commodore

COMPUTING

April/May 1982 £1.25

the vic centre

Big savings for starters

If you're looking for a home computer, you'll already know that the VIC does it all. To help you get started, we have put together this attractive cost-saving package.

VIC-20 Colour computer

+ Cassette Deck
+ 10 Blank Cassettes
+ Introduction to Basic Part 1
+ User Manual
+ 12 months warranty
+ Fitted 13 amp plug

for only **£212**

(price plus VAT £243.80)

NEW

Convert your VIC to a 40 Column 32k machine with this exciting new add-on

32k RAM Expansion + 40 column screen format **£199.95** plus VAT

40 column screen format only **£99.95** plus VAT

Dot Matrix Printer

Tractor feed, 80 characters per line at 30 characters/seconds. **£199.96** plus VAT

Memory Expansion Board

Multi-slot unit used to accept memory and/or game and program cartridges **£85.00** plus VAT

Plug-in Memory Expansion

3k – **£26.04,** 8k – **£39.09,** 16k – **£65.17** plus VAT

Single Drive Floppy Disk Unit

£395.00 plus VAT

The standard features of the Vic are: ★ 5K RAM expandable to 32K ★ 16 screen colours ★ 8 character colours ★ 3 tone generators, each of 3 octaves – plus white noise generator ★ Screen display 22 characters × 23 lines deep ★ Full PET type graphics ★ High resolution graphics capability ★ 8 programmable special functions.

Goods required — Price

Add £4.00 post, packing and insurance for special deal and £1.00 post, packing and insurance for all other items. Add 15% to all prices for VAT

Total £

Name:

Address:

MAIL ORDER to: Adda Home Computers Ltd. FREEPOST, London W3 6BR or telephone your order (24 hours a day) to 01-992 9904 quoting your Visa, Access or American Express number.

*I enclose a cheque, made payable to Adda Home Computers Limited for £.....

*Please charge my Visa/Access/American Express account. My acount number is

*Please add my name to your mailing list

*Delete as applicable

Date .....

SHOP ADDRESS: Adda Home Computers Ltd. 154 Victoria Road, Acton, London, W3. (near North Acton tube station) Tel 01–992 9904 OPEN: 10am–6pm (Tuesday-Friday), 10am–5pm (Saturday).

V

adda

TERMS AND CONDITIONS: All goods sold subject to Adda terms and conditions of sale. Full details available on request, but include: 7 day money back guarantee, Adda 12-month hardware warranty. Please allow 21 days for delivery. Allow 7 days for personal cheques to be cleared. Quoted prices are exclusive of VAT.

Contents

Editorial

Editor
Pete Gerrard

Advertsing Manager
David Lake: tel 01 839 1881

Editorial Assistant
Fiona McCormick

Production
Sue Brown

Contributing Editors
Mark Clarke, Dave Middleton

Managing Editor
Nick Hampshire

Commodore Computing is published 10 times per year by Nick Hampshire Publications. It is not in any way connected with Commodore Business Machines U.K. Ltd.

Printed by Edwin Snell printers, Yeovil, England.

If you would like to contribute to Commodre Computing, please send articles or programs to :-

Commodore Computing
Hobhouse Court
19 Whitcomb Street
London WC2

We will pay 10 pounds for each program printed, and 20 pounds for each article published, which should be approximately 1,000 words long.

There is certainly a market for a professionally produced and presented magazine based on one particular micro. With all due respect to User Clubs and groups, they have neither the funds nor, more usually, the time to devote to producing such a magazine. Consequently owners of just one breed of computer have to wade through a lot of material in any magazine before finding something that is of interest to them.

Perhaps that's how it should be: you may find another machine that is significantly better than the one you've already got. However, people are not going to spend (in some cases) a considerable amount of money, only to be persuaded a short time later that they want a machine that does XYZ rather than ABC. In almost every case a lot of time will have been spent on choosing your computer: you want to find out what is the best for your machine, rather than reading about different ones.

Which brings us back to the Commodore family of machines. It is extremely difficult to pin down a precise number of users, but a reasonable estimate of the number of people who own, use, or have access to a Commodore computer must border very close to a quarter of a million, if not many more. Just consider the number of businesses, schools and colleges who have PET's and/or VICs.

It is because of that 'dedicated' audience that this magazine exists. There is no other like it on the market at the moment, nor is there ever likely to be. Everything in this magazine is aimed at the dedicated user, and we attempt, as far as is possible, to have articles that are aimed 'across the board', covering the entire specturm of Commodore equipment. Obviously this will not be possible in all instances, but we will try!

Letters

Dear Sir,

Commodore Club News - January 1982

I tried some of the programs that were listed (in Club News), only to become disappointed when they didn't work. This is particularly so with machine code programs, which are really boring to enter, and usually result in nothing happening at all, except for the PET locking up. I have learnt the hard way that codes that use the second cassette buffer (decimal 826 onwards) should not be tried on our PET 8032.

It has been suggested before, but couldn't a MACHINES code be used to signify which machines programs will work on?

M Code A - PET 8032/8050
M Code B - PET 2001, etc.

I thank you in anticipation.

Allan Thompson
Grimsby

Dear Allan,

Thank you for your letter. Please accept our apologies for some of the programs in the January issue of Commodore Club News not working. I suspect the major part of the problem was that Club News was almost a one man show; time precluded checking everything! Rest assured that all programs in Commodore Computing are checked prior to publication.

Your suggestion for heading each listing with a code signifying which particular machine it is intended for we will certainly use in some way.

Dear Sirs,

What I am writing to you for is some more information on Machine Code Programs i.e. examples of machine code programs, where I can obtain these, and if you publish any books that cover the area from BASIC to machine code programming.

I have tried in vain to obtain the VIC Programmers Reference Guide, hoping this could help me in this area, but I think that like most of Commodore software this is still in its development stage.

Any information you can give me would be appreciated.

Yours faithfully
J. Stanton
Birmingham

Dear Mr. Stanton,

Thank you for your letter. We would be the first to agree that machine code programming is a much neglected area, certainly as far as the beginner is concerned. We shall be endeavouring to correct that in later issues. We've seen the VIC Programmers Reference Guide, and think that you'd be far better off sticking to the VIC Revealed!

Two books that you would certainly find useful are "Programming the 6502", by Rodney Zaks; an excellent guide for the newcomer to machine code programming. Secondly, a superb reference book is "Programming the PET/CBM", by Raeto West. Both of these you should be able to get from your local Commodore dealer.

Dear Sir,

I have a few questions about BASIC 4.0 and the disk drive (4040).

Why is it necessary to check for less than 768 bytes free, asking :-

IF FRE (0) < 768 THEN PRINT "OUT OF MEMORY" : STOP

Isn't it possible to use a smaller value, as 768 bytes is a lot of memory to lose?

About the disk. Does it matter which secondary address you use for a disk file (NON-SEQ)? I read that 0 and 1 are for LOAD/SAVE and 15 is the command channel. So is it therefore possible to use any other (2-14)?

Thank you for your co-operation.

Edmund Doyle
Bedford

Dear Edmund,

Thanks for your letter. The reason for checking for 768 bytes free is to prevent the occurence of the dreaded 'garbage collection'. Despite everything, this still happens in BASIC 4.0 machines, and the only way around it, other than sitting and waiting for hours, is to stop it happening yourself. When you reach the stage where you have less than 768 bytes left, strange things can happen if you start playing about with strings. So, you check for when you get to that stage, and then bomb out!

