
                   <space> <assignment>
<R statement> : : -( * <label> ) R : <space> (
                   <string> )
<U statement> : : -( * <label> ) U ( <conditioner> ) (
                   <relational expression> ) :
                   <space> <label>
<D statement> : : -( * <label> ) D : <space>
                   <variable> ( <number> ( ,
                   <number> ) )
<XI statement> : : -( * <label> ) XI ( <conditioner>
                    ) ( <relational expression> )
                    : <space> <string variable>
<label> : : - <letter> ( <alphanumeric> )
<modifier> : : - H ! J ! S ! X ! <empty>
<conditioner> : : - Y ! N ! ( <expression>
                   <relational operator>
                   <expression> ) ! E ! C  !
                   <digit> ! <empty>
```

Some examples will illustrate what can be done.

```
T:    ARE YOU TIRED ?
A:
M:    NO, NOT, NEVER
TY:   GOOD, THEN WE'LL BEGIN
TN:   GO AND GET SOME SLEEP
 :          BYE
JN:   END
```

The A statement accepts any text message from the keyboard and stores it in the input buffer. The M statement scans the input buffer and searches for 'no' or 'not' or 'never'. If any of these are found a Yes flag is set, otherwise a No flag is set. Y and N are then available to use as conditioners to following statements.

The TY statement will type text only if the match was successful, and the TN statement will type text only if the match was unsuccessful. The colon statement is a continuation of the preceding type statement. The J statement causes a jump to the label 'END' elsewhere in the program, provided the last match was unsuccessful.

As can be seen, the Accept and Match facilities are at the heart of the language, and it is those statements which are capable of most adjustment.

All statements, however, may be subject to modifiers, conditioners and relational expressions. The Y and N conditioners depend on the last Match. There is a conditioner which depends on an error condition being set, and there is a conditioner which matches the number of times the last Accept statement has been used consecutively.
For example:

```
T:     TYPE DOG
* ANS A:
M:     CAT
JY:    NEXT
T1:    WRONG
T2:    WRONG AGAIN
T3:    WRONG YET AGAIN
T4:    TYPE CAT
J:     ANS
* NEXT R:
```

The computer will type "TYPE DOG" and you will answer "DOG" presumably. The first time this happens the program fails to get a match and types "WRONG" and jumps back to *ANS. You type "DOG" again and the computer types "WRONG AGAIN". After the fourth message from the computer you presumably type "CAT" and this gives a successful match, and the program jumps to the line labelled *NEXT.

The Match statement can be modified so that an automatic jump occurs to the next Match statement if the match is unsuccessful. For example:

```
T:     WHAT COLOUR IS THE SKY?
A:
MJ:    BLUE
T:     THAT'S NICE
J:     NEXT
MJ:    GREY
T:     SOUNDS LIKE RAIN
J:     NEXT
M:     PINK
TY:    THE SHEPHERD'S DELIGHT
```

If the answer given is "BLUE", then the program types "THAT'S NICE" and jumps to *NEXT. If the answer was not "BLUE" the program jumps to the next match statement, which tries to match with "GREY". If this is unsuccessful the program jumps to the next match statement which tries to match with "PINK". If this is also unsuccessful the program will continue.

The Match statement may be modified so that any letter may be wrong and two letters reversed. For example:

```
    T:     What are you reading?
    A:
    MS:    article
will match with:  a Computer article
                  an artical
                  an arty clerical assistant
                  a particularly bad match
```

Any statement can be made subject to the satisfaction of a relational expression to the satisfaction of a relational expression. For example:

```
T (ABS (x - 36·9) (·5) :Your temperature
                       : is normal
```

This will have the effect of typing "Your temperature is normal" only if the difference between X and 36·9 is less than ·5.

The Match statement is made more flexible by the use of 'and' (&) and 'or' (!). For example:

```
    M: CALCIUM & IDE will match with CALCIUM
CHLORIDE.
    M: WINDSOR ! BUCKINGHAM ! SANDRINGHAM
will match with WINDSOR CASTLE OR
BUCKINGHAM PALACE or with SANDRINGHAM
ESTATE.
    M: red ! blue & green ! yellow and purple will
match with a red house, bluebottle and greenfly
orange and yellow, but not purple, but not with
Fred saw green leaves with purple spots greenfly
and bluebottle
```

Suppose we wanted to elicit the response Richelieu, we could use M: %R&H&L&U where the % indicates that there must be a space immediately in front of the R.

Any line may be given a label and those labels are used as a destination for a Jump statement.

Here is a section of a program which is attempting to extract the solution of an equation. I have numbered the lines for ease of reference, though these numbers are not part of the program. The language is a bit flowery, but this is excusable in a demonstration program.

(See program on page 57)

1– This a label some considerable distance through the program. It marks the beginning of a logical section and is followed by a statement setting no options.

2– $B1$ is a stored variable containing the name of the user of the program, which has been requested at the beginning.

3– A match is attempted with '3m/8'. In fact, the match merely looks for the sequence 3 and m and / and 8. It also looks for '3/8,'. Mathematically this is hardly foolproof, but it is a vast improvement on what could be done in Basic. If the match is good the program continues and jumps to *ANAL3, but if the match is bad the program jumps to the next match statement.

4– The same analysis is made here for solving the equation the other way round. A good match causes a jump to the alternative section *ANAL4.

5– This tests to see if the user has the answer upside down, and if this proves to be the case, sends him back to the last Accept statement (from point 6) so that he can try again.

7– This looks for either 6b or 6m, but preceded by a space or the start of

(continued on next page)

```
1*SLAVE4
    PR:
    T:


2   T: Now, consider, $BI$ , what we know. The
    merchants began
    T: with 3m slaves. Of these slaves, m/2 remain
    to the
    T: merchants, 6b fell to the brigands, and m–2b
    escaped.
    T: In other words, in the language of the scribes:
    T:      3m = m/2 + 6b + m − 2b.
    T: Can you solve this equation for b (or m– – take
    pick)?
    *SLAVA1
    PR:
    C: J=0
    A:


3   MJ: 3&/&8&m ! 3&m&/&8
    J: ANAL3


4   MJ: 8&/&3&b ! 8&b&/&3
    J: ANAL4


5   MJ: 3&/&8&b ! 8&/&3&m ! 3&b&/&8 ! 8&m&/&3
    T: Oh, $BI$ , you have that upside down! Try
    it again.
6   J: CA


7   MJ: %6b ! %6m
    T: WHAT ! A thousand curses upon your algebra
    teacher for
    T: allowing you to sleep through your lectures
    $BI$.
    T: To simplify 3/2 m = 4b
    T: you must multiply both sides by the same
    factor.
    T: (Like either 1/4 or 2/3 ! )      Now try again.
    J: GA
    MJ: yes
    T: Very well, go ahead and solve it!
    J: GA
    MJ: no
    T: May the desert palms protect you, my child!
    Give it
    T: a try, and remember the 3 steps:
    T:   1.   gather like terms (set the b's on one side
    T:      —of the equation, and the m's on the other
    T:        side)
    T:   2.   subtract
    T:   3.   multiply both sides of the equation by·
              the same
    T:        factor to solve for b (or m).
    J: GA


8   M: *
    C: S6=0


9   U: SWEAR
    J(S6=0) : GA
    R:


10  J(J>0) : GM
    C: J=1
    T: My friend, merely scritch b = (or m = ) in
    the dust
    T: here, then scritch what it equals.


11  J: GA
    M: *
    J(J>1) : GM
    C: J=2
    T:
    T: $BI$ , my friend, merely solve the infidelish
    equation!
    T: (I'll even help you:
    T: you should get 3m – m – m/2 = 6b – 2b
    solve this ! !
    J: GA
    M: *
    J(J>2) : GM
    C: J=3
    T: Oh, your algebra is long rusted! 3m – m – m/2
    = 6b – 2b
    T: reduces to
    T: 3/2 m = 4b. Okay, see if you can solve this.
    J: GA
    M: *
    J(J>3) : GM
    C: J=4
    T: GIVE ME THAT STICK ! ! ! $BI$ , if the
    T: desert wind should
    T: blow past your ears, your head would resonate.
    T: To solve 3/2 m = 4b multiply both sides by a
    factor
    T: to get rid of either the 3/2 or the 4. What
    T: equation
    T: do you set?
    J: GA


12  M: *
    T: The answer is m = 8/3 b or, b = 3/8 m
    T: so you see, my child, that you can have your
    pick . . .
    T: NOW, WHAT IS THE ANSWER?
    J: GA
```

```
13  *ANAL3
    PR:
    C: J=0
    T: Well done! This gives us:
    T:
    T:         m merchants
    T:         m camels
    T:       3m slaves
    T:     3/8m brigands
    T:
    T: Since it would be interesting to say the least,
    T: to see 3/8 of a brigand charging over the dunes,
    T: we must obviously make m greater than one.
    What number would you pick for m?
    A:
    MJ: 8!eig


14  J: CONCU


15  M: *
    C: S6=0
    U: SWEAR
    J (S6=0) : GA
    J (J>0) : GM
    C: J=1
    T:   No. What number would you pick for m, so
    T: that there is the smallest possible number of
    T: brigands, and yet no fractions of brigands?
    J: CA
    M: *
    J (J>1) : GM
    C: J=2
    T: Think, my friend, think before you try again.
    J: GA
    M: *
    J (J>2) : GM
    C: J=3
    T: (Curse my ignorance for not taking payment
    in advance ! ! )
    T:   $BI$ , if I may be so bold, try eight as an
    answer.
    J: CA
    M: *
    T:   GET OUT OF MY SIGHT YOU YOUNG
    FOOL!
    J: CA


16  *ANAL4
    PR:
    C: J=0
    T: Well done! This gives us
    T:             b   brigands
    T:          8/3 b  camels
    T:             8b   slaves
    T:          8/3 b   merchants
    T: since it would be interesting, to say the least,
    T: to see 2 and 1/3 camels humping along, it is
    T: obvious that b must be some number greater
    T: than one, to multiply our 1/3 camel into
    T: something a little more presentable. What
    T: number would you pick for b!
    A:i
    MJ: 3!thre


17  J: CONCU
    M: *
    C: S6=0
    U: SWEAR
    J(S6=0) : GA
    J (J>0) : GM
    C: J=1
    T:   No. What b gives us whole camels and
    T: merchants, but the smallest number of brigands!
    J: GA
    M: *
    J (J>1) : GM
    C: J=2
    T:   Think, my friend, think before you try again.
    J: CA
    M: *
    J (J>2) : GM
    C: J=3
    T:   (Curse my ignorance for not taking payment
    in advance ! ! )
    T: $B!$ , if I may be so bold, try three as an
    answer.
    J: GA
    M: *
    T:   GET OUT OF MY SIGHT YOU YOUNG
    FOOL !
    J: GA
    *CONCU
    PR:
    C: C1=0
    T: Beautiful, $BI$ !
    T: This gives us:
    T:             3  brigands
    T:             8  merchants
    T:             8  camels
    T:            24  slaves
    T:
    T: So we have at least three brigands swooping
    T: from the dunes. Well, my young desert runner,
    T: that is an example of problem solving. All
    T: lovely and logical – – do you still wish to be
    T: apprenticed to me?
    A:
    MJ: no ! on
    T: Oh well – – you are probably cut out for an
    T: arts student. anyway. Farewell, $BI$ , and
    T: long life.
    E:
```

(continued from previous page)

the line. I think the logic is questionable here, but you can see the type of mistake the program is trying to spot.

8– This attempts to match the answer with any one character, and will always be successful.

9– This sends the program to a subroutine at *SWEAR to find if the answer contains a rude word. If it does, the routine will print an appropriate reply and will set S6=0. The program then jumps back for another answer if the rude word was detected, but continues otherwise.

10– J is at present zero, but if the user comes past this point again, J will have been incremented, and in that case a jump would be caused to the next match statement.

11– This sends the user back to the Accept statement on the first occasion he gets that far. The following lines deal with the subsequent times he gives wrong answers.

12– As you can see, by the time the user has given four wrong answers, the computer responses are getting more and more to the point. There is obviously a limit to the number of wrong answers which can be dealt with differently, and this program will keep repeating this last response, given further wrong answers.

13– This section deals with the correct answer of 3m/8 and seeks to get the next answer of a multiple of 8. Some thought should, I think, have gone into this, rather than accepting only 8.

14– A correct answer sends the program to *CONCU.

15– This section deals with three incorrect answers, looks for rude words, and finally gives the right answer. If you disagree with giving the answer in the end, I suggest you try this program when possible; it takes so long to give enough wrong answers that you would lose patience rapidly and make an effort to finish.

16– This deals with the correct previous answer of m=8b/3.

17– This section just about wraps it up, and does, in fact, have a possible End point.

I have not given a run of this section of the program as there are so many ways in which it could run. It is worth the effort of working through it to see what the possibilities are, as this language represents a totally different concept from the one we are used to.

Provided you have the right kind of microcomputer in the first place then it is available now. Hopefully, it will soon be available on all the others.

Charles Sweeten is director of computing at Oundle School and is secretary of MUSE (Minicomputer Users in Secondary Education).

TAKING THE CHORE OUT OF VAT: PART II

IN THE first part of this article published last month we considered the problems of computerised VAT accounting and looked at the first two programs of the VAT suite. We also considered the hardware and software those programs require.

For those who missed it, this was a Cromemco Z-2 with 48K, a dual Persci 8 in. disc unit from Info 2000, an ADM-3, and a Teletype 33, with the whole system running under CP/M and the programs written in TDL Basic.

The first two programs in the VAT suite were a data entry and transaction create program, and a transaction file memory sort. Now we look at the remaining two programs and consider how we run the whole suite of VAT programs.

PROGRAM THREE, THE MASTER FILE UPDATE

10–390 PROGRAM INITIALISATION & FILE SET UP.
400– 450 INITIAL READS OF UPDATE & MASTER INPUT FILES.
460– 770 'BALANCE LINE' LOGIC FLOW.
460– 520 LOWEST KEY SELECTION & END OF JOB DETECTION.
530– 650 MASTER FILE (DRIVER) INNER LOOP.
660– 770 UPDATE FILE INNER LOOP.
780–1730 VARIOUS FILE MATCH & NOMATCH PROCESSING.
810– 950 MASTER FILE RECORD PRESENT, NO UPDATE WITH SAME KEY.
960–1130 MASTER RECORD NOT PRESENT, - UPDATE IS 'ADD'. (O/P TO MASTER).
1140–1180 MASTER RECORD NOT PRESENT, UPDATE IS 'DELETE' OR 'REPLACE'.
(THIS IS AN ERROR CONDITION—UPDATE NOT ACTIONED)
1190–1330 MASTER RECORD PRESENT, BUT UPDATE IS 'ADD'. (ANOTHER ERROR)
1340–1450 MASTER RECORD PRESENT, UPDATE IS DELETE. (NO O/P TO MASTER)
1470–1620 MASTER RECORD PRESENT, UPDATE IS 'REPLACE'. (MASTER RECORD
REPLACED BY RECORD ON UPDATE FILE, WHICH IS O/P TO MASTER)
1630–1730 ERROR ROUTINE TO HANDLE DUPLICATE KEY UPDATE RECORDS.
1740–1820 UPDATED MASTER FILE O/P ROUTINE.
1830–2000 OLD MASTER FILE INPUT ROUTINE.
2010–2200 UPDATE FILE INPUT ROUTINE.
2210–2340 DUPLICATE RECORDS ON INPUT MASTER FILE ERROR ROUTINE.
2350–2600 END OF PROGRAM CONTROL TOTALS ROUTINES & END OF JOB.

This program updates the old input master file with the records on the transaction update file, producing a 'new' updated master file. There are many techniques used to achieve this, but I have used the one known as the 'balance line'. I am indebted to Chris Dunford of Altergo Ltd for introducing me to it in 1973.

The technique has a number of advantages; the code can be modified to handle as many update files as required; it takes care automatically of end of file conditions on either file; it presents correctly the records from all files matched by key to the processing logic; it is very simple, once understood.

The two variants of the method are presented in flowcharts 1 and 2. I have used the second one outlined in flowchart 2.

Variant one uses the lowest key from all files to auto-sequence the matching process. The files are read in sequence to determine the lowest key; with the record being stored if from the master file, and processed if from the update.

In variant two, the update processing does not occur until the output of the master file record processing in variant one. This is legal only if a two-file match is taking place and duplicate key update records are not to be allowed.

by
GEOFF LYNCH

The program starts by asking the user for the full file names of the three files to be used. The user should enter the name of the master file output by the last run of the update program when asked for the name of the input master file (otherwise known as the 'brought forward' master).

When asked for the name of the update file one enters the name of the latest *sorted* transaction file. When asked for the name of the output file. A new file name is entered, preferably containing an increasing numbers for easy record-keeping, e.g.