On your second point, you are quite correct. You can use any number between 2 and 14 as a secondary address.

Dear Sirs,

I wondered if you could give the answers to some problems.

1) Where can I get the VIC assembler/disassembler cartridge ? Does the assembler allow the use of labels, makro definitions, .word, .byte and other pseudo operators?

2) I have read that 1024 X 1024 resolution can be obtained. Is this true? It is possible to have more than two colours in the same 'square' on the screen in hi-res mode?

3) Is there going to be an IVUG for VIC as opposed to IPUG?

4) Any news on a 40 column convertor for the VIC 20?

Thank you.
Yours faithfully

J. Bell
Fareham

Dear Mr. Bell,

To answer your queries in order, the VIC assembler/disassembler cartridge is available from any Commodore VIC dealer, and costs 34.95 pounds. Ask for the machine code monitor cartridge. However, it does not meet your latter requirements as it is only a mini-assembler, and therefore everything is done directly.

The question of a resolution of 1024 by 1024 has risen its head a number of times. Perhaps this is the time to quash that one on the head. It is true that using the Super Expander cartridge in high resolution mode allows you to plot points to that scale, but when they are actually displaced on the screen they are reduced down to a resolution of 160 by 160. As to colours, in hi-res mode you can only have one character colour per 8 by 8 dot area, i.e. a specific space on the screen. Of course, you can still specify foreground and border colour.

IPUG have recently changed their name to ICPUG (Independent Commodore Products Users Group), to reflect their broadening of horizon onto the VIC as well as the PET. Details of specific regional groups will be appearing in this magazine.

There isn't as yet a 'true' 40 column convertor for the VIC 20, although Beelines in Bolton (0204 382741) have produced an expansion unit which displays the VICs screen to Viewdata format (40 columns by 24 rows). Incidentally, their unit also gives you an additional 32K of RAM! However, you can not program it in 40 columns yourself; it is only a display area. 'Rumours from Commodore' department: there are speculations about a certain person within that company writing a 40 column editor for the VIC, with which you WILL be able to write true 40 column programs. Watch this space!

KEEP IN TOUCH

If you've any point of view that you'd like to air, or any question that you'd like an answer to, drop us a line. It's your chance to keep in touch, both with us, and with other users.

The Editor reserves the right, prior to publication to amend/alter any letter as he sees fit.

New Product News

Basic Aid

What is a BASIC AID? It is a programmer's delightful utility on disk for 16k or 32k PETS, BASIC #2.0 or #4.0, similar to the well-known TOOLKIT. It's in machine code and therefore fast, and fully relocatable: for beginners this means YOU can decide where you want the code to be stored in memory - usually top of memory is best, and an example is provided to enable you to do so. Whilst it may be similar to the TOOLKIT; it offers many more facilities (see later), and requires 3k or RAM.

On my disk there are two programs; (1) the machine code routines suitable for relocation as required; (2) a set of commands with instructions on how to relocate the machine code in (1). My early version did not have a manual; so later versions may well replace this second program with a manual (?). In the event that a manual is not supplied; I suggest that users first LOAD & RUN the second program: which then usefully asks whether screen or printed output. If you choose printer output, and this is especially recommended for later reference, note that the printer is assumed to be device # 4.0 - for most persons this will be acceptable: if not LIST line # 170; this calls the printer and should be changed to accommodate a different printer device number (say, OPEN1,5).

When located and ready the screen displays the facilities available:

COMPATABILITY: Appears to be quite compatable with Supersoft's SUPERCHIP - turning off SUPERCHIP also disables BASIC AID.

COMMENTS: Having used Toolkit before upgrading to BASIC 4.0; I can certainly recommend BASIC AID. Not only is it about 1/3 the price of Toolkit but there is the improved and / or extra functions. If located at top-of-memory BASIC AID can be LOADed and RUN without disturbing the current program - thus no real disadvantage between software and firmware. So now that you have decided that you really need BASIC AID (which of course you have); once you have purchased it you can buy the PLUSDOS chip which goes on from where BASIC AID left off (and that's quite a long way). However NOTE; for copyright reasons PLUSDOS is only available to those who can produce documented proof of purchase of BASIC AID. See subsequent review of PLUSDOS and BASMON chips in next months issue.

COMAL 80 Board

COMAL (a language somewhere between BASIC and PASCAL) has for some while been available for certain of the Commodore range of micros, but even on those you were very limited in the amount of work space available. With the introduction of this 64K PROM board from Instrutek, you now have access to COMAL on any BASIC 2.0 or 4.0 PET, and up to 30K of work space on a 32K PET, 14K on a 16K PET. The board fits easily into the PET, and can be installed by even the most novice of engineers. Power on of the PET now puts you into COMAL mode, but just by typing in the command BASIC, you're back into BASIC again, so all those packages you've already got will still quite happily work. There's also an interesting new keyword added to the vocabulary, SAFE. This protects any program from unauthorized listing. Two more options on this useful board include 16K of EPROM storage area, and a high resolution extension board with a display of 512 by 256 pixels. Anticipated cost is somewhere between 200 and 300 pounds.

VIC 20 Expansion System

This product, from Afron Microelectronics Ltd. has recently been given the Commodore seal of approval as one of the first arrivals in the VIC Approved Product scheme. Competing directly with Commodore's own motherboard, the unit is priced at 85 pounds (plus VAT), and fits nicely in with your VIC. All is enclosed in a finished metal case, which is approximately the same colour as the VIC. It holds up to seven cartridges, thus giving you the facility to have up to 30K of RAM, various ROM cartridges, printer software, disc software, RS232 and IEEE interface, and so on, all at the same time. It comes complete with its own power supply, and hence you have enough power for any expansion work you care to undertake. The VIC and its screen modulator all fit inside, and there is an optional 'lid' to the unit to allow your T.V. or monitor to sit on top, giving a totally intergrated system. All the existing VIC ports are still accessible: all told, a most useful system.

VIC Light Pen

The rapidly growing number of available VIC peripherals is particularly in evidence from a company called Stack Computer Services in Liverpool: here we take a brief look at their VIC Light Pen.

Economically priced at 28 pounds 75 pence, including VAT, the pen connects to the control port on the right hand side of the VIC. The flexible wire from the port to the pen itself is sufficiently long enough for the T.V. or monitor to be placed a number of feet away from the VIC, if you're one of those people who do not like the visual display being too close.

Perhaps one of the reasons for the economy of

price is the sparcity of accompanying documentation. It must be stressed that it is not only Stack who are guilty in this area: many, many microcomputer companies seem incapable of providing clear and concise written material to go with what, in the majority of cases, is a good product. Essentially, you have to learn about the product yourself, rather than being able to learn from the manufacturer's instructions.

Badly presented documentation can often be compensated for by being accurate; thankfully, Stack managed that! Still, one's final opinion should depend on the product itself, rather than anything else.

The position of the light pen is detected by looking at registers 36860 (X co-ordinate) and 36861 (Y co-ordinate); this gives you an effective resolution on the screen of approaching 90 by 90. For more efficient use of the pen this needs to be reduced to the VICs own resolution of 22 by 23, which can easily be accomplished by a line or two of BASIC. Thus, from software, you can detect the position of the pen on the screen and scale your response accordingly.

The light pen would probably be of most use in a 'question and answer' scenario, where the user is required to input an answer related to a question on the screen. Many users will not have encountered a microcomputer before, and will perhaps be reluctant to actually 'touch' the keyboard. This light pen gives the opportunity for getting around that problem.

Merely by touching the pen onto an area of the screen produces a change in the two memory locations mentioned earlier; assuming that the software is correct, the programmer detects where the pen is, takes the answer, and moves on. Simple and easy to use, one cannot really say more about the pen other than, that it works, and performs the job it was designed for. I'd have liked to have seen better documentation, but these are (relatively) early days for the VIC: perhaps this situation will improve with time.

Ultimax

Rumours have been filtering through the press lately with regard to new machines coming out of the Commodore factories. From what we've seen, 5 and 7 megabyte disks are on their way (Commodre's own this time!), along with a variety of PETs. Somewhat suprisingly though, the rumours that have attracted the most attention have been those concerning variations on the VIC theme, in both directions.

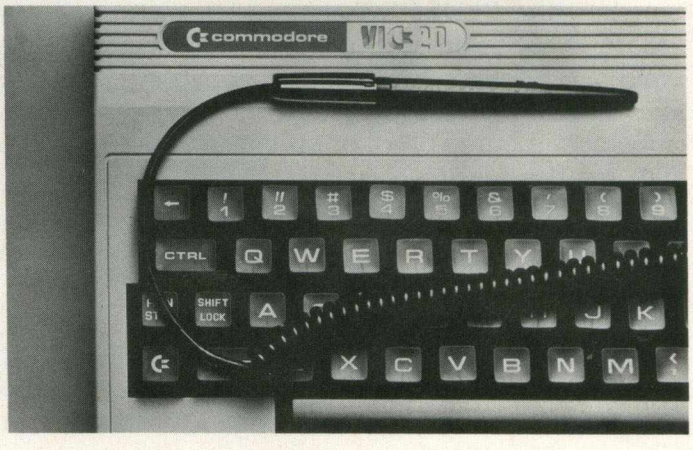
I say suprisingly because I would have thought that bigger and better business machines would have been of the most interest to the average computer user. However, it's the VIC brothers and sisters that seem to be hitting the presses the most, so lets take a look at the newest, and smallest, of these the Ultimax.

It's an unusual departure for Commodore to give a brand new machine a name rather than a number. However, this particular machine is breaking totally new ground for them, so perhaps that's the answer. The VIC came out as a confusing machine; what audience was it aimed at? At 200 pounds it cannot really be classifed as a true home computer; on the other hand the cost of taking it up to a complete business system (albeit with 22 columns across the screen) is rather prohibitive. An excellent machine, but it fell into a limbo between the PETs and something like the Sinclair ZX81. The Ultimax is aimed quite deliberately at that latter market: there is no confusion over this one!

The raw facts first; it has a U.S. selling price of $149.95, so we can expect a U.K. price of something like 99 pounds. Delivery dates are not quite so clear; people are quoting midsummer, but from past experience I don't expect we'll see any great quantities until somewhere nearer Christmas. I hope I'm proven wrong! On board memory stands at 2 or 2.5K (no-one seems quite sure at the moment), with an additional 8K expansion cartridge. However, as only one cartridge can be plugged in at a time, and BASIC comes on, you guessed it, a plug in cartridge, memory expansion seems like a lost cause. All this is still subject to change, so things may be different when it appears 'en masse'.

It has full colour capabilities, and like the VIC connects into your own T.V. Also on board is something called S.I.D. (Sound Interface Device?) which I recall Jack Tramiel mentioning at the PET Show last year. Basically it turns the machine into a synthesiser, and from what I heard it sounds very impressive.

Screen layout is 40 by 25, and perhaps because of



that there is not much compatibility with VIC cartridges and periperals. On the other hand Commodore have signed an agreement with Bally to transfer some of their arcade games over to Ultimax, so who knows what we might see ?

One major reservation is that there are no plans for any printers, disk drives (or tape decks, according to one source)to be even connectable to the Ultimax: I believe that therein lies a mistake. Still, Commodore are obviously going all out for the games market, and probably believe that there will be no demand for such peripherals.

We'll be bringing you more details of this in a future issue, and indeed more details of:-

2016 & 2064

These are two new VICs, although somewhat confusingly Commodore doesn't refer to them both as such. The first one is quite simply a 16K VIC, and still with a 22 column screen. The second one is being referred to as The Commodore 64, but looks remarkably like a VIC. It has 64K as standard, and this time we have a 40 column screen. Full colour and sound, like the VIC, and it takes all VIC peripherals (and indeed Ultimax cartridges). The U.S. price is $599, so you can expect a U.K. price of around 399 pounds when it appears. When ....

All three of the above machines should be on view at the PET Show, so ....

The PET Show

This year's bash has been re-named The Third International Commodore Computer Show, and takes place on the 3 - 5 of June this year, at the Cunard Hotel in Hammersmith, London. Interesting enough, this year sees the first ever Apple Show, (on the 4 - 6 of June!), and they've chosen to have their exhibition at the Fulcrum, in Slough, and Slough is as you know, where Commodore hang out. Taking coals to Newcastle ...?

18,000 square feet of exhibition space this time, and an estimated 120 exhibitors. There'll be the usual mixture of seminars, and the legendary Jim Butterfield will be here. It'll be good to see him back again. Exhibition times are 12 till 6 on the Thursday, 10 till 6 on the Friday, and 10 till 5 on the last day, Saturday.

More on the PET (sorry, Commodore Computer) Show next month.

Club News

Last issue we published a list of local independant user groups around the country, but due to lack of space we couldn't fit in all the details of where and when they meet. To make up for that here we present the full list.

In addition to these, there are a number of other user groups world wide. The principal ones, i.e. the most active ones, are listed below :-

CBM in Canada, who publish the magazine Transactor, and the man to contact is Karl Hildon.

3370 Pharmacy Avenue
Agincourt
Ontario M1W 2K4
Canada

CBM in America, who publish the Commodore Magazine.

681 Moore Road
King of Prussia
Pa. 19406
U.S.A.

CBM in Australia, who again publish a monthly magazine, under the editorship of Stephen Lee.