```
VATQT02.MAS.VATQT03.MAS
```

These file names will be repeated on the console for user checking before the files aıe opened within the program. A non-'Y' answer will repeat the user data entry process for the file names. The user is also asked if a full audit trail should be printed, consisting of all records output to the master file. Even unchanged ones. If this question is answered with 'N', only changed records will be printed.

Each line of printout consists of a single character code to indicate the action which has occurred, followed by an unformatted print of each record.

INDICATORS

A—RECORD ADDED TO THE OUTPUT MASTER FILE.
U—RECORD OUTPUT UNCHANGED FROM INPUT TO OUTPUT MASTER FILE.
D—RECORD DELETED FROM MASTER FILE (IE NOT OUTPUT, BUT WAS ON INPUT)
O—OLD RECORD FROM MASTER FILE WHICH IS REPLACED BY NEW MASTER RECORD.
N—NEW REPLACEMENT RECORD ON MASTER FILE. (FROM UPDATE FILE)
E—UPDATE RECORD IN ERROR. NO ACTION OCCURRED. RECORD IGNORED.

When all processing has finished, control totals are printed. See 'Running the System' for further explanation.

PROGRAM FOUR, THE VAT RETURN GENERATOR.

10– 610 PROGRAM INITIALISATION & FILE SET UP.
620– 702 MAIN READ FILE LOOP, DETERMINES EXTRACTION.
730–1100 ACCUMULATION INTO VAT & VALUE ARRAYS.
1110–1640 PRINT OF PART A OF RETURN.
1650–2030 PRINT OF PART B OF RETURN & END OF JOB.
2040–2120 SUBROUTINE TO VALIDATE EXTRACTION DATES.
2130–2370 MASTER FILE RECORD READ SUBROUTINE.
2380–2460 PAGE HEADINGS SUBROUTINE.
2470–2520 LINE PRINT SUBROUTINE.

This program is very simple, given that you are familiar with VAT. All the data to fill out a VAT return must be added up from the data on the master file, but only for the VAT period pertaining to the currently-required report To enable this to be done, only records which have a tax point date within the period are extracted for accumulation, This is done by reading all records, but only passing on to the addition routines those which have a valid tax date. The next most complicated problem is that of ensuring that the information is collated correctly into the various categories required by Customs and Excise. As far as I am aware, this is achieved correctly within my program but no guarantee can be given, as I am not a VAT expert. I know that the returns made by me have been scrutinised once by the Vatman, and now fit in with his interpretation of the various categories.

The array used to hold the data is organised so that the second subscript pertains to the VAT rate with the highest valid subscript being set aside for the total of all values within the same first subscript. Thus V(2, 4) holds the total of V(2, 1), V(2, 2), and V(2, 3).

The first subscript is the 'line' number on the VAT return, and thus an array element can hold either VAT values or the value required for control purposes on lines 11–17.

In the latter case the use of the second subscript is irrelevant, but convenient, as the VAT rate is not a part of the control discrimination on the VAT return for lines 11–17.

I still use it, however, to add the relevant value into the array, but then add up into element (X, 4). This means that should the format of the return change and this information be required, it is readily available.

The user is asked on startup of the pro-

(*continued on next page*)

(*continued from previous page*)

gram if a full audit trail is required. Answering with a 'Y' will produce a printout of all records which make up the totals on the VAT return.

SYSTEM STARTUP

The system needs a null file to start the update cycle as the update program must have two files input to it, and on the first run only the update file contains valid information. This is achieved by creating a 'null' file using program one, the transaction create program. Run program one, and when asked for the filename to be created, answer 'NULL.DAT or some other meaningful name. On the very first data entry, input ',0,0' when asked for the data, transaction code and serial number. The program will then ask if all data has been entered, to which one answers 'Y . This will create a file with no transactions at all on it, though still a perfectly valid file.

Keep this file until required in the first run of the program three, the update program. When one runs program three for the first time, a normal transaction file containing real transaction must already have been created.

When asked for the name of the input master file, one answers with 'NULL-DAT', the name of the file created.

DATA ENTRY, SORT AND UPDATE CYCLE

Correct running of the system consists of repeated runs of the transaction create, sort and update programs in 'cycles', followed by a run of the VAT report program when required. The data entry, sort and update cycle is run normally in the same session, unless you have system experience with CP/M and wish to do a number of runs of the data entry program, and concatenate all the transaction files not yet actioned into one large transaction file prior to the sort.

The cycle starts by running program one and creating a transaction file as described last month in the discussion of program one. A typical set of output from such a run is seen at the beginning of sample run 2. Please notice I have included two transactions which will result in errors on the fjrst update run. The sample includes sales, purchases and an import with deferred VAT. When all data entry has been completed for a run, the sort is run. The only action on your part is to specify the name of the input file, the file just created in the transaction create program. The program does the rest, including changing the name of the input file. The last process is to run the update program. The input master file name must be specified, and the name of the sorted transaction file, the file just output by the sort. You will notice that this name is printed as the last line of information by the sort, so that all you have to do is refer to the printout.

You also have to provide the name of the output master file, a file about to be created for the first time. I advise users to use a naming system for their master files where the name incorporates a number, such as 'VATMAS01.MAS'. This has two advantages, firstly when one is entering the filenames, one mentally adds 1 to the number of the input master file as it is entered. Secondly, if you have entered the wrong input master file name, the output will be wrong as a result, and when the program opens the files, a file open error will result, with a message telling you that a duplicate file already exists.

Don't forget that the input master file should be the last file output by the update program. In the case of your first run, it should be the null file as created in sample run 1. See sample run 2 for an example of such a run.

The transaction file used in this run has a replacement transaction and a delete. Neither of these are valid in this example, as no records exist on the master file to be replaced or deleted. As you can see, they have been flagged by the update program with asterisks on the right, and they have an 'E' at the beginning to indicate that they are in error. They have not been actioned. All the other records were additions to the file and they have been added successfully, as indicated by the 'A' prefix.

In sample run 3, this file created (updated) in run two is being updated with the transactions listed at the beginning of run 3. Notice there are two additions, a replacement for the import and a delete of 'test sale 3'. On examining the listing output of the update in run 3, you will notice that I have made no errors, and a number of transactions have a 'U' prefix. This indicates that they are unchanged since the last run. The record with the 'D' prefix has been deleted, and is not on the output master file. The record printed is the one from the input master file.

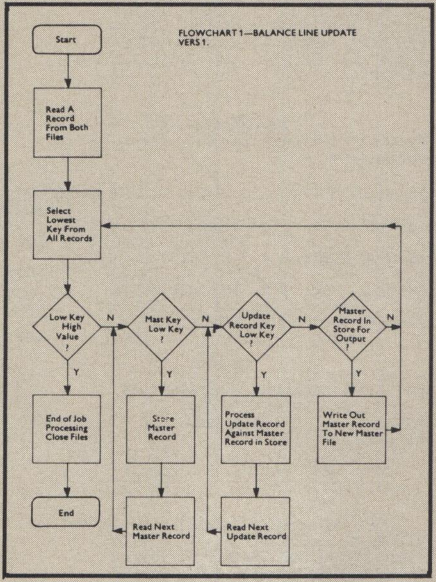
The records printed with a dotted line before and after are the old record which has been replaced, and the new one replacing it. Again it is a good idea to check that the right one has been replaced.

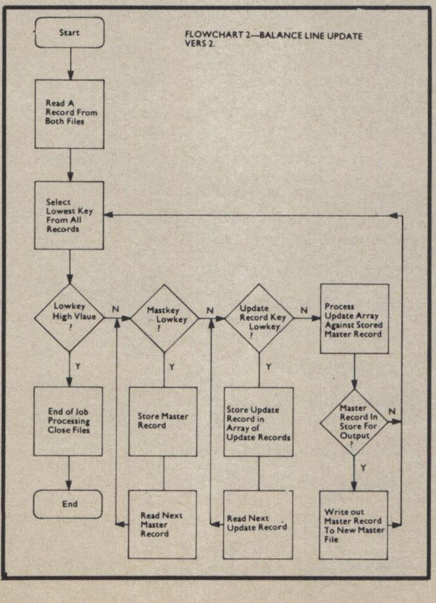
THE VAT REPORT

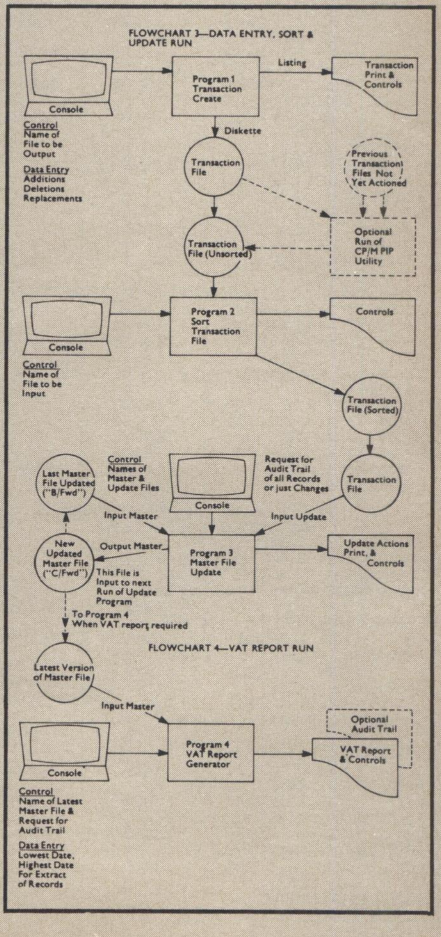
If you have gone this far, very little remains to be done since running the VAT report program is very simple. Most of it has been described in the discussion of program 4.

Before running it, however, make sure you have all your transactions into the

(*continued on next page*)


FLOWCHART 1—BALANCE LINE UPDATE VERS 1.
Start
Read A Record From Both Files
Select Lowest Key From All Records
Low Key High Value ?
N
Y
Mast Key Low Key ?
Update Record Key Low Key ?
Master Record In Store For Output ?
End of Job Processing Close Files
Store Master Record
Process Update Record Against Master Record in Store
Write Out Master Record To New Master File
End
Read Next Master Record
Read Next Update Record


FLOWCHART 2—BALANCE LINE UPDATE VERS 2.
Start
Read A Record From Both Files
Select Lowest Key From All Records
Lowkey High Vlaue ?
N
Y
Mastkey Lowkey ?
Update Record Key Lowkey ?
Process Update Array Against Stored Master Record
Master Record In Store For Output ?
End of Job Processing Close Files
Store Master Record
Store Update Record in Array of Update Records
End
Read Next Master Record
Read Next Update Record
Write out Master Record To New Master File


FLOWCHART 3—DATA ENTRY, SORT & UPDATE RUN

FLOWCHART 4—VAT REPORT RUN
Console
Program 1 Transaction Create
Listing
Transaction Print & Controls
Control
Name of File to be Output
Data Entry
Additions Deletions Replacements
Diskette
Transaction File
Previous Transaction Files Not Yet Actioned
Optional Run of CP/M PIP Utility
Transaction File (Unsorted)
Console
Program 2 Sort Transaction File
Controls
Control
Name of File to be Input
Transaction File (Sorted)
Last Master File Updated ("B/Fwd")
Control
Names of Master & Update Files
Console
Request for Audit Trail of all Records or just Changes
Transaction File
Input Master
Input Update
New Updated Master File ("C/Fwd")
Output Master
Program 3 Master File Update
Update Actions Print, & Controls
This File is Input to next Run of Update Program
To Program 4 When VAT report required
Latest Version of Master File
Input Master
Optional Audit Trail
Console
Program 4 VAT Report Generator
VAT Report & Controls
Control
Name of Latest Master File & Request for Audit Trail
Data Entry
Lowest Date, Highest Date For Extract of Records

(*continued from previous page*)

system, and please run against the latest version of the master file. The listing produced in sample run 4 is that from the VAT report program run against the master file updated in run 3.

CONTROLS, AUDIT TRAIL & SECURITY

The purpose of an audit trail and controls is to ensure that data entering this system can be accounted for throughout its life in the system, and, once in, stays in. This means that some method of ensuring that files do not lose their data, or that records do not change their value, must exist.

I file all my runs in chronological order in a ring binder, providing the most recent runs ready-to-hand when I am running. When an update cycle is about to be run, I look up the name of the last master file updated, and once the run is complete, check that the totals shown for that file on the run just completed are the same as those shown when it was created.

I also check that within the run, the totals for transaction file created agree, firstly between the output at program one and the input at program two, secondly, between input and output within program two; and thirdly, between the output at program two and the input at program three. This ensures that none of the programs has lost any data.

The last reconciliation to be performed is that occurring during the update program; at the end of the program extensive totals are printed. The "effective results' should be added to the number of input records shown as coming in on the input master file as follows:

Number of master records input + EFF additions—EFF deletes = number of master records output.

If this equation is not equal, an error has occurred, and the audit trail should be examined for clues as to the error. The audit trails are provided so that you can trace a transaction through the system until it appears on the listing in the VAT report. This ensures that you can always be

(*continued on next page*)

SAMPLE RUN 1

Special startup run of program one to create 'NULL' file for input to first run of update program.