P.O. Box 336
Artarmon
New South Wales
Australia

Region B — Glos./Bristol Area: Mrs. Janet Rich, 23, Sheppard Leaze, Wotton-under-Edge, Glos. Informal meeting last Friday of every month, except December, at above address.
Region C — Hampshire Area: Ron Geere, 109, York Rd, Farnborough, Hants. Informal meetings 3rd Wednesday of the month at the Civic Hall, Fleet.
Region D — South East Region: Wing Cdr. Mike Ryan, 164 Chesterfield Drive, Riverhead, Sevenoaks, Kent. Meetings every 3rd and 4th Thursday of every month at Charles Darwin School, Jail Lane, Biggin Hill, Kent at 7.30p.m.
Region D — Canterbury Group -For details please contact John Bickerstaff, 48 Martin Down Road, Whitstable, Kent, Home tel. 0227 272702 Office tel. No. 01-499-9102.
Region F — Berkshire Group - Brian Jones, Dept. of Maths and Computing, Slough College of High Education, Wellington St., Slough. Tel Slough 34585 ext. 81. Their first meeting was held on 29th March, please contact Brian for future dates.
Region G — Essex Area - Please contact Walter Green, 151, The Hatherley, Basildon, Essex for further details.
Region H — Norfolk/suffolk Area - We need a volunteer in this area.
Region I — Herts/Watford Group -Stephen Rebagliati, c/o Institute of Grocery Distribution, Grange Lane, Letchmore Heath, Watford. This group meets on the second Monday of every month.
Region I — Herts./Stevenage Group: Brian Grainger, Tel: Stevenage 27925. Informal meetings on the last Wednesday of every month at Provident Mutual Life Assurance, Purwell Lane, Hitchin Herts.
Region J — Wales - Contact - F.J. Townsend, The Mill, Rhydowen, Llandyssul, Dyfed, SA44 4QD. Tel: 05455 5291.
Region K — Birmingham Area: J.A. McKain P.P.I Ltd., 177 Lozells Rd, Birmingham. Tel: 021-554 0202. We have the venue, but some need some help, any offers?
Region K — South Midlands: M.J. Merriman, 12 York St., Stourport-on-Severn. Meeting on the last Thursday of every month at the above address, where one can receive help with business programming problems.
Region L — Derby Area: R. Davies, 105, Normanton Rd, Derby, Tel: 0332 514016
Region N — Manchester Area - We have the venue, but need someone willing to organize the group.
Region N — Blackpool Area: David Jewett, 197 Victoria Road, East, Thornton-Cleverleys, Blackpool, Tel. Cleverleys, 869103. Informal meetings every 3rd Thursday at Arnold School, Blackpool.
Region N — Liverpool Area: Tony Bond, 27 Incre Rd., Liverpool 23, Tel; 051-924-1505. Informal meetngs every 2nd Thursday of every month at The merchant Taylor School for Boys in Crosby.
Region P — London - Barry Miles, Department of Business Studies, Polytechnic of North London, Holloway Road, London N7. This is the PET group of the Amateur Computing Club which meets every other Tuesday at the Poly at 6p.m.
Region P — West London - Please contact Geoff Squibb, 108 Teddington Park Road, Teddington, Middlesex. Tel: 91-977-2346.
Regions S — Durham Area: Jim Cocallis, 20 Worchester Rd., Newton Hall Estate, Durham tel (0385) 67045. Meetings at Lawson School, Burtley.
Region T — Scotland: Dr. Jim MacBrayne, 27, Paidmyre Crescent, Newton Mearns, Glasgow. Tel 041-639-5696.
Region W — Yorkshire: Bob Wood, 13 Bowland Crescent, Ward Green, Barnsley, S. Yorks. Tel (0246) 811585.

If there are any further queries please let me know.

My first view of the system was as one of a group huddled around an 8032 in the heart of Commodore's software department. With us were Dave Parkinson and Mike Bolley of Ariadne Software Ltd., the company which has provided the implementation of PETNET on Commodore equipment, ADP Limited have provided the host facilities.

As the system operates over the Public Switched Telephone Network subscriber lines one is very dependant on quality of reception of your particular telephone, Slough does not have the best of lines, but nonetheless Dave Parkinson was able that day to amply demonstrate the concept of PETNET.

Using the Commodore 8010 modem (300 baud, synchronous) we logged onto the host, and proceeded to watch a rundown of the system. At this point you realise that standing around in a circle watching a demonstration is not the best way to view a new idea such as this, so we shall skip forward in time to the day when I acquired an 8010 for my own office, and was able to get my system password, user number and user password.

First Explorations

Initial encounters with PETNET were frustrating, due to a number of different factors. Not least of these was the inherent difficulty of getting to grips with a new product without having access to a manual. In PETNET's defence, it is so robust, and well organised architecturally, that after a short while a manual becomes unneccessary. There is very little (if anything) that you can do wrong within the system: menu driven most of the time, it's easy enough to find your way around.

Other early problems were the telephone itself, and choosing the right time to log on. However, you soon learn when to try and when not to, and the telephone seems to have cured itself of whatever it was suffering from: PETNET was ready for access!

After a succesful logon, and acceptance of passwords etc., you are presented with a menu of 7 options, and usually one would first of all go to the 'directory' of information that is available. The tree structure comes into affect here, for by taking the general directory entry heading 'Education', say, a short wait will bring up onto the screen all the subheadings under that section, and you can then branch further.

At present, as PETNET is still really a development system, there is not an over-large amount of material there. However, it is growing all the time, and it is up to you, the user, to contribute to that growth.

One area of PETNET that is expected to expand fairly rapidly is the availability of Public Domain software, and the ability to download such software onto your own PET. Whether or not there will be a price charged for this is not as yet certain. If there is, I am assured that it will be kept to a bare minimum. Most of the information presented though will still be free.

Screen Editor

There are various other options available on the menu, but the one that would probably be used most is 'Exit to Editor'. This allows you to go to the PETNET Screen Editor, for use in preparing messages, information etc, for subsequent uploading to the host computer and consequent access for other users. At this point it is as well to disconnect your 'phone, otherwise you'll be paying for a call that you're not really making. You can always log on again later.

Again menu driven, the screen editor presents seven courses of action, and there is a further 'help' screen if you find yourself completely lost. Principal functions will be the designing of message 'frames', and these are just typed in direct from the keyboard. Frames can be saved to disk for later uploading, or earlier frames can be loaded from disk into the PET. Thus, an earlier downloaded item could have been downloaded (and thus saved) to disk, for viewing at some later date, without having to once again pay for the cost of the telephone call.

Summary

PETNET is an exciting new development for Commodore, Commodore equipment, and users of such equipment. The ability to very easily gain access to a large amount of information, and a large amount of free software, is a significant step forward. PET to PET communications are also well under way now, so we have yet another string to our bow.

Ariadne Software Limited, and Commodore's Nick Green, are to be congratulated on producing such a robust system. If at times you feel it's running low, bear in mind we're only operating on 300 baud: almost completely error-free however, which is fair compensation.

This page will be used to regularly keep you up to date with PETNET developments, or indeed you can go direct to PETNET itself. For further information on this subject, write to:-

Commodore Business Machines
675 Ajax Avenue
Trading Estate
Slough
Berkshire

FORTH

Until about a year ago, I was a dedicated BASIC programmer, anything was possible just so long as speed was not too important. However speed is important for a great many applications and so I drifted into machine code.

Writing machine code can be soul destroying as bugs are elusive and very time consuming to track down. The larger the program the longer it takes to delouse it.

I had been using a version of FORTH for a few months and was beginning to like the freedom and power of the language. The resulting code was fast and easy to write.

I was still writing machine code and felt that some of the aspects of the stack could be incorporated into machine code programs, so I wrote a simple machine code stack processor.

This grew after a few weeks into a rudimentary FORTH system aimed at high speed execution. Now a few months later the complete FORTH system is ready.

So what is FORTH?

FORTH is a structured high level language with very powerful commands which can be linked together to form totally new commands which are specific to the application. Thus it is not necessary to provide commands for making music, controlling the user port or plotting hi-resolution graphics; these can all be built up from the general commands to provide a far more specific set of commands which would otherwise not be possible in the space available.

FORTH has three features which programmers who have used only BASIC will not have come across before. These being:-

The dictionary

FORTHs dictionary is like any dictionary and consists of a list of words and their definitions. Each word is linked to the previous word by a pointer and for this reason FORTH is called a threaded language.