```
78/08/12   VAT TRANSACTION CREATE PROGRAM
                                          PAGE 1
  END OF JOB
FILE CREATED NAMED NULL.DAT
  TOTAL RECS CREATED = 0
ADDITIONS = 0  DELETIONS = 0  REPLACEMENTS
= 0
  TOTAL GROSS VALUE     =   0·00
  TOTAL NETT VALUE      =   0·00
  HASH TOTAL OF VAT     =   0·00
```

SAMPLE RUN 2

First sample run of transaction create, sort and update cycle.

```
78/08/12   VAT TRANSACTION CREATE PROGRAM
                                          PAGE 1
78/04/01  00001  0 A  PRINTING INV 203
I  P   8.00        30.13       27.89       2.24
78/04/03  79014  0 A  WILLIS INV 22546
I  P   8.00        28·50       26.39       2·11
78/04/05  00001  0 A  TEST 2
I  P     8.00      28.51       26.39       2.12
78/04/06  00002  0 D  ERROR DELETE
X  X     0.00       0.00        0.00       0.00
78/04/01  00002  0 R  ERROR CHANGE
O  S     8.00      100.0       92.59       7.41
78/05/24  78023  1 A  TEST SALE
O  S     8.00     108.00      100.00       8.00
78/05/30  78022  0 A  TEST IMPORT
O  2     8.00    1590.00     1472.22     117.78
78/06/02  00001  0 A  TEST SALE 2
O  S     8.00    1296.54     1200.00      96.04
78/06/02  00001  1 A  TEST SALE 3
O  S    12.50     112.50      100.00      12.50

  END OF JOB
FILE CREATED NAMED VATES01.UNS
  TOTAL RECS CREATED = 9
ADDITIONS = 7   DELETIONS = 1
REPLACEMENTS = 1
  TOTAL GROSS VALUE    = 3294.18
  TOTAL NETT VALUE     = 3045.98
  HASH TOTAL OF VAT    =  248.20

  VAT TRANSACTION SORT RUN ON 78/08/12
TOTAL RECORDS INPUT            9.00
HASH TOTAL INP GROSS        3294.18
HASH TOTAL INP NET          3045.98
HASH TOTAL INP VAT           248.20
FILE INPUT AS VATES01.UNS NOW RENAMED TO
VATES01.OLD
TOTAL RECORDS OUTPUT           9.00
HASH TOTAL OUT GROSS        3294.18
HASH TOTAL OUT NET          3045.98
HASH TOTAL OUT VAT           248.20
FILE NOW SORTED & OUTPUT AS VATES01.NEW
END OF JOB
     VAT UPDATE PROGRAM RUN ON 78/08/12
           FULL AUDIT TRAIL REPORT
MASTER IN     = NULL.DAT
MASTER OUT  = VATMAS01.MAS
UPDATE IN     = VATES01.NEW
A780401000010APRINTING INV 203
IP         800        3013        2789          224
E780401000020RERROR CHANGE
OS         800       10000        9259          741***
A780403780140AWILLIS INV 22546
IP         800        2850        2639          211
A780405000010ATEST 2
IP         800        2851      2639            212
E780406000020DERROR DELETE
XX           0           0           0            0***
A780524780231ATEST SALE
OS         800       10800       10000          800
A78053078022ATEST IMPORT
O2         800      159000      147222        11778
A780602000010ATEST SALE 2
OS         800      129654      120050         9604
A780602000011ATEST SALE 3
OS        1250       11250       10000         1250

INPUT MASTER FILE
NO OF RECORDS     0
HASH TOTAL CROSS  VALUE    0.00
HASH TOTAL NET      VALUE    0.00
HASH TOTAL VAT      VALUE    0.00

INPUT UPDATE FILE
NO OF RECORDS     9
HASH TOTAL GROSS VALUE    3294.18
HASH TOTAL NET      VALUE    3045.98
HASH TOTAL VAT      VALUE     248.20
ADDS = 7   DELETE = 1   CHANGES = 1

OUTPUT MASTER FILE
NO OF RECORDS     7
HASH TOTAL GROSS VALUE    3194.18
HASH TOTAL NET      VALUE    2953.39
HASH TOTAL VAT      VALUE     240.79
EFFECTIVE RESULTS TO MASTER FILE
ADDS = 7   DELETES = 0   CHANGES = 0
ERRORS =      2
```

SAMPLE RUN 3

Second sample run of transaction create, sort and update cycle.

```
78/08/13   VAT TRANSACTION CREATE PROGRAM
                                          PAGE 1
78/06/02  00001   1 D  DELETE TEST SALE
X  X    0.00          0.00          0.00       0.00
78/04/06  00002   0 A  NEW TEST PURCH
1  P    8.00        162.00        150.00      12.00
78/05/30  78022   0 R  CHANGE IMPORT
0  2    8.00       1472.22       1363.16     109.06
78/05/21  00001   0 A  A TEST SALE 4
0  S    8.00        120.00        111.11       8.89

—END OF JOB
FILE CREATED NAMED VATE502.UNS
  TOTAL RECS CREATED = 4
ADDITIONS = 2   DELETIONS = 1
REPLACEMENTS = 1
  TOTAL GROSS VALUE     = 1754.22
  TOTAL NETT VALUE      = 1624.27
  HASH TOTAL OF VAT     =  129.95

    VAT TRANSACTION SORT RUN ON 78/08/13
TOTAL RECORDS INPUT              4.00
HASH TOTAL INP GROSS          1754.22
HASH TOTAL INP NET            1624.27
HASH TOTAL INP VAT             129.95
FILE INPUT AS VATES02.UNS NOW RENAMED TO
VATES02.OLD
TOTAL RECORDS OUTPUT             4.00
HASH TOTAL OUT GROSS          1754.22
HASH TOTAL OUT NET            1624.27
HASH TOTAL OUT VAT             129.95
FILE NOW SORTED & OUTPUT AS VATES02.NEW
END OF JOB

     VAT UPDATE PROGRAM RUN ON 78/08/13
           FULL AUDIT TRAIL REPORT
MASTER IN     = VATMAS01.MAS
MASTER OUT  = VATMAS02.MAS
UPDATE IN     = VATES02.NEW
U780401000010APRINTING INV 203
IP         800        3013        2789          224
U780403780140AWILLIS INV 22546
IP         800        2850        2639          211
U780405000010ATEST 2
IP         800        2851        2639          212
A780406000020ANEW TEST PURCH
IP         800       16200       15000         1200
A780521000010ATEST SALE 4
OS         800       12000       11111          889
U780524780231ATEST SALE
OS         800       10800       10000          800
- - - - - - - - - - - - - - - - - - - - - - - - - -
O780530780220ATEST IMPORT
O2         800      159000      147222        11778
N780530780220RCHANGE IMPORT
O2         800      147222      136316        10906
- - - - - - - - - - - - - - - - - - - - - - - - - -
'U780602000001ATEST SALE 2
OS         800      129654      120050         9604
D780602000011ATEST SALE 3
OS        1250       11250       10000         1250

INPUT MASTER FILE
NO OF RECORDS              7
HASH TOTAL GROSS VALUE    3194.18
HASH TOTAL NET      VALUE    2953.39
HASH TOTAL VAT      VALUE     240.79

INPUT UPDATE FILE
NO OF RECORDS              4
HASH TOTAL GROSS VALUE    1754.22
HASH TOTAL NET      VALUE    1624.27
HASH TOTAL VAT      VALUE     129.95
ADDS = 2   DELETES = 1   CHANGES = 1

OUTPUT MASTER FILE
NO OF RECORDS              8
HASH TOTAL GROSS  VALUE   3245.90
HASH TOTAL NET      VALUE   3005.44
HASH TOTAL VAT      VALUE   240.46
EFFECTIVE RESULTS TO MASTER FILE
ADDS = 2   DELETES = 1   CHANGES = 1
ERRORS =      0
```

SAMPLE RUN 4

Run of VAT report program, using master file output in sample run 3.

```
78/08/13   VAT REPORT PROGRAM                  PAGE 1
EXTRACT OF RECORDS FROM FILE VATMAS02.MAS
WITH FULL AUDIT TRCIL FOR PERIOD FROM
780401 TO 780631
780401000010APRINTING INV 203
IP         800        3013        2789          224
780403780140AWILLIS INV 22456
IP         800        2850        2639          211
780405000010ATEST 2
IP         800        2851        2639          212
780406000020ANEW TEST PURCH
IP         800       16200       15000         1200
78052100010ATEST SALE 4
OS         800       12000       11111          889
780524780231ATEST SALE
OS         800       10800       10000          800
780530780220RCHANGE IMPORT
O2         800      147222      136316        10906
780602000010ATEST SALE 2
OS         800      129654      120050         9604

PART A OF RETURN
  TAX DUE STANDARD RATE                        112.93
           RATE A                                0.00
           RATE B                                0.00
                                               - - - - -
  TOTAL OUPUT TAX                              112.93
                                               - - - - -
    TAX ON IMPORTS ETC.                        109.06
    TAX UNDERDECLARED & NOTIFIED                 0.00
    TAX UNDERDECLARED ON RETUNS                  0.00
                                               - - - - -
  TOTAL TAX DUE***                             221.99
                                               - - - - -

  TAX DEDUCTIBLE (PURCHASES)                    18.47
       TAX OVERPAID & NOTIFIED                   0.00
       TAX OVERPAID ON RETURNS                   0.00
                                               - - - - -
  TOTAL TAX DEDUCTIBLE                          18.47
                                               - - - - -
  TAX PAYABLE                                  203.52
                                               - - - - -

PART B OF RETURN
  VALUE OF OUTPUTS TAXABLE                    1411.61
  VALUE OF EXPORTS                               0.00
  VALUE OF ZERO RATED OUTPUTS                    0.00
  TOTAL VALUE OF TAXABLE OUTPUTS              1411.61
  VALUE OF EXEMPT OUTPUTS                        0.00
  TOTAL VALUE OF OUTPUTS***                   1411.61

  TOTAL VALUE OF INPUTS****                    230.67

  END OF JOB
  TOTAL RECORDS READ = 8
TOTAL GROSS VALUE            3245.90
TOTAL NETT VALUE             3005.44
HASH TOTAL OF VAT             240.46
```

(continued from previous page)

certain after the event if an invoice has been accounted for or not.

If you choose, however, not to perform the reconciliations, nothing will prevent you running the system, but I advise you against it.

The next most important thing is security back-up. This entails you making copies through the use of your operating system of the most recent versions of all your files. After each run of the update cycle, I copy the sorted transaction file and the input master file onto another diskette, which I then remove to another part of the house.

At every VAT period end I take this diskette to the bank for safe keeping. This ensures that the most loss I can incur is one whole VAT period if the house burns down.

The most obvious enhancement is to provide an automatic method of allocating file names, and to input the latest version of master files during update runs. Within CP/M and TDL Basic this is possible, but it is system-dependent.

LISTING 3—VAT SYSTEM SUITE MASTER FILE UPDATE PROGRAM