Microsoft BASIC does not require delimiters between functions:-

```
10 FORCO=1TO1000:PRINTCO;:NEXT
```

While this is compact code it is difficult to read and most programmers add spaces to enhance readability. In FORTH the space is used as a delimiter between words, just as with English. The above example in FORTH would be:-

```
1000 0 DO I . LOOP
```

The words DO I. and LOOP are all held in the dictionary.

Rather than having a series of statements executing sequentially by line number as with BASIC, the FORTH programmer adds new words and their definitions to the dictionary. New words are always defined in terms of words which are already present in the dictionary. For example:-

```
: LOOPER 1000 0 DO I  LOOP ;
```

This would define a new word in the dictionary called LOOPER. The colon ':' at the start tells FORTH to compile the following name as a new dictionary entry, linking it into the rest of the threaded dictionary. The compiler then adds the following words as the definition. When it comes to the 1000 it searches the dictionary for the word and when it can not find it an attempt is made to convert it into a 16 bit number. This is then compiled as a literal. The process continues until a semi-colon ';' is encountered, which terminates the definition and switches the compiler off.

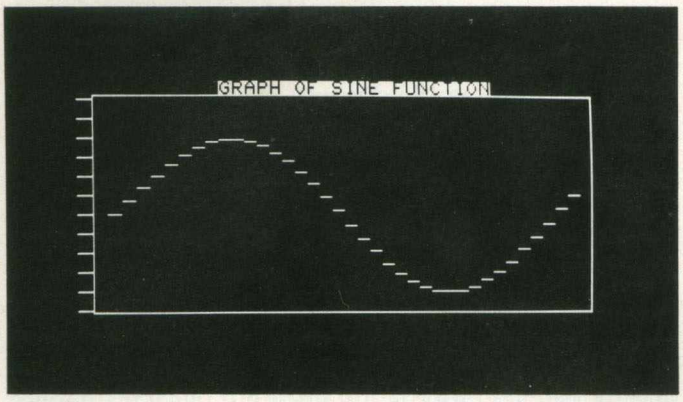
The version of FORTH I have designed compiles down to machine code. To show how compact it would be here is how the compiled code looks:-

```
        $86,'LO',11
        .WOR LINK
        JSR LITERAL
        .WOR 1000
        JSR CLITERAL
        .BYT 0
        JSR DO
LOOP 1  JSR I
        JSR I
        JSR DOT
        JSR LOOP
        BEQ LOOP1
        RTS
```

Notice that the name of the definition has been reduced to only 4 bytes. The first byte gives the length ORed with $80, followed by the first two characters of the name and then a checksum of the entire word. As far as I know this is unique amongst FORTH system. To the user and application programs this is transparent as is the fact that the compilation is directly to machine code ; most versions of FORTH use an interpreter.

The stacks

Standard FORTH (as defined by the FORTH Inc. in California) uses two stacks, a parameter stack and a return stack. In the 6502 version of FORTH the parameter stack is in zero page and the return stack is the normal 6502 stack. However, because the version of FORTH I have designed compiles



right down to machine code subroutines the 6502 stack can only be used with great care. For this reason a third stack is used as the return stack. Once again, this is transparent to the user.

Forth communicates mainly through the parameter stack (henceforth called 'the stack'). The return stack is used for controlling the duration of loops and for temporary storage of data.

All the stacks used in this system are termed last-in, first-out (LIFO). The stacks can be thought of as a pile of plates when only the top plate and its contents is visible, if the top item is removed then the plate beneath becomes visible. When the stack is empty it has a high value and as each new value is placed onto the stack the stack pointer is DECREMENTED.

5	8	top item
3	5	
7	3	
	7	bottom item

There are numerous words for manipulating the stack so that items can be recorded. For example:-
1 2 3 4 ok

The user types the numbers and presses return. The FORTH interpreter converts the numbers into 16 bit binary and pushes each on to the stack. The 'ok' at the end of the numbers means that everything up to the carriage return has been executed successfully.

. . . . 4 2 3 1 ok

The word '.' (dot) takes the 16 bit value at the top of the stack and prints it.

Here are some examples:-

1 2 3 4 SWAP . . . . 3 4 1 2 ok

1 2 3 4 ROT . . . . 2 4 3 1 ok

1 2 3 4 OVER . . . . 3 4 3 2 1 ok

1 2 3 4 DROP . . . . 321 ok

SWAP exchanges the position of the top two items.
ROT moves the third item to the top of the stack.
OVER copies the second item to the top of the stack.
DROP removes the top item from the stack.

Reverse Polish Notation

Firstly RPN is far simpler than its fearsome title suggests. Because the stack is the central feature of a FORTH application it is sensible to use a maths system that makes the best use of it.

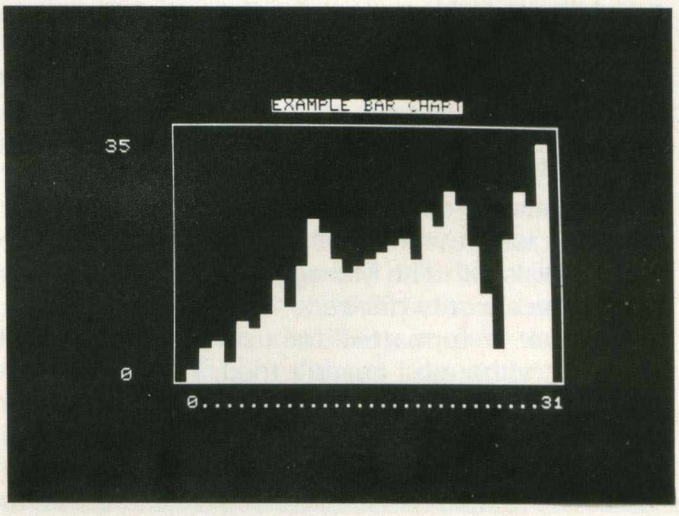
Here are a few simple examples :-

Algebraic	RPN
10-1	10 1 -
10+3+7	10 3 7 + +
(2+3)*(5+6)	2 3 + 5 6 + *

Running through the third example 2 and 3 are pushed to the stack '+' adds the top two stack items leaving 5 on the stack. Now 5 and 6 are pushed to the stack and added together leaving 11 on the stack. Finally the '*' (called STAR in FORTH) takes the 5 left from the previous operation and multiplies it by 11 leaving 55 on the stack.

It takes a while to master stacks and RPN notation but the results are worthwhile. I hope this brief visit to FORTH has roused your interest. In the next few issues I will be going into the details of programming in FORTH.

FORTH for the VIC for use with cassettes is available from Nick Hampshire Data Systems at 59.95 pounds as a ROM pack with 3k of internal RAM. FORTH for the PET also from Nick Hampshire Data Systems on disk at £59.95



The Manager....

The idea of using a computer in the home or place of work is, or should be, to make life easier for yourself and for your staff. There is absolutely no point in installing a system that is simply going to increase the work load. The common office practices that lend themselves to computerisation are stock control, word processing, payroll, accountancy and so on, and there are many commercially available packages that cover these demands. The usefulness of a data base system is perhaps, at first glance, rather more difficult to explain. This begs the question; what is a data base?