```
10 REM VAT MASTER FILE SEQUENTIAL UPDATE
20 REM COPYRIGHT (1978) XITAN SYSTEMS
30 REM WRITTEN BY G.C.LYNCH
40 REM N$ = MASTER FILE INPUT RECORD
50 REM SM$ = STORED MASTER FILE INPUT
RECORD
60 REM U$ = UPDATE FILE INPUT TECORD
70 REM US$ = UPDATE RECORD STORE ARRAY
(ERROR IF >1 RECORD SAME KEY)
80 REM OM$ = OUTPUT MASTER FILE RECORD
90 REM LK$ = LOWEST KEY FROM TWO FILES
100 REM MK$ = CURRENT MASTER FILE RECORD
KEY
110 REM UK$ = CURRENT UPDATE FILE RECORD
KEY
120 REM CLEAR FOR 3 FILES AT ONCE (TDL BASIC
FEATURE)
130 CLEAR 2000, 3
140 DS$ = STRINGS("- ", 32)
150 INPUT "ENTER DATE OF RUN (DD/MM/YY)
";D$
160 INPUT "ENTER INPUT MASTER FILE NAME IN
FULL ";FI$
170 INPUT "ENTER OUTPUT MASTER FILE NAME IN
FULL "FO$
180 INPUT "ENTER UPDATE FILE NAME IN FULL
";FU$
190 INPUT "DO YOU WANT UNCHANGED
RECORDS PRINTED (Y/N) ";A$
200 IF LEFT$(A$, 1) = "Y" THEN ST$ = "FULL
AUDIT TRAIL" ELSE ST$ = "CHANGES ONLY"
210 PRINT "MASTER IN = ";FI$
220 PRINT "MASTER OUT = ";FO$
230 PRINT "UPDATE IN = ";FU$
240 INPUT "ARE THEY CORRECT (Y/N) ";Z$
250 IF LEFT$(Z$, 1)<>"Y" THEN 150
260 PRINT #2: PRINT #2
270 PRINT #2,"      VAT UPDATE PROGRAM RUN
ON ";D$
280 PRINT #2, TAB(25);ST$;" REPORT"
290 PRINT #2
300 PRINT #2, "MASTER IN     = ";FI$
310 PRINT #2, "MASTER OUT    = ";FO$
320 PRINT #2, "UPDATE IN     = ";FU$
330 PRINT #2
340 SI$=" 'LLLLLLLLLLLLLL #####"
350 S2$="'LLLLLLLLLLLLLLLLLLLL
#######.##"
360 OPEN #5, "I",FI$
370 OPEN #6, "I",FU$
380 OPEN #7, "O", FO$
390 DIM US$(4)
420 REM NOW GOTO MASTER FILE READ
ROUTINE
430 GOSUB 1830
440 REM NOW GOTO UPDATE FILE READ ROUTINE
450 GOSUB 2010
460 REM UPDATE PROGRAM BALANCE LINE
MAIN FLOW
470 REM SELECT LOWEST KEY
480 IF MKS < UK$ THEN LK$ = MK$
ELSE LK$ = UK$
490 REM CHECK FOR EOF ON BOTH FILES
500 REM IF LOWEST KEY ALL 9'S THEN END OF
JOB
510 IF LEFT$(LK$,11)="99999999999" THEN 2360
530 REM MASTER FILE INNER LOOP
550 IF MK$<> LK$ THEN 660
560 REM STORE MASTER RECORD
570 SM$ = M$
580 SK$ = LEFT$(SK$, 12)
590 REM NOW GOTO MASTER FILE READ ROUTINE
600 GOSUB 1830
610 IF MKS = SK$ THEN 2230
620 GOTO 550
640 REM END OF MASTER FILE INNER LOOP
670 REM UPDATE FILE INNER LOOP
690 IF UK$<>THEN 780
700 K=K+1
710 US$(K)=U$
720 REM NOW GOTO UPDATE FILE READ ROUTINE
730 GOSUB 2010
740 GOTO 690
760 REM END OF INNER LOOP
790 REM MASTER FILE VS UPDATE FILE ROUTINES
810 REM FIRST WHERE NO UPDATE WITH SAME
KEY EXISTS
820 REM I E MASTER RECORD TO BE PROCESSED
UNCHANGED
840 IF K>0 THEN 990
850 OM$=SM$
860 REM NOW GOTO MASTER FILE O/P ROUTINE
870 GOSUB 1740
880 REM NOW PRINT IT IF FULL ABOUT TRAIL
REQUESTED
890 IF LEFT$(AS,1) = "Y" THEN PRINT #2, "U";
OM$
910 REM MAKE MASTER FILE RECORD NULL
920 SM$=""
930 REM GO BACK TO LOWEST KEY ROUTINE
940 GOTO 480
970 REM ROUTINE WHERE MASTER KEY HIGH AND
UPDATE IS ADD
980 REM I E UPDATE RECORD IS TO BE ADDED TO
MASTER FILE
990 IF K>1 THEN 1630
1000 IF SM$="" AND MID$(US$(K), 13, 1)="A"
THEN 1020
1010 GOTO 1140
1020 REM NOW INTO REAL ADD TO MASTER
1030 OM$=US$(K)
1040 AC=AC+1
1050 REM NOW GOTO MASTER FILE O/P ROUTINE
1060 GOSUB 1740
1070 PRINT #2, "A";OM$
1080 US$(1)=""
1090 K=0
1110 REM NOW GOTO LOWEST KEY ROUTINE,
ALL FINISHED HERE
1120 GOTO 480
1150 REM TEST FOR NO MATCH BUT REPLACE
OR DELETE
1160 REM I E THIS IS AN ERROR CONDITION
1170 IF SM$=""THEN
IF MID$(US$(K), 13, 1)="D" OR MID$(US$(K), 13, 1)=
                         "R"
          THEN PRINT #2, "E"+US$(K);" *** "
                    : E=E+1:K =0GOTO 480
1190 REM NOW WHERE MASTER FILE IN STORE
BUT UPDATE IS ADD
1200 REM WHICH IS AN ERROR CONDITION
1210 IF SM$<>"" AND MID$(US$(K), 13, 1)="A"
THEN 1230
1220 GOTO 1350
1230 PRINT #2, "E";US$(K);" *** "
1240 OM$=SM$
1250 E=E+1
1260 US$(K)=""
1270 K=0
1280 REM NOW PRINT MASTER RECORD IF FULL
AUDIT TRAIL REQUESTED
1290 IF LEFT$(AS, 1) = "Y" THEN PRINT #2, "U";
OM$
1300 REM NOW GO TO MASTER FILE O/P ROUTINE
1310 GOSUB 1740
1330 GOTO 480
1350 REM NOW WHERE MASTER EXISTS BUT
UPDATE IS DELETE
1360 IF SM$<>="" AND MID$(US$(K), 13, 1)="D"
THEN 1380
1370 GOTO 1470
1390 D=D+1
1400 PRINT #2,"D";SM$
1410 US$(K)=""    ..
1420 K=0
1430 SM$=""
1450 GOTO 480
1470 REM NOW WHERE MASTER EXISTS BUT
UPDATE IS REPLACE
1500 IF MID$(US$(K), 13, 1)<>"R" THEN 1630
1510 R=R+1
1520 PRINT #2, DS$
1530 PRINT #2, "O";SM$
1540 PRINT #2, "N";US$(K)
1550 PRINT #2, DS$
1560 OM$=US$(K)
1570 GOSUB 1740
1580 US$(K)=""
1592 K=0
1610 GOTO 480
1630 REM UPDATE ARRAY IN ERROR
1650 FOR J=1 TO K
1660 PRINT #2, "E";US$(K);" *** "
1670 US$(K)=""
1680 NEXT J
1690 REM THIS UNCHANGED RECORD ALWAYS
PRINTED TO SHOW WHY ERROR EXISTS
1700 IF SM$<>"" THEN OM$=SM$:GOSUB 1740:
PRINT #2, "U"; OM$:SM$=""
1720 K=0
1730 GOTO 480
1740 REM MASTER FILE OUTPUT ROUTINE
1750 REM OUTPUTS FROM OM$ AND ACCUMS
TOTALS
1760 QR=QR+1
1770 QG=QG+ VAL(MID$(OM$, 44, 7))
1782 QN=QN+ VAL(MDI$(OM$, 51, 7))
1790 QV=QV+ VAL(MID$(OM$, 58, 6))
1800 PRINT #7, OM$
1810 RETURN
1840 REM MASTER FILE INPUT ROUTINE
1850 REM DERIVES KEY AND ACCUMS HASH
TOTALS
1860 IF EM=1 THEN RETURN
1870 ON EOF #5 GOTO 1960
1880 INPUT #5, M$
1890 MK$=LEFT$(M$, 12)
1900 IR=IR+1
1910 IG=IG+ VAL(MID$(M$, 44, 7))
1920 IN=IN+ VAL(MID$(M$, 51, 7))
1930 IV=IV+ VAL(MID$(M$, 58, 6))
1940 RETURN
1960 MK$=STRINGS("9", 12)
1970 M$=MK$
1980 EM=1
1990 RETURN
2010 REM UPDATE FILE INPUT ROUTINE
2020 REM DERIVES KEY AND ACCUMS HASH
TOTALS
2030 IF EU=1 THEN RETURN
2040 ON EOF #6 GOTO 2160
2050 INPUT #6, U$
2060 UK$=LEFT$(U$, 12)
2070 UR=UR+1
2080 UG=UG+1 VAL(MID$(U$, 44, 7))
2090 UN=UN+ VAL(MID$(U$, 51, 7))
2100 UV=UV+1 VAL(MID$(U$, S8, 6))
2110 IF MIDS(U$, 13, 1)="A" THEN UA=UA+1
2120 IF MIDS(U$, 13, 1)="D" THEN UD=UD+1
2130 IF MIDS(U$, 13, 1)="R" THEN UI=UI+1
2140 RETURN
2160 UK$=STRING ("9", 12)
2170 U$=UK$
2180 EU=1
2190 RETURN
2230 REM DUPLICATE KEYS ERROR ON MASTER FILE
2250 PRINT #2, "DUPLICATE KEYS ON MASTER
FILE"
2260 PRINT #2, "RECORD 1"
2270 PRINT #2, SM$
2280 PRINT #2, "RECORD 2"
2290 PRINT #2, M$
2300 PRINT #2
2310 PRINT #2, " * * * JOB ABORTED * * *"
2320 PRINT " * * * JOB ABORTED * * * DUPLICATE
KEYS * *"
2330 CLOSE
2340 END
2370 REM END OF JOB ROUTINES
2380 REM CLOSE FILES
2390 CLOSE #5, #6, #7
2400 REM NOW OUTPUT CONTROLS
2410 PRINT #2:PRINT #2
2420 PR=IR:PG=IG:PN=IN:PV=IV
2430 PRINT #2, "INPUT MASTER FILE"
2440 GOSUB 2600
2450 PRINT #2
2460 PR=UR:PG=UG:PN=UN:PV=UV
2470 PRINT #2, "INPUG UPDATE FILE"
2480 GOSUB 2600
2490 PRINT #2, "ADDS=";UA;" DELETES=";UD;"
CHANGES=";UI
2500 PRINT #2
2510 PR=QR:PG=QG:PN=QN:PV=QV
2520 PRINT #2, "OUTPUT MASTER FILE"
2530 GOSUB 2600
2540 PRINT #2
2550 PRINT #2, "EFFECTIVE RESULTS TO MASTER
FILE"
2560 PRINT #2, "ADDS =";AC"; DELETES =";D;"
CHANGES =";R
2570 PRINT #2, "ERRORS =", E:PRINT #2
2580 CLOSE
2590 END
2600 REM PRINT CONTROLS SUBROUTINE
2610 PRINT #2, USING SI$;"NO OF RECORDS"  PR
2620 PRINT #2, USING S2$;"HASH TOTAL GROSS
VALUE", PG/100
2630 PRINT #2, USING S2$;"HASH TOTAL NET
VALUE" PN/100
2640 PRINT #2  USING S2$;"HASH TOTAL VAT
VALUE",PV/100
2650 RETURN
```

LISTING 4—VAT SYSTEM SUITE MAIN VAT RETURN REPORT PROGRAM

```
10 REM VAT REPORT PROGRAM
20 REM COPYRIGHT 1978 (C) XITAN SYSTEMS
30 REM WRITTEN BY G.C.LYNCH MARCH 1978
60 REM READS VAT MASTER (OR TRANS) FILE &
CREATES
70 REM REPORT IN VAT RETURN FORMAT
100 REM VARIABLES USED
120 REM DL$ = LOW DATE FOR SELECT OFF FILE
130 REM DH$ = HIGH DATE
140 REM D$ = DATE OF REPORT (YY/MM/DD)
150 REM F$ = FILENAME TO BE READ
160 REM V(17 4) = ARRAY OF VAT RETURN TOTALS
170 REM 1ST SUBSCRIPT CORRELATES TO
NUMBER USED ON VAT RETURN
180 REM 2ND SUBSCRIPT 1=STANDATD RATE,
2=RATE A, 3=RATE B  4=TOTAL
190 REMS DR$ = TAX DATE ON RECORD
200 REM A$ = AUDIT TRAIL REQUEST INDICATOR
(Y=REQUESTED)
280 REM START OF PROGRAM
290 CLEAR 3000, 1
300 DIM V(17, 4)
310 ER=0
320 SI$="#######.##"
```

(continued on next page)

(continued from previous page)