What is a Data Base

It is essentially an electronic filing system, enabling you to store information in any form you desire. Good data bases allow you to subsequently interface that information to other packages, for use in many varied forms. For example, stock control figures developed on the data base could be linked into a word processor for standard letter output to various clients, incorporating those stock control figures. There are many other such examples.

Having decided that you can store information, you then have to make the decision as to what format you store that information in. This will obviously depend on your own particular application, and there should be a large degree of flexibility in the program being used that allows you to alter, if necessary, your format later.

Finally, there should be various arithmetical functions within the program that lets it become more than just an electronic filing cabinet, and gives you the ability to truly manipulate information as you require.

And so, having decided what we want, onto The Manager.

Contents

The Manager comes in a padded four ring binder, complete with two manuals, a command summary chart, a disk with The Manager suite of programs on it, and the security device.

The disk is formatted for use on an 8050 disk drive, but although I haven't tried, I would imagine that it would also work quite happily on a 4040 or an 8422, once you'd copied over all the programs. Other disk drives would not be suitable as a) information is stored in relative files and b) you need two disk drives active for a fair amount of the time you're actually using the package. It only works on the 8032 CBM machine, and will accomodate Commodore and ASCII printers.

The first thing (quite sensibly) that the manual tells you to do is to take a backup copy of the disk, as only one disk is supplied with the package. You'll need further disks for data storage, as although you can use the program disk this is not recommended, as you'll rapidly run out of space.

You can make as many copies of the program disk as you like, but as the program has a protection device, you'll still only be able to use it on one machine. This protection device is yet another 'dongle', a little bit of hardware that sits on the cassette port of your CBM; the program periodically checks that the dongle is there. This method of protection does pose problems: so many commercial packages currently available use a dongle that swopping from one program to another can at times be quite difficult. The idea of a family of dongles nesting at the back of your CBM might prove popular to naturalists, but to the average computer user this will not be the case. Perhaps this might be the time to pose the plea to some hardware company to produce a board that enables a number of dongles to be used at the same time; it's been done before for security ROMs, so, is anyone listening out there?

The Manual

Commodore are to be congratulated on making the effort at producing a good manual. However, having said that, I have a number of reservations.

The two manuals supplied (User Guide and Reference Manual) constantly refer to each other, and within the individual manuals there is a lot of jumping about from one section to another, so I feel that the first time user would have quite a difficult job finding his way around. For the beginner there are many useful tips and guidelines about Commodore machines, storage media, and so on, but as far as an introduction to The Manager goes it leaves something to be desired. Aesthetically pleasing to look at, and the index is a great help, but it lacks a consistent introductory approach. Nonetheless, a significant step in the right direction.

The Program

The Manager is a menu driven suite of programs, presenting initally 16 main options. Of these, one is simply an exit to Basic, which performs essentially a SYS 64790 warm re-start, one backs up your data or program diskette, and a third is just used to format a diskette.

I'm a firm believer, when reviewing a package, to

attempt as far as possible to make the product 'crash'. In other words, doing something which you're not told to do, but which, at the same time, you're NOT told NOT to do. For instance, formatting a diskette; as far as I can tell, every other use of the disk drive checks to see whether a diskette is actually present in the drive. This option doesn't! In other words, you spend an awful long time waiting for The Manager to get back to normal. Everything else seems just about 'uncrashable'.

In its defence, the manuals do give you details of known 'bugs' in the system (two at present), and invites you to write in to Commodore with any others you may discover yourself.

The most commonly used option will be the one that allows you to enter and edit your data. This is drawn up on a blank screen, with you choosing exactly how you care to layout your information, defining everything to be used in terms of 'fields' of information; these are then used later. The normal CBM screen editing controls work here, without the annoying 'insert mode' coming into effect. Once you are happy about your screen layout you simply go ahead and enter your data, various files are set up, and your data is stored to disk for subsequent later retrieval and manipulation.

Essentially, what you will have created is your electronic filling cabinet. Your raw data is there, it can be retrieved at any time, and it can be added to and amended as you wish.

The file I set up in testing The Manager was a simple but effective one. It was basically a list of real ale pubs in London, along with their address, beers sold, amenities available, and so on. The Manager quite happily coped with requests like 'give me all the pubs in the West End, that sell Samuel Smiths OBB, and have video games in'. That information could easily be transferred to something like 'give me all the houses currently for sale in Aylesbury, costing under 30,000 pounds, with three bedrooms' etc. I think that from those two examples you can rapidly gather just some of the capabilities of The Manager.

These reports can equally well be printed to the screen, or a suitable printer, but printing to the screen has one or two drawbacks. For instance, the request 'give me all the pubs ....' prints all those that sell Samuel Smiths OBB whether or not they have video games. On the printer, there is no such problem.

There is also a pseudo 'not' search. That is; 'give me all the pubs that DON'T sell ....'. It can be done, but one has to be careful about initial entering of your data.

I would imagine that these would be the most frequently used options of The Manager, but there are a number of other available. All files can be sorted on any field within your file, using a number of options. All files can be globally updated, and this can be done on either alphabetic or numeric fields. All fields can be 'sub-filed' for later use with a word processing package; a valuable feature, which can save a lot of office time.

Combine all this with the ability to extract relevant information, perform both alphabetic and numeric manipulation of that information for printed reports, and you have a relatively sophisticated 'data base' package: certainly one that comes well into our definition presented earlier.

There is an additional point to be made. Large commerical packages like this ought really to be written in Machine Code, or at least compiled from BASIC. BASIC has it's inherent difficulties, not least of which is slowness, and when a lot of work is being done this slowness can become a hindrance.

Conclusion

The Manager is NOT all things to all men; no package can be. There are one or two things I would have liked to have seen implemented, one or two things I would like corrected (user friendly can only be taken so far), and the documentation could certainly do with improvement. However, what you pay is what you get, and 275 pounds represents a reasonable selling price for this package.

Technical Information

This is related to an 8032/8050 configuration.

Record format; user defined; up to 120 data fields over two input screens; display ability to calculate data and store it on record, or display up to 80 calculated results; field lengths of up to 79 characters.

Record storage; record lengths of up to 253 characters; number of records up to 16,000 (approximately 720 at maximum record size); as many data files as you need.

Data Manipulation: arithmetic capability with 99 registers for intermediate results; multi-key sorts with up to 16 keys; sub-file creation; global update facility; link to word processor.

Selecting and Reporting: sort in ascending or descending alphabetic or numeric order; up to 20 selection criteria per report; data from one record printed on one line or up to 60 lines; output to 80 or 132 column printer, diskette or screen; ten levels of breakpoint with or without sub totals; multiple reports can be defined for any data file.

It is available from Commodore's nationwide network of dealers.

VIC 1540 Disk Drive

The ability and usefulness of any microcomputer is dependant on many things, not least of which is, of course, the microcomputer itself. Given that we have settled on a particular computer, one ought to look at a) the amount and quality of software available, and b) the number and quality of peripherals that are also available. Those two options are not intended to be in any order; the order must be decided upon by yourself, depending on what you require your computer to do.

Looking at option b), and looking at the VIC as our computer, there are indeed a large number of peripheral devices available and easily connectable. Here we concentrate on the VIC disk drive, designated (obscurely, but then aren't they all?) as the VIC-1540.

First and foremost must come the question 'do I really need a disk drive '?'. Hopefully this review will enable you to answer that question.