```
330 S3$=" 'LLLLLLLLLLLLLLLLLLLLLLLLL
######.##"
340 FOR J= 1 TO 17
350 FOR I= 1 TO 4
360 V(J  I)=0
370 NEXT I
380 NEXT J
390 REM INIT END OF FILE FLAG
400 E$="0"
410 PRINT:PRINT "      VAT REPORT PROGRAM    "
420 PRINT
430 INPUT" ENTER TODAYS DATE (YY/MM/DD)
"; D$
440 INPUT "DO YOU WANT FULL AUDIT TRAIL
OF RECORDS EXTRACTED (Y/N) ";A$
450 A$=LEFT$(A$, 1)
460 IF A$ = "Y" THEN RT$ = "FULL AUDIT TRAIL"
                 ELSE RT$ = "NO AUDIT TRAIL"
470 ER=0
480 PRINT:INPUT"ENTER LOWEST DATE,
HIGHEST DATE (YYMMDD) FOR REPORT"
;DL$, DH$
490 CD$=DL$
500 GOSUB 2040:IF ER=1 THEN 470
510 CD$=DH$
520 GOSUB 2040:IF ER=1 THEN 470
530 PRINT :INPUT"FILENAME TO BE USED ";F$
540 PRINT
560 REM OPEN FILE AS UNIT 5, NOMINAL
RECORDSIZE OF 64
570 OPEN #5, "I". F$, 64
590 REM PRINT HEADINGS
600 GOSUB 2380
620 READ RECORD AND ENTER MAIN PROGRAM
LOOP
630 REM GOTO READ RECORD ROUTINE
640 GOSUB 2140
660 REM IF EOF GOTO REPORT PORTION
670 IF E$="1" THEN 1120
690 REM IGNORE IF TAX DATE NOT IN EXTRACT
DATE RANGE
700 IF DR$<DL$ THEN 620
710 IF DR$>DH$ THEN 620
720 REM IF EXTRACTED GOTO PRINT ROUTINE
730 GOSUB 2470
740 IF V1 = 800 THEN
      R = 1:V1 = 8.00
750 IF V1 = 1250 THEN
      R = 2:V1 = 12.50
760 IF V1 = 0 THEN
      R = 0:V1 = 0.00
770 REM REMEMBER THAT ALL VALUES ARE IN
PENCE (NO DECIMAL PLACES)
790 REM DO NOT ADD INTO ARRAY IF VAT=0
800 IF V1 = 0 THEN 940
820 REM ADD VAT INTO APPROPRIATE ARRAY
ELEMENT FOR PART A OF RETURN
830 IF ST$="S" THEN ST$="1"
840 IF ST$="P" THEN ST$="6"
850 IF ST$="E" THEN 940
860 IF ST$="0" THEN 940
870 IF ST$<"1" OR ST$>"8" THEN 940
880 REM SET UP VAT ARRAY SUBSCRIPT
890 ST=VAL(ST$)
900 V(ST, R)=V(ST, R)+V2
930 REM ADD NETT VALUE INTO APPROPRIATE
ELEMENT FOR PART B OF RETURN
940 REM THIS IS COMPLICATED BY THE VAT RULES
FOR FILLING OUT RETURN
960 REM EXPORT SALE FIRST
970 IF ST$="0" THEN V(12, 1)=C(12, 1)+N
980 REM NORMAL PURCHASES
990 IF ST$="6" THEN V(17, 1)=V(17, 1)+N
1000 REM SALES, TWO DIFFERENT DEPENDING ON
ZERO RATED OR NOT OUTPUTS
1010 IF ST$="1" THEN
        IF V1=0 THEN V(13, 1)=V(13,1)+N
                ELSE V(11, 1)=V(11, 1)+N
1020 REM EXEMPT OUTPUTS(SALES), SEE NOTE ON
PART D OF RETURN
1030 IF ST$="E" THEN V(15, 1)=V(15, 1)+N
1060 REM ALL ACCUMULATION DONE
1080 REM GET NEXT RECORD
1090 GOTO 620
1120 REM MAIN REPORT ROUTINE
1140 REM TOTAL UP ARRAY VALUES
1160 FOR J= 1 TO 17
1170 V(J, 4)=V(J, 1)+V(J, 2)+V(J, 3)
1180 NEXT J
1190 IF LC>35 THEN GOSUB 2380
1200 PRINT #2
1210 PRINT #2, "PART A OF RETURN"
1220 PRINT #2, TAB(10); "TAX DUE STANDARD
RATE"; TAB(60);
1230 PRINT #2, USING S1$;V(1, 1)/100
1240 PRINT #2, TAB(10);"        RATE A";TAB (60);
1250 PRINT #2, USING S1$;V(1, 2)/100
1260 PRINT #2, TAB(10);"        RATE B";TAB(60);
1270 PRINT #2, USING S1$;V(1, 3)/100
1280 PRINT #2, TAB(68);"----------"
1290 PRINT #2, TAB(10;"TOTAL OUTPUT TAX";
TAB(60);
1300 PRINT #2, USING S1$;V(a, 4)/100
1310 PRINT #2, TAB(60s;"----------"
1320 PRINT #2, TAB(10)"   TAX ON IMPORTS
ETC.";TAB(60);
1330 PRINT #2, USING S1$;V(2, 4)/100
1340 PRINT #2, TAB(10);"   TAX UNDERDEDLARED
& NOTIFIED";TAB(60);
1350 PRINT #2, USING S1$;V(3, 4)/100
1360 PRINT #2, TAB(10);"   TAX UNDERDECLARED
ON RETURNS";TAB(60);
1370 PRINT #2, USING S1$;V(4, 4)/100
1380 FOR J= 1 TO 4
1390 V(5, 4)=V(S, 4)   V(J, .4)
1400 NEXT J
1410 PRINT #2, TAB(60);"----------"
1420 PRINT #2, TAB(10);"TOTAL TAX DUE***";
TAB (60);
1430 PRINT #2, USING S1$;V(5, 4)/100
1440 PRINT #2, TAB(60);"----------"
1450 PRINT #2
1460 PRINT #2, TAB(10);"TAX DEDUCTIBLE
(PURCHASES)";TAB(60);
1470 PRINT #2, USING S1$;V(6, 4)/100
1480 PRINT #2, TAB(10);"   TAX OVERPAID &
NOTIFIED";TAB(60);
1490 PRINT #2, USING S1$;V(7, 4)/100
1500 PRINT #2, TAB(10);"   TAX OVERPAID ON
RETURNS";TAB(60);
1510 PRINT #2, USING S1$;V(8, 4)/100
1520 PRINT #2, TAB(60);"----------"
1530 PRINT #2, TAB(10);"TOTAL TAX
DEDUCTIBLE"; TAB(60);
1540 FOR J = 6 TO 8
1550 V(9, 4)=V(9, 4)   V(J, 4)
1560 NEXT J
1570 PRINT #2, USING S1$;V(9, 4)/100
1580 PRINT #2, TAB(60);"----------"
1590 V(10, 4)=ABS(V(5, 4)-V(9, 4))
1600 IF V(5, 4)>V(9, 4) THEN S2$="PAYABLE"
ELSE S2$="REPAYABLE"
1610 PRINT #2, TAB(5);"TAX "; S2$;TAB(60);
1620 PRINT #2, USING S1$;V(10, 4)/100
1630 PRINT #2, TAB(60);"----------"
1640 PRINT #2:PRINT #2
1650 REM NOW DO PART B OF RETURN
1670 PRINT #2, "PART B OF RETURN"
1680 PRINT #2, TAB(10);"VALUE OF OUTPUTS
TAXABLE";TAB(60);
1690 PRINT #2, USING S1$;V(11, 4)/100
1700 PRINT #2, TAB(10);"VALUE OF EXPORTS";
TAB(60);
1710 PRINT #2, USING S1$;V(12, 4)/100
1720 PRINT #2, TAB(10);"VALUE OF ZERO RATED
OUTPUTS";TAB(60);
1730 PRINT #2, USING S1$;V(13, 4)/100
1740 PRINT #2, TAB(10);"TOTAL VALUE OF
TAXABLE OUTPUTS";TAB(60);
1750 FOR J= 11 TO 13
1760 V(14, 4)=V(14, 4)+V(J, 4)
1770 NEXT J
1780 PRINT #2, USING S1$;V(14,4)/100
1790 PRINT #2, TAB(10);"VALUE OF EXEMPT
OUTPUTS"; TAB(60);
1800 PRINT #2, USING S1$V(15,4)/100
1810 V(16,4)=V(14, 4)+V(15, 4)
1820 PRINT #2, TAB(10);"TOTAL VALUE OF
OUTPUTS***";TAB(60);
1830 PRINT #2, USING S1$;(V16, 4)/100
1840 PRINT #2
1850 PRINT #2, TAB(10);"TOTAL VALUE OF
INPUTS****";TAB(60);
1860 PRINT #2, USING S1$;V(17, 4)/100
1870 PRINT #2: PRINT #2
1900 REM NOW PRINT END OF JOB CONTROLS
1930 PRINT #2,"   END OF JOB      "
1940 PRINT #2,"—TOTAL RECORDS READ = ";TR
1950 PRINT #2, USING S3$;"TOTAL GROSS VALUE",
TG/100
1960 PRINT #2, USING S2$;"TOTAL NETT VALUE",
TN/100
1970 PRINT #2, USING S3$;"HASH TOTAL OF VAT",
TV/100
1980 PRINT #2: PRINT #2
2030 END
2050 REM ROUTINE TO VALIDATE DATES GIVEN
INY YMMDD FORMAT
2070 M=VAL(MID$(OD$, 3, 2))
2080 D=VAL(MID$(CD$, 5, 2))
2090 IF M<1 OR M>12
        THEN PRINT"ERROR IN MONTH OF
        OF";CD$;ER=1:RETRUN
2100 IF D<1 OR D>31
        THEN PRINT "ERROR IN DAY OF"; CD$:
        ER=1:RETURN
2110 ER=0:RETURN
2140 REM READ RECORD ROUTINE
2150 REM ALSO DISSECTS RECORD & TOTALS UP
HASH TOTALS
2160 ON EOF #5 GOTO 2360
2170 INPUT #5, Z$
2190 TR=TR+1
2200 I$=MID$(Z$, 37, 1)
2210 ST$=MID$(Z$, 38, 1)
2220 V1=VAL(MID$(Z$, 39, 5))
2230 G=VAL(MID$(Z$, 44, 7))
2240 N=VAL(MID$(Z$, 51, 7))
2250 V2=VAL(MID$(Z$, 58, 6))
2260 DR$=MID$(Z$, 1, 6)
2280 REM NOW DO HASH TOTALS
2290 TG=TG+G
2300 TN=TN+N
2310 TV=TV+V2
2330 REM NOW RETURN
2340 RETURN
2350 REM EOF REACHED
2360 E$#"1":RETURN
2380 REM PAGE HEASINGS ROUTINE
2390 PRINT #2:PRINT #2: PRINT #2
2400 PC=PC+1
2410 PRINT #2, TAB(10);D$;TAB(24);"VAT REPORT
PROGRAM";TAB(60);"PAGE ";PC
2420 PRINT #2, "   EXTRACT OF RECORDS FROM
FILE ";F$;"   WITH   ';RT$
2430 PRINT #2,"   FOR PERIOD FROM   ";DL$;
"   TO   ";DH$
2440 PRINT #2, PRINT #2
2450 LC=10
2460 RETURN
2470 REM LINE PRINT ROUTINE, ONLY IF
REQUESTED
2480 IF A$<> "Y" THEN RETURN
2490 IF LC > 66 THEN GOSUB 2380
2500 LC=LC+1
2510 PRINT #2, Z$
2520 RETURN
```

Kim projects

THIS is the first of what I hope will be a regular series of articles for users of 6502-based machines with particular emphasis on the Kim 1, but also applicable to the Pet and Vim 1. The aim is to provide readers who have, or are thinking of purchasing, one of those machines with a source of applications ideas.

The ideas I hope will go some way towards answering the question "Now I have a computer what do I do with it?". Applications ideas under development include a variety of low-cost projects which should be of interest to hobbyists and a large number of people in industry and education.

This month we introduce the Kim to the analogue world and detail how you can use it to build a digital voltmeter and a storage oscilloscope.

Conversion

If, like me, you have embedded hundreds of LEMs into the moon's surface, you might be forgiven for asking if there was a better way of inputting your desired engine thrust other than via the keyboard. Where computing is concerned, there is invariably a better way. One of the prime options in this instance would be a rotary or slider control for the thrust control. As the potentiometer control produces an infinitely-variable voltage and the microprocessor works on discrete binary numbers, some form of conversion between the two must take place.

This conversion, analogue to digital, is a most useful technique for setting data into a microprocessor. There are many physical quantities analogue by nature which can be converted easily into a voltage. For instance, thermistor probes for temperature, photocells for light intensity, and microphones for sound.

Versatile

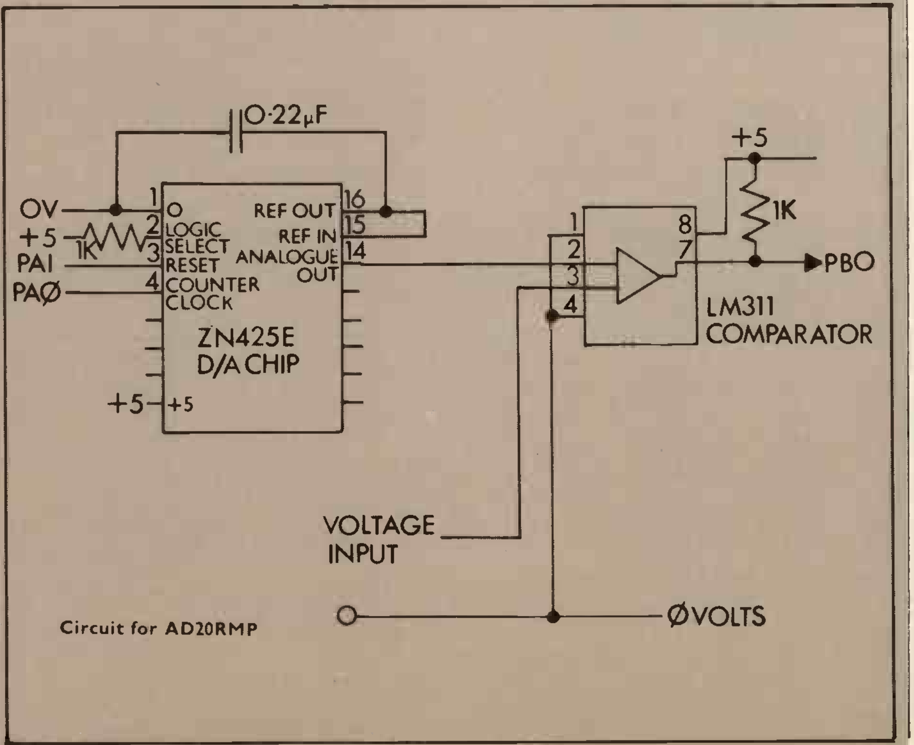
The simplest, and cheapest, analogue-to digital converters are based on the reverse process, digital-to-analogue conversion. The devices take a binary number from, say, the 6230 parallel output port of a Kim 1 and produce a proportional voltage. D/A converters used to be constructed from vast numbers of expensive, high-precision components. Now a D/A converter of respectable performance can be obtained for a few pounds in a 16-pin d.i.l package, the ZN425E. The only other component required is a voltage comparator.

In essence, these A/D conversion algorithms require us to guess about the unknown voltage. If our guess is too low, then the output of the comparator is '0' otherwise it is '1'. If the guess was too low, the next one should be higher, or lower if the last one was too high. This can be repeated until the desired accuracy is achieved.

The ZN425E is a versatile device; in addition to the D/A converter it also contains an eight-bit binary counter. If the LOGIC SELECT (pin 2) is held high (to

(continued on next page)

Figure I



Circuit for AD20RMP

(continued from previous page)

+5v.) each pulse to the CLOCK input (pin 4) will advance the eight-bit counter by one. The ANALOGUE OUT output (pin 14) will increase by about 8·52mV. When the counter reaches 255 the converter produces its highest output of 2·18V; the counter then resets to zero.

By clocking continuously the ZH425E, a staircase voltage waveform of 256 steps is generated. Moreover, the counter and output may be reset to zero at any time by applying a low signal to the reset pin (3).

Inspiration helps

Since there are only two integrated circuits and a minimal number of other components required for A/D converters, there is no need for any elaborate circuit construction. All the prototyping was done on 'u-Dec' breadboards. They look like slabs of plastic with a matrix of holes on the surface. Underneath the holes, through which components, leads and connecting wires may be inserted, are connecting strips.

Each strip connects about five holes together and this is marked clearly on the surface. Integrated circuits can be inserted into the breadboard slab using a special carrier for a d.i.l. socket. Remember that, viewed from the top—i.e., with the pins sticking in something—pin 1 is in the top left-hand corner and that pins are counted anti-clockwise. There is a little dimple to mark pin 1 or a groove—or deeper groove if both ends are cut—to mark the pin 1 and top of the device.

It is next a matter of inserting the ICs into the breadboard, using the carriers. Locate pin 1 and connect wires and components from each pin to their destinations according to the circuit digaram. A degree of method, rather than inspiration or madness, helps here. Electronic circuits are curiously humourless when made with missing or misplaced connections. Normally I reserve the topmost row of holes for the +5 v. supply and the bottom row for the 0V supply. In those particular rows the strips join all the holes.

Ribbon cable was used to connect the user outputs on the Kim 1 to these circuits; the colour coding can be used to good advantage and it isn't so prone to tangling as many individual lengths of wire. Power was provided from the Kim 1 5v line.

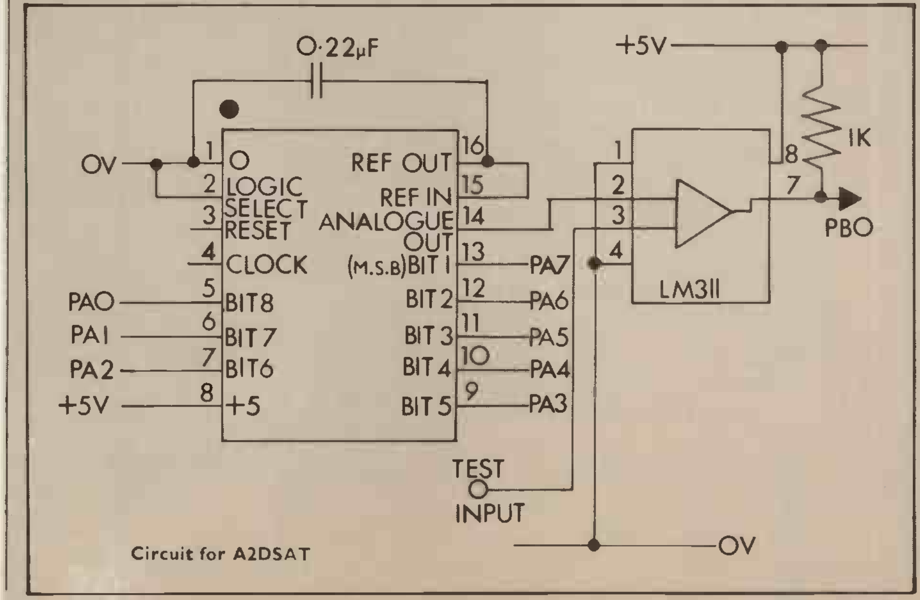
There is no reason why these circuits should not be constructed in any other way. Veroboard wire-wrap and printed circuit construction all have advantage and disadvantages. The disadvantage of the breadboard is that it is expensive initially. It does, however, leave the components unsoldered and both the components and the board are reversible. It is also rather easy to disturb the wiring on the breadboard but solderable boards of the same connector configuration (BLOBBOARDS) are available to make fully-tested circuits permanent.

Sub-routines

The program listing is given as a complete package to drive and use the circuits in figs. 1 and 2. In each case the main work the microcomputer will have to perform is coded as subroutines—i.e., SETUP, A2DRMP, A2DSAT, RECORD and PLAY. The program execution must not start at the beginning of a subroutine, since if it did items would be removed from the stack by the return from subroutine instruction (RTS) which were not put there initially. Smaller programs are written to call the various subroutine

(continued on next page)

Figure 2



Circuit for A2DSAT

(continued from previous page)

'building-blocks' in the correct sequence —TEST, DVM1, DVM2 and SCOPE.

A2DRMP and A2DSAT use the same electronic components in two distinct ways. Either subroutine may be called from any program in much the same way as GETKEY or any other input routine. An early victim of such as alteration might well be that moonlander program.

Freezing

Since they convert a voltage into a binary number, an obvious application for these routines is a digital voltmeter. The unknown voltage is fed into the comparator and the voltage is directly displayed on the Kim 1 LED displays. DECMAL is a binary-to-binary-coded decimal (BCD) routine so that the result may be displayed in decimal ıather than hex by the Kim 1 monitor routine SCANS. By adding a suitable input attenuator network of resistors, a three-range (0–2·55, 0–25·5 and 0–255v) instrument can be made with very few extra electronic components or program instructions.

The other 'ready-made application' here simulates a storage oscilloscope—SCOPE. It is often most useful to be able to freeze a portion of an oscilloscope trace so that it can be inspected more carefully. The incoming signal waveform is both digitised by the routine and displayed on an oscilloscope. If a feature appears meriting further investigation, a switch is pressed and the program continues to display the last 512 samples of the waveform on the oscilloscope.

The Kim 1 is available from the following mail order suppliers at £149 + 8% VAT:

G.R. Electronics Ltd, 80 Church Road, Newport, Gwent. Tel: Newport 67426.

A. Marshall (London) Ltd, 40–42 Cricklewood Broadway, London. 01-452 0161.

The ZN452E D/A converter chip is available from R.S. Components, Doram, and Commodore Systems Division, 360 Euston Road, London NW1. 01-388 5702.

NEXT MONTH we continue our Kim projects, detailing all the circuits and programs you need to know to build a digital voltmeter and a storage oscilloscope. Plus how a photographer, a gardener, or a games player can use A/D converters.

Programming the structured way

TO MOST people entering computing, programming seems difficult and mysterious. Often it appears to be more of an art than a science, lacking as it does any firm rules and laws, and is thus seen by many as the province of the wild genius rather than the ordinary person, writes **Nick Hampshire.**

These views, although popular and I am afraid promoted sometimes by members of the computer profession, are wrong. Most people are capable of writing long and complex computer programs, provided they know what they want the program to do, and in writing them they employ the fairly simple methodology outlined in this article.

Having learned to program a computer in a language like Basic, the user will soon be writing small 10- or 20-line programs. Within a short time as ambitions grow, he will want to write more complex and interesting programs, requiring perhaps 50 lines.

A 50-line program, however, is not just twice as difficult and complex as a 25-line program; it is probably five or 10 times as complex, and with complexity follows the problem of program errors.

It would thus be simpler if we could divide a long program into a set of short, easily-written, easily-debugged programs. This is one of the first principles of our methodology, based as it is on the ideas of structured programming.

Structured programming has been used for many years by professionals writing large commercial programs. For those people it has a great many advantages; a program can be written by a group of programmers each doing a different section; it also reduces the time taken for debugging.

Amateur advantage

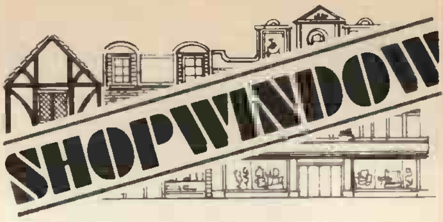
Thus, by using a method which simplifies the structure, the professional can reduce the time taken to write a program and, therefore, the cost. The main advantage for the amateur in using the techniques of structured programming lies in the fact that his program is easier to debug as a result of his being forced to adopt a clear logical structure.

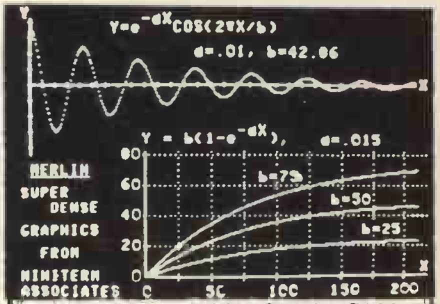
There are four basic rules in the theory of structured programming:

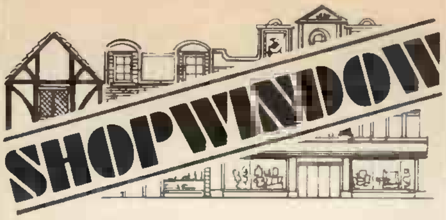
The program is split into a collection of small or subprograms, each capable of being self-standing.

Only a limited set of program "structures" is permitted and each subprogram

(continued on next page)







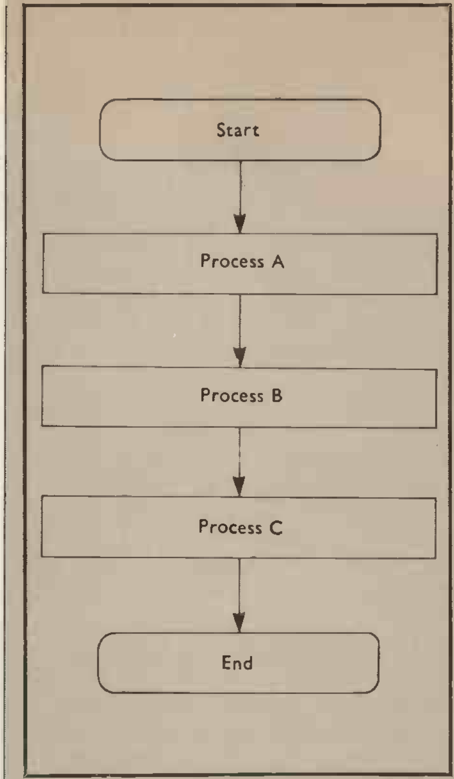


Figure 1

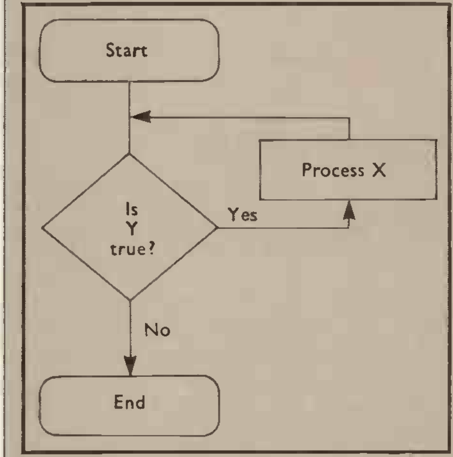