The VIC disk drive comes priced at 396 pounds, including VAT, which could be argued as an expensive price to pay. Indeed, at twice the price of the basic VIC there is no other real way of looking at it. So, what are you getting for your money?

The Basic Unit

The disk drive is conveniently packaged in a fairly smart box and comes complete with manual, test disk, connecting cable from the VIC to the drive, and a mains lead (but no plug on the end? Surely manufacturers can afford plugs! At 396 pounds they ought to be able to). Since it is the manual that is going to help you get the best out of the disk drive, let's take a look at that.

It is regrettably an almost inevitable comment where Commodore manuals are concerned, but there is still a great deal of room for improvement. Certainly their manuals have been getting better as time goes by, and indeed the U.K. division of Commodore (and one assumes the others as well) now has a Technical Publications Manager. However, they have a long way to go! In fairness, this paragraph could probably be equally applied to any other microcomputer manufacturer, but that is no excuse.

The manual IS good where the basics are concerned: how to unpack the unit, connect it up, see that it's working, and so on. It's when it attempts to go any further that it begins to flounder somewhat. For instance, there's a page devoted to telling you how to copy files from one drive to another. All very well, but the VIC disk drive is a single drive! Even connecting two VIC disk drives up in parallel would not solve the problem, as you'd have to change the device number rather than the drive number.

There are a number of other 'errors': the spelling mistakes, the sections that look like they've been almost literally translated from Japanese to English, and so on. But let's take a look at the disk drive in action.

The Disk Drive

The VIC disk drive has a very pleasing aesthetic appearance; standing just 10 centimetres high by 20 centimetres wide, it is a smart looking piece of equipment. Taking standard 5 1/4 inch diskettes, it has a capacity of just over 174K, stored in the form of 664 blocks of information. It is read/write compatible with the existing Commodore 4040, and read only compatible with earlier 3040 and 2040 disks. Disks formatted on an 8050 haven't got a chance!

Files can be stored either as program, sequential or relative files, although relative file handling is rather difficult to perform, as the VIC itself does not have the DOPEN etc. commands. Sequential file handling is (if you'll excuse the pun) relatively easy to operate, and goes a long way towards turning the VIC into a business computer. Program files will presumably be the most used option, so we'll take a look at those first.

Programs, as you know, are entered into the VIC from the keyboard and subsequently SAVEd to tape or disk to save re-typing it in again. Those of you who've used tapes will realise how slow it is to SAVE and LOAD programs. The disk drive is significantly faster, and a recent estimate of speed is of the order of 250 characters per second data transfer rate. Once you get used to the speed of a disk drive it becomes almost impossible to go back to tape. A further disadvantage of tape is that if you want to load, say, the 5th program, you have to wander through the other four programs that come before it on the tape: either that or keep a careful eye on the tape counter! The disk drive goes straight through to the program you require.

Also, there is no way of knowing on the tape precisely what a program is called. Each disk has a 'directory' of all the programs on it, so that you can instantly see what the name of each one is. One

word of warning: without some additional software in the VIC, calling up the directory will erase your program, so make sure you save it first!

With the disk unit comes a sample disk containing 7 programs (the listings are displayed in the manual if the disk fails to operate, although mine worked quite happily). There is a form of DOS support, to simplify the use of using the disk in immediate mode, but not in program mode. There are a number of programs to check out the performance of your unit, and look at various information stored on the disk, and finally demonstration sequential and random access programs.

File Handling

To go onto sequential and relative file handling requires the use of rather more commands than the cassette user is used to. Here the manual is rather good, as it gives a description of each command, and an example of that command 'in action' as it were. There are sample programs for sequential and relative file access, which, although not offering the best quality of reproduction, are certainly readable and, more important, usable.

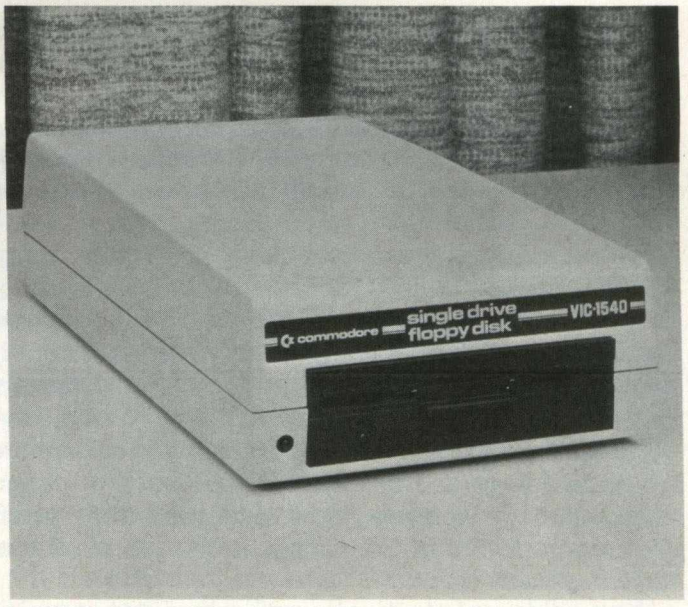
Both forms of file handling are fairly straightforward, and the beginner should have no problems in quickly getting to grips with them. Relative (or random, same thing) is particularly a boon: tape decks just weren't built for this! As with most options on the disk, after just a short while in use you'll be amazed at how you ever managed with a cassette deck.

I would recommend studying the listings for this sort of programming carefully, and don't be too bothered with the descriptions of precisely how information is stored on the disk: that can come later. Just follow the listings through, and establish for yourself why certain code is happening when it is, Error checking, for instance, will be a relatively new experience for most people, but in using disk drives it is vital, so find out why and how it's done.

Having mastered file handling, the manual moves onto (and so must you) the disk utility command set, covering such items as block read, write, execute and allocate, and memory read, write and execute. Again examples are given of each command being used, and they are clear enough for you to soon get the hang of them. The manual must be gone through with you sitting next to, and using, the disk drive: hands on experience is vital.

Conclusion

The VIC Disk Drive is a welcome addition to the VIC range of peripherals. Its neat, compact design fits well into the VIC family tree, and the ability to very quickly LOAD and SAVE programs and files is, almost, a necessity. I've yet to discover any 'serious' bugs in it: one or two idiosyncracies perhaps, but those are only to be expected. The documentation could be better, but it is certainly adequate from a beginner's point of view. The read/write compatibility with the earlier PET 4040 disk drives is extremely useful, and the ability to read disks formatted on 3040/2040/2031 disk drives is similarly useful. Of my two main complaints, one is inherent in it being a single disk drive: you can't, other than tediously, make a backup copy of a disk! My main complaint is price: at 396 pounds it IS expensive, and many people may think twice before purchasing.



The VIC 1540 Disk Drive

Technical Information

Operating version of DOS: DOS 2.6

Total Capacity per disk	: 174848 bytes
Sequential	: 168656 bytes
Relative	: 167132 bytes
	: 65525 records per file
Directory Entries	: 144 per diskette
Sectors per track	: 17 to 21
Bytes per sector	: 256
Tracks	: 35
Blocks	: 683 (664 blocks free)

2K RAM on board, in other words it is an intelligent drive, and takes no memory from the VIC.

Book Review

Three new books this time. One from Ellis Horwood Publishers in their series of 'Computers and their Applications', one from our very own Nick Hampshire Publications, namely the long awaited 'VIC Revealed', and the third from Prentice Hall International, called 'The PET Personal Computer for Beginners'.