Figure 2

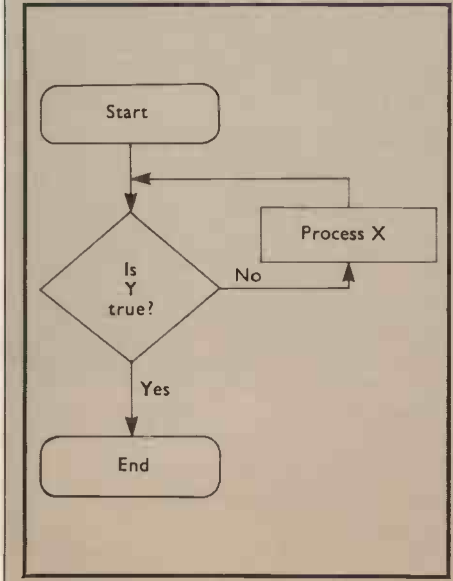


Figure 3

(continued from previous page)

should conform to one of those structures.

Each structure has a single entrance point and a single exit point.

The use of unconditional jumps or GOTO statements are not allowed.

The theory of structured programming states that the short program which we have called a structure must act like a single instruction. Thus, program control passes to our "instruction", an operation is performed and control passes to the next "instruction". Although all our sub-programs will be different, each one should conform to the structure of one of a small group of "structures", i.e., "instructions".

The set of different structures is, in fact, very small. It can be proved mathematically that all programs can be written using a set of just three structures. To those three we can add two more which are variations of two of the others. A description of the five "structures" follows:

Building blocks

Sequence. The simplest structure consisting of one operation following another, as in Fig. 1. They are executed by the computer in the order in which they are written.

DoWhile. A conditional loop structure; thus we do process X while condition Y is true, as in Fig. 2. As soon as Y is false, we exit from the structure.

DoUntil. A variation on the DoWhile structure where we now do process X Until Y is true; then we cease to do X and exit from the structure as in Fig 3.

IfThenElse. A conditional branch structure which does not involve a loop, as in Fig. 4. If a condition A is true we perform process X and if it is false, then we do process Y, or If A is true Then X Else Y.

Select. A variation on the IfThenElse structure, it is a multiple conditional branch shown in Fig 5. The first part of the structure compares an input value to a set of values each associated with one of the processes to be selected; if there is a match, then that process is done.

We have looked at the reasons why we need a methodology in programming and we have met the building blocks or structures to be used in writing a program; we must now consider how we use them.

If you look at the flow diagrams for the structures, you will notice that they are displayed in very general terms. Viewed in this way, structures can be employed at all levels of abstraction from the initial idea to the final program. This is because the processes labelled A, B, X, Y and the like are at all levels, except the lowest—that of the final program, themselves, one or more of the five structures.

This ability to use program structures

(continued on next page)

(continued from previous page)

from the inception of an idea is valuable in helping the programmer to write clear, logical programs. It is this clear, logical construction of a program which helps greatly reduce the time and effort in debugging it after it has been written.

Having had an idea for a program, there are several things which must be taken into consideration. The first—is your computer large enough? Does it have enough memory to run the proposed program?

These are very difficult questions since the answer lies with the type of machine, the language used and, to a great extent, experience. If your program is likely to be long, it is probably wise to consider splitting it into independent units and loading only one of the units at a time.

The next question is what kind of data storage is required, and how much. This also is a difficult question but we can make the answer easier to find by considering the database as a list of items where each item is a group of associated data.

If we want to find a particular set of data rapidly which could be in any position on our list and with no relationship to the previous set of data read, we require random access storage. In hardware terms this means a floppy or hard disc, if the database is large; or if it is small enough it could be stored in the main computer memory as an array.

If we want to look at the first item on our list, then the second, and so on, we require only a sequential file, which could be a floppy disc or a cassette tape store.

I shall not go any further into the problem of file structures and the associated hardware requirements, as this would necessitate a complete article. Having looked at the problems and decided you have the necessary hardware to run your proposed program, now is the stage to start planning and writing it.

Typical application

We have looked at the common structures and considered the principles of our programming methodology, but how do we convert, say, a business application idea into a program? Let us consider a typical application, a program to produce quotations for a company making specialist products from a range of standard components.

Instead of having to type a description of each component, its unit price, the quantity, and total price, we want to be able to type just a component number and quantity. The computer will then retreive from memory the relevant data from a list of components, labour charges and the like, perform the required calculations, and produce a neatly-printed quotation for the customer.

The printing of the quotation will be only one part of the program. We will

(continued on next page)

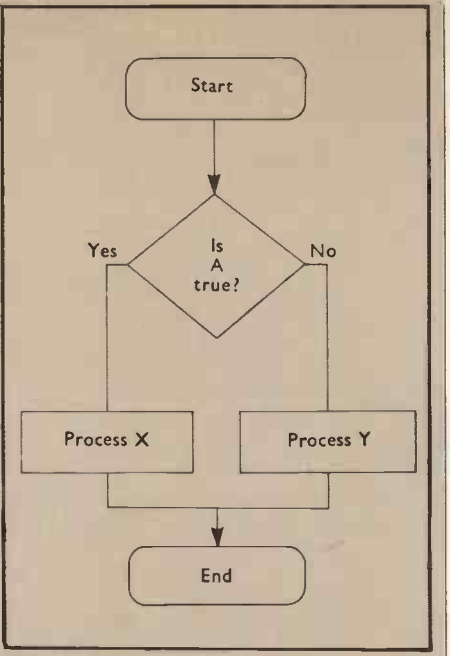


Figure 4

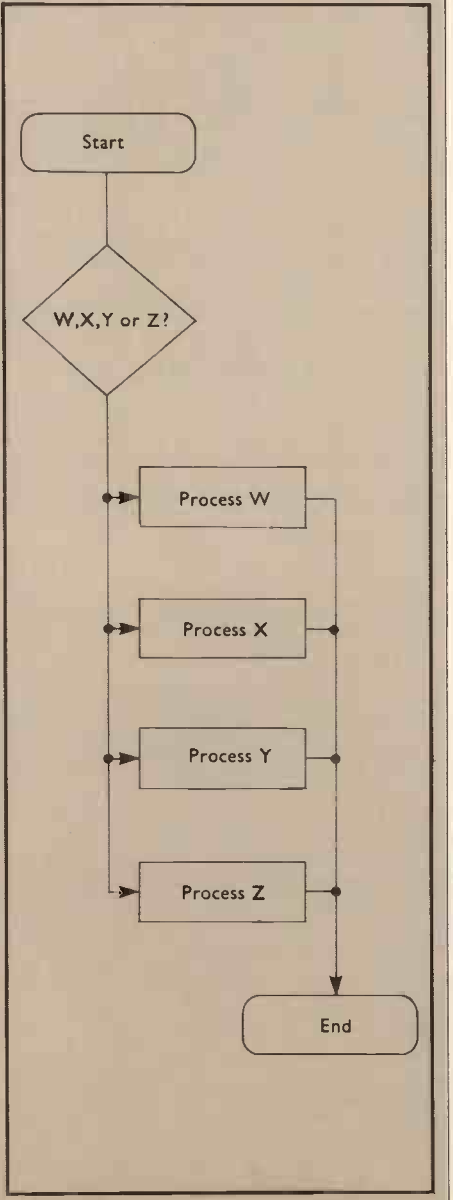
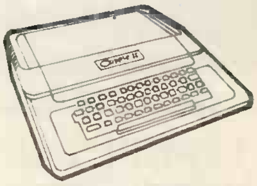


Figure 5



(continued from previous page)

want to input data about new components. We will also want to change data, such as the price of a component already in our database, and finally we will want to be able to print-out a complete list of all components, with component number, price and any other data which may be required.

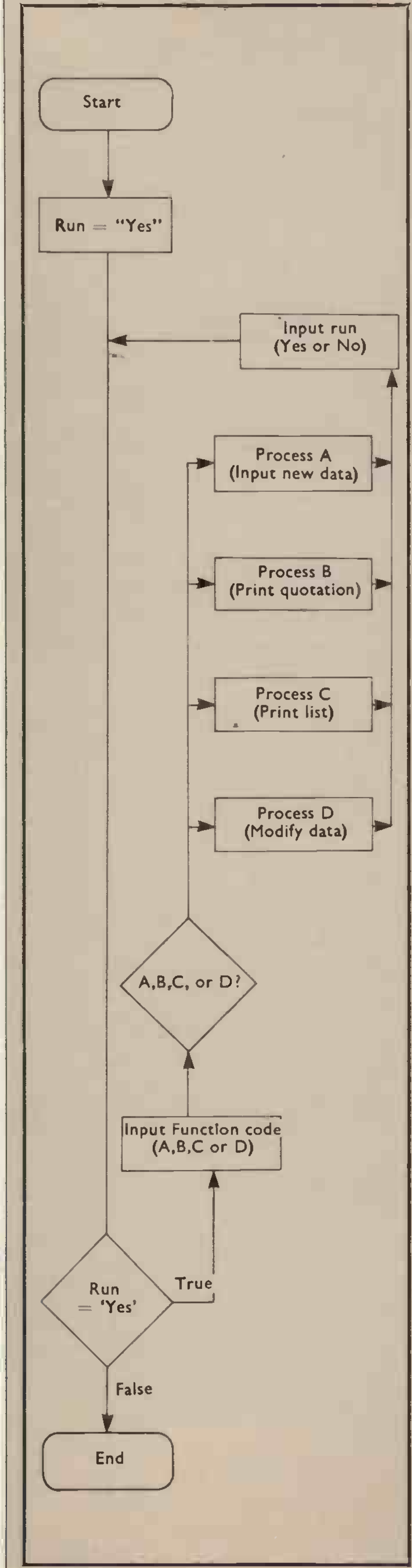


Figure 6

Each of those functions can be regarded as an independent program linked to the others only by a common entry and exit point and a common database.

When we run the quotation program we will want to select one of the four functions outlined—this, you will notice, corresponds to the Select structure.

Having performed one of the four functions, however, we want the facility to return to the beginning of the Select structure and select and run another function. We can do this by making our Select structure the process of a DoWhile loop.

Three further simple sequence structures are required to construct a viable program; thus we must input a code representing which function we wish to perform before we enter the select structure. Similarly, we must input a command after the select structure to be used by the conditional branch section of the DoWhile loop to determine if we exit or perform another function.

Shorthand method

Since the conditional branch section of the DoWhile structure requires a command immediately upon entry of that structure, we must insert a sequence structure before our DoWhile loop to set-up that command to cause it to loop.

We then have an outline for a program which will select one of four independent programs; the flow chart is shown in Fig. 6.

Obviously this is only a small part of the proposed quotation program but it does, I hope, show the way we can use structures to help turn an idea into a program.

Flow diagrams are, however, a rather tedious and space-consuming way of displaying the logical flow of a program. This is particularly so when a program contains many "nested" conditional or loop structures—structures which contain other similar structures. Such flow diagrams are difficult to draw and almost impossible to alter.

Fortunately, structured programming gives us a shorthand method of writing a program; Fig 7 is the quotation example written in such a manner. You will notice that the current structure is indented and preceded by a colon to separate it from the previous conditional or loop structure within which it is nested. We use the names of the structures at the beginning of the structures and END DO to mark the end of a loop and END IF to mark the completion of a conditional structure.

Every methodology has its diasadvantages, not least the one I have just outlined. A great disadvantage is that a structured program does not necessarily run as quickly as possible or use the least amount

(continued on next page)

(*continued from previous page*)

of memory space. It also introduces another step in the designing and writing of a program, thus adding to the work.

The major disadvantage though, lies not in the methodology but in the language in which we will finally write our program, and for most people that means Basic.

```
RUN = "YES"
DO WHILE RUN = "YES"
:    PRINT FUNCTIONS AND CODES
     "NEW—INPUT NEW COMPONENT DATA"
     "PRINT—INPUT CUSTOMERS NAME &
     ADDRESS, ITEM NUMBER & QUANTITY
     AND PRINT QUOTATION"
     "LIST—PRINT LIST OF DATA FOR ALL
     COMPONENTS"
     "EXAM—EXAMINE, AND MODIFY IF
     NECESSARY COMPONENTS DATA"
:    INPUT FUNCTION CODE
:    SELECT PROGRAM
:    :         IF 'NEW' THEN
:    :           : INPUT NEW COMPONENT DATA
:    :         IF 'PRINT' THEN
:    :           : PRINT QUOTATION FROM INPUT
                   DATA
:    :         IF 'LIST' THEN
:    :           : PRINT LIST OF DATA ON ALL
                   COMPONENTS
:    :         IF 'EXAM' THEN
:    :           : EXAMINE & MODIFY DATA
:    :         ELSE PRINT "ILLEGAL COMMAND"
:    END IF.
INPUT RUN (EITHER YES OR NO.)
END DO.
```

Figure 7

One of the great weaknesses of Basic is that it lacks structure, a factor attributable partly to Basic being a language of second-generation machines, and now more than 15 years old. It is an unfortunate fact that while most of us are using fourth-generation machines, we do not do the hardware justice by continuing to use ancient and inefficient software.

Increases confidence

This situation should be relieved fairly soon, since there are strong indications that Basic will be replaced by PASCAL, which is an up-to-date, structurally-orientated, high-level language.

A structured - programming - based methodology, if applied carefully, can be of great value to the programmer. It brings disciplines to programming which should result in the writing of programs with fewer errors. This, in turn, will increase confidence that you can write complex programs.

It is certainly not the only or probably the best aid to program writing. I hope this article has enabled some readers to write a better program and perhaps tackle programs they had previously considered themselves incapable of doing.

Library for CP/M users now available

COPIES of all 24 volumes of discs in the CP/M users' group library are now available in the U.K. from the Computer Centre, 20 Durnsford Avenue, Fleet, Hants (02514) 29607, price £10 per volume. This should prove a valuable source of low-cost software to users of systems capable of supporting CP/M. If you have an 8080 or Z-80 machine with an 8 in. drive. The Computer Centre can supply copies of CP/M from Digital Research.

Geoff Lynch, of Xitan Systems, 31 Elphinstone Road, Highcliffe, Dorset, (04252) 77126 has succeeded in getting CP/M to run both on 8 in. and 5 in. disc versions of the Cromemco Z-2; anyone interested should contact him.

Try all programs

If you are running a Cromemco Z-2 or System 3 under CDOS you should be able to run CP/M programs, provided they are written for a true version of CP/M. Recently I managed to get TDL Basic written for CP/M to run on a Cromemco System 3 but was unable to run either the Basic-E or Basic-C compilers. It may well be worthwhile for a Cromemco user with 8 in. discs to try all the programs in the CP/M users' group library.

When writing to people mentioned in this article, please don't forget to enclose a stamped self-addressed envelope.

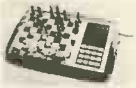
Display unit

THE SIRTON Computer Interactive Display unit which, when coupled to a TV set makes a self-contained VDU system with a 16-line, 64 character display, costs £97·50.

The ASCII keyboard which is ready for plugging into your computer system is priced at £38·50.

Sirton Products is at 13 Warwick Road, Coulsdon, Surrey. Tel. 01-660 5617.

NOTE: Certain programs reproduced in *Practical Computing* remain the copyright of the contributor. Readers wishing to use such programs should first obtain the permission of the copyright holder. Every attempt has been made to ensure accuracy of the programs and listings but *Practical Computing* cannot be held responsible for any errors or the effect of errors contained therein.



A PRACTICAL GLOSSARY

Continuing the terminological gamut from C to D

Core

There used to be plenty of it about, which is why old-style computer people often use core as a synonym for memory. Memory used to consist of little iron rings called ferrite cores—tiny but visible—which could be magnetised or not to record a 0 or a 1; and on some systems it still does. But core takes up plenty of room.

It does, however have the great virtue of not losing the information it held if you turned off the power—semi-conductor memory is volatile, which means its contents are lost when you switch off. There are ways, though, of overcoming that.

CPU (Central Processing Unit)

The central processing unit used to be easy to define, since it was the black box into which everything else on a computer system was hooked. It was big, and performed all the calculations and control functions in the system. A popular analogy was with the human brain, and that was about as useful as most analogies.

It is harder today to identify the CPU as a unit in its own right. It is still the part of the computer system which carries out the arithmetic and logical processes to which data is subjected, and it exercises final control over the physical components of the computer system.

CRT

A cathode ray tube, or CRT, is used to provide the visual display in most types of visual display unit. It is the same kind of cathode ray tube used as a television screen which is why a TV set can be utilised as the display monitor of a terminal. The term is sometimes used sloppily as a synonym for VDU.

Cursor

A cry of rage of finding the gerbils have made a nest out of Star Trek source listings. Also, a means of indicating that position on a VDU screen where the next character will appear, either by typing in the character in or the computer sending it to the screen. The censor might be a square of light, a blinking underline, or an illuminated hollow square.

Cycle Time

The time taken to access and read from a location in memory—not to be confused with access time. Cycle time is a good measure of the performance of a mini or micro system; there tend to be too many other considerations for it to be as useful a measure on a mainframe.

Cycle time is a function of memory, the processor and their connections, rather than of the processor alone.

Data

This is what it's all about; everyone knows data is what the computer 'does' or 'uses' or 'has'. Data is just information which is processed, stored or produced by a computer. There are two wrinkles, though. In fact 'data' is plural noun—one piece of data is a 'datum'; but you don't find many people following the strict grammatical rule. The language moves on and so data ·s generally singular. In any case, it is more helpful to think of data as a composite rather than a collection of individual items. The other interesting point is the distinction between data and *information*. Data is the abstract and uncoloured subject on which the blind, rule-following computer performs its programmed operations; information is data which has been given some meaning, the kind of data humans can use.

Database

Sometimes two words, but not in our style. It is a large file of data organised so that all users draw upon a common pool of consistent, up-to-date information. A database usually needs large backing memories and a filing system developed with all potential users and applications in mind. The emphasis is on the shared pool. In a non-database environment which is the norm, of course, each user generally has to set up separate files for each application. The database approach means that if the programmer knows the information is already there, it is no trouble at all to get at it. Micro systems typically lack the storage capacity to make a database worthwhile.

Database management system

A DBMS is a software system for designing, setting-up, and subsequently managing a database. In practice, they smooth the interface between a programmer and the data, providing the kind of facilities to allow the user to specify which data a program requires. Database managers are complicated and are usually expensive pieces of software; a couple have been developed for micros, even so.

Data capture

Getting the information at the instant a transaction occurs—the recording till information as customers check out in a supermarket point-of-sale system.

Data dictionary

Or data directory. A complex but usually useful software system which describes the forms and characteristics of data within the computer and also defines the relationship between data. A data dictionary usually applies to big systems, especially those using databases, where the exact nature of data items can be hard to find. The programmer has to know the agreed name for a field in a particular record, its size and the type of data it can contain, and with which other data and programs it can interrelate. The data dictionary guarantees uniformity. It's very wonderful.

Data entry

Getting data into the system; validation often takes place at this stage.

Data manipulation language

Part of a DBMS.

Data preparation

Pre-processing of 'raw' data before a batch data entry input. Typically, it means card punching or keying on to mag tape or disc. Subsequent data entry from input devices like a card reader or disc drive is much faster than any o the direct data entry methods, like input from 'a VDU. Those by-pass the need for data prep, though, and if you have to key-in some information once, it might as well be at a VDU rather than a card punch keyboard.

Data transmission/communications

Sending and receiving of data via a telecommunications network.

Data vetting

Procedure for checking automatically that new data corresponds to defined criteria.

Debug

Remove errors from system. A canny debugger is a good programmer. You can have plenty of jokes with this word.

Decay time

Time during which an electronic impulse fades. It usually applies to characters on a CRT screen, which have to be 'refreshed' before the decay becomes noticeable.

Decimal notation

System of writing numbers where successive digit positions are presented by successive powers of radix 10. It's what we use in day-to-day living.

Deck

Frequently-used alternative to *drive* (qv) when referring to mag tape. Alternatively a pack of puched cards.

Degradation

The computer business is full of scope for anyone lusting for *double entendres*. This one relates to a lower level of service in the event of a computer break-down. Graceful degradation is the ability to ensure that failure of certain parts of equipment does not result in complete breakdown but allows limited operation.

Diagnostic

Routine or program designed to detect faults. Minis often have them built-in on ROM chips. For micro systems, diagnostics are generally little software programs which have to be loaded and run.

Digital

As opposed to analog, a system or device using discrete signals to represent data numerically. Most digital representation in computing is based on the binary system (qv).

Digitising pad or tablet

Neat input device which allows information in the form of hand-drawn lines to be entered into the computer.

Direct access

Ability to extract immediately required data from memory, regardless of its location. As opposed to sequential or serial access, so you can't have direct-access mag tape systems.

Direct address

Or absolute address, machine address, specific address; a standard, permanent identification of location in main memory.

Diskette

Or floppy disc, or flexible disc both of which are very explicit descriptions. A standard mini-floppy gives direct-access (qv) storage for about 71K bytes; at the other end of things, a double-sided, double-density 7¼ in. diskette stores one megabyte.

Down-time

Period a machine or system is out of action due to malfunction or maintenance.

INTERESTED IN HOME COMPUTING?

Nascom Microcomputers

BLANK C12 Racal Quality CASSETTES £4.90 for 10

Start now and don't get left behind

THE NASCOM 1 is here

Ex-stock with full technical services

Plus the opportunity to join the fastest moving club of personal computer users enabling you to get the most our of your computer. You can OBTAIN and EXCHANGE programs and other software – many now available.

The Powerful Z80 Microprocessor
Professional Keyboard
1 Kbyte Monitor in EPROM
2 Kbyte RAM (expandable)
Audio Cassette interface
Plugs into your domestic TV

Easy construction from straightforward instructions
– no drilling or special tools
– Just neat soldering required.

Only **£197.50** + 8% VAT (includes p & p + insurance)

Manuals seperately	3.50	Monitor quality improved TV Modulator	2.50
Z80 programming Manual	6.90	Super Modulator (10MHz)	5.50
Z80 Technical Manual	3.40	Power supply suitable for NASCOM	19.90
PIO Technical Manual	3.40		
(All prices add 8% VAT)			

Available from COMP now. *4K and 16K RAM expansion boards for NASCOM 1 with sockets on board for MINIBASIC EPROMS. Trade, government and educational enquiries invited. Can be supplied ready built tested and guaranteed.

KEY BOARD

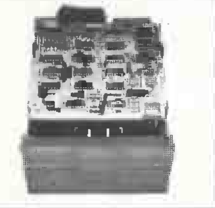
Brand new professional ASCII keyboards (USA)

Full technical details included.

Only **£49.90**
+ 8% VAT.

Ready built, tested and guaranteed.

NEW– AVAILABLE NOW



Shugart floppy drives	**£290** each
Floppy drive controller direct from USA	**£190** each

ETI 8080 ONE BOARD COMPUTER

ET1 8080 COMPONENTS IN STOCK

SEND SAE FOR PRICE LIST OF PARTS AND KIT.

Introducing the personal computer you've waited for. THE EXIDY SORCERER.

SORCERER COMPUTER SYSTEM

The Sorcerer Computer is a completely assembled and tested computer system. Standard configuration includes 63-key typewriter-style keyboard and 16-key numeric pad, Z80 processor, dual cassette I/O with remote computer control at 300 and 1200 baud data rates, RS232 serial I/O for communications, parallel port for direct Centronics printer attachment. 4K ROM operating system, 8K ROM Microsoft BASIC in Rom Pac^{TM}, cartridge, composite video of 64 char/line 30 line/screen, 128 upper/lower case ASCII set and 128 user-defined graphic symbols, operation manual, BASIC programming manual and cassette/video cables, connection for S-100 bus expansion.

only £950 Credit facilities available.
+ 8% VAT

AS SEEN ON OUR STAND AT PCW SHOW

LOOK!

*32k RAM (16K available)
*RS232 interface * 8K BASIC ROM
*CUTS interface * 4K MONITOR
*KANSAS CITY interface * S100 BUS
*User defined graphic symbols. * Z80 cpu

VIDEO GAMES & Components

UHF Vision modulator Standard channel 36	2.90
UHF Vision modulator 10 MHZ bandwidth high quality	5.90
VHF Vision modulator channel 3	3.10
Sound modulator compatible with above	2.90

AY-3-8500 6 Games	4.90	PCB	1.90
AY-3-8550 6 Games	3.90	PCB	1.90
AY-3-8610 10 Games	5.90	PCB	1.90
AY-3-8760 Motor-bike	6.90	PCB	1.90
AY-3-8710 Tank-battle	6.90	PCB	1.90
AY-3-8603 Road Race	5.90	PCB	1.90

*** ETI Tank Battle kit** – just add controls – **£14.90** reduced from **£19.90.**

JOYSTICKS – **£1.90 each** Dual 200K lin

COMPUTER COMPONENTS

A selection of our Computer components

CPU AND SUPPORT DEVICES

Z80	cpu	14.90
Z80A	cpu	20.50
Z80	PIO	9.90
Z80	CTC	9.90
8080A	cpu	7.95
8212	8 Bit I/O Port	3.50
8214	Priority Interrupt Control	9.00
8216	Bi-Directional Bus Driver	3.75
8224	Clock Generator & Driver (2MH$_z$)	3.50
8224-4	Clock Generator & Driver (4MH$_z$)	9.95
8226	Inverting Bi-Directional Bus Driver	3.95
8228	System Controller & Bus Driver	7.95
8238	System Controller & Bus Driver	7.50
8251	Programmable Communication Int.	9.95
8253	Programmable Interval Timer	18.95
8255	Programmable Peripheral Int.	12.00
8257	Prog. Direct Mem. Access Cont.	17.95
8259	Programmable Interrupt Cont.	17.95
6800	cpu	11.90
6810P	128 x 8 Static Ram (450ns)	4.95
68B10P	128 x 8 Static Ram (250ns)	6.00
6820P	Peripheral Interface Adaptor	7.50
6821P	Peripheral Interface Adaptor	7.50
6828P	Priority Interrupt Controller	11.25
6834P	512 x 8 Bit Erasable Prom. 500	16.95
6850P	Asynchronous Comm. Adaptor	9.75
6825P	Synchronous Serial Data Adaptor	11.75
6860P	0-600 RPS Modem	10.00
6862P	2400 RPS Modulator	14.50
6871P	Clock	28.00
6875P	Clock	8.75
6880P	MPU Bidir. Bus Ext.	2.50

MEMORIES

2102	1K	1.20
21L02	1K	1.40
2101	1K	2.95
2111	1K	3.95
2112	1K	2.95
2114	4K	9.95
4116	16K	19.90
1702A	256B	5.00
2708	1K	10.90
2708	(NASBUG)	13.90
2716	(5V, INTEL)	29.00

OTHER USEFUL DEVICES

6576	char gen	10.90
6402	UART	9.90
1771	Floppy disk controller	49.00

Full selection of US and British Magazines and books for the Computer Hobbyist.

Prices always changing (usually downwards) Phone with your requirements

All prices include VAT except where shown. Orders over £5 post and packing free otherwise add 20p. Please make cheques and postal orders payable to COMP, or phone your order quoting BARCLAYCARD or ACCESS number. For technical information and advice ring JIM WOOD, BSc(Eng), ACGI, MIEEE – Consultant to COMP.

BARCLAYCARD VISA

We welcome Access

COMP COMPUTER COMPONENTS

14 STATION ROAD · NEW BARNET · HERTFORDSHIRE · TEL: 01-441 2922
01-449 6596
CLOSE TO NEW BARNET BR STATION — MOORGATE LINE
OPEN–10am to 7pm–Monday to Saturday

● **Circle No. 309**

We're getting bigger to give you a better service

We've opened a Sales and Customer Service Centre in London. And our new factory is fully operational. Southwest Technical Products provide a range of superb computer systems with technical backing second to none.

Systems
To suit all types of user – OEM, process control, data handling, small business systems, and all accounting functions.

Software
Low cost packages for word processing, selective mailing, progress control and invoicing. Our Software Development Unit available to prepare programmes to customer specification.

Training
Inexpensive courses (at Dover Street): BASIC – programming for the businessman; microcomputers in EDUCATION; WORD PROCESSING made easy; SOFTWARE DEVELOPMENT – make your microprocessor work.

Maintenance
Comprehensive national service by Computer Field Maintenance Ltd.

Sales Office: 38 Dover Street, London W1.
Tel: 01-491 7507 Telex: 268913.
Factory: 12 Tresham Road, Orton Southgate, Peterborough.
Tel: 0733-234433 Telex: 32600.

SWTPc

the symbol of reliability

72

● Circle No. 310