Microcomputers in Education

The use of microcomputers in the educational field has grown tremendously in the last few years, and it's fair to say that the PET has contributed more than most to this growth. The fact that a conference on the use of microcomputers in education could be organised around the PET says a lot for their use in schools and colleges: the book 'Microcomputers in Education' is the result of the first such conference.

The editing of the book, and indeed the organising of the conference, was done by Dr. Christopher Smith, of Queen Elizabeth College, University of London, where he lectures in Physiology. He is to be congratulated on putting together a clear, concise and well informed book.

The subject matter is a collection of the talks given at the conference, which was held in September of last year. These are broken down into three main fields: the development of computer assisted learning, languages for microcomputers in education and applications of microcomputers in education.

Each field is then broken down further into five or more individual sections. For instance, the applications of microcomputers in education covers PETNET, controlling equipment, school administration and so on. There are many distinguished speakers presented here: Borge Christensen, Bob Lewis, Danny Doyle, Nick Green, Dr. Smith himself, and many others.

Whilst each of the contributors is something of a specialist in his or her own field, they all have a valuable contribution to make to the role of microcomputers in the modern educational world.

It is this that makes the book worthwhile. Any school teacher who is interested in exploring the possibility of investing in micros would be wise to take a look at this book first, as many avenues which others have already gone down are discussed in some detail. Thus any potential trouble spots could be gone over before any financial commitment were made.

Similarly anyone who has already got micros to hand at a school could do far worse than buy a copy of the book. The numerous ideas that are discussed are far more than any other individual could ever investigate, and as I stated earlier each contributor can be regarded as an expert in his or her own area. Consequently, each of their contributions makes for interesting, and stimulating, reading.

To sum up, a most useful addition to the large number of books on microcomputers.

Available from Ellis Horwood Pubishers, at a price of ten pounds.

The VIC Revealed

I make no apologies for reviewing an in-house publication, because with a book of this nature, and considering the microcomputer that it concerns itself with, it is important to know whether it comes up to the standard that everyone was expecting.

We all know (or should know!) Nick Hampshire's The PET Revealed, the author's first contribution to the literary world of microcomputers. An invaluable guide for any PET user, it was the first book to deal under one cover with the entire range of PET related facts and figures. The success of that book led to two others, namely the Library of PET Subroutines and PET Graphics. With that experience behind him, the author has now returned with a book that sets out to explore the inner workings of Commodore's VIC 20 microcomputer.

As the author himself says. "This book is a collection of discoveries about the VIC, how and why it works, and how to use these facts to write better programs, and perform more interesting functions". Well, is it?

The short answer is yes! I think Commodore themselves would be the first to admit that the original documentation supplied with the VIC left quite a lot to be desired. Agreed, it was probably intended for the beginners market, at which the VIC must be aimed, but beginners learn rapidly, and once they'd reached the end of the manual there was nowhere else to go. Even that manual fell far short of what one would expect from a company that could arguably be described as the market leader in microcomputers. Too many errors, and most of the time merely hinting at what could be done.

So, the market was there for a book that could take the user further, and what's more explain in rather more detail than the Commodore manual how the machine actually worked.

The VIC Revealed performs these functions admirably. The book is broken down into five main sections: the 6502 Microprocessor, VIC system software, the 6561 Video Interface Chip, the 6522 VIA and the User Port, and the VIC I/O functions. Each of those sections is then sub-divided into many more, explaining in clear and precise terms the operation of whatever system topic it happens to be covering at the time.

Complete with an appendix of circuit diagrams, and more, and containing en route VIC memory maps, detailed notes on colour, sound, joystick control, high resolution graphics, and anything and everything you've ever wanted to know about the VIC but were afraid to ask, this book is invaluable for any serious VIC owner. It is difficult to think of a more comprehensive book on the market at the moment.

Available from bookshops at a price of ten pounds.

PET Personal Computer for Beginners

There are so many books of this nature available at the moment that one would have to be quite exceptional to single it out from the crowd. Unfortunately this one isn't.

Written by Seamus Dean and Valerie Morgan, and published by Prentice Hall International at a price of 4.95, pounds in paperback 7.50 pounds in hardback, it at least has the virtue of being inexpensive. And to be fair to the book it is trying to cover a very large area in being a totally beginners guide on using the PET. However, rather like a football team chasing the treble it tends to fall between three stools: assuming too much knowledge in some places, repetitively assuming too little in others, and occasionally throwing up some disastrous mistakes.

For instance, in a section on using the cassette deck, it takes you through 8 steps from initial program creation to final saving and verifying of your program. All well and good, until it tells you what to do in case of ?VERIFY ERROR, which is to go back to the first step: this is typing in the program! Okay, you or I may know that one doesn't have to type the program in again if a verify error occurs but the complete beginner, which is who we must assume this book is aimed at, could have many a frustrating hour at the keyboard before realising the book's mistake.

It is a pity that, out of all the books on the market at present Commodore should choose to make this one Commodore Approved. They would have done far better to have gone for something like R. West's Programming the PET/CBM, which we shall take a detailed look at next time.

Butterfield

This month Jim talks about PRINTING .....

The usual point of programs is that they produce output. The normal way of producing output is by using the PRINT command or its cousin, PRINT#. We can abbreviate PRINT with question mark (?), but oddly enough, PRINT# can't be shortened that way: typing ?# will produce a program line that lists as PRINT#, but doesn't work.

Other ways.

We can generate output without using PRINT. It's not always good practice, but we can POKE to the screen memory area. This can be good for graphic games and animations, but there are several bonus things we get by using PRINT. First, PRINT keeps track of the line position for us, and starts a new line as necessary. Secondly, PRINT doesn't limit output to the size of the screen: when the screen fills up, scrolling is automatic. Finally, and most important, PRINT can easily be changed to PRINT# to allow output to be directed to other devices such as printer, modem, disk or disk cassette tape. In contrast, screen POKEs are absolutely limited to the size of the screen, and can't be easily redirected anywhere else.

Punctuation

If you say 'PRINT X' you will print the value of X and start a new line. The absence of punctuation at the end of the PRINT command signifies, "That's the whole thing; print it and wrap the line up." In contrast, if you say 'PRINT X;' you will print the value of X but you won't go to the next line: the invisible cursor will wait behind the printed value. This sounds a little backwards: you do something extra if you have no punctuation, but you do nothing if you have a semicolon.

There's one other form of "formatting" punctuation: the comma. If you type 'PRINT X,' you will print the value of X and then skip ahead to the next "column". Columns are considered to start at positions 11, 21, 31 and so on up to position 71. They exist only on the screen; saying 'PRINT#4, X,' to send to printer or other device won't set up columns properly. The comma will produce quick and convenient output to the screen, but it may be a bad habit since you can't use it anywhere else.

We can use this punctuation within a Basic PRINT statement as well as at the end. 'PRINT A;B$;C;' will generate the values of variable A, string B$, and variable C one behind the other and will leave the cursor positioned behind the value of C.

Neat input

We can use this punctuation to generate prompting for INPUT statements. For example, if we wanted to add ten numbers, we might code:

```
100  PRINT "INPUT EACH NUMBER:"
110  FOR J=1 TO 10
120  PRINT J;
130  INPUT X
140  T=T+X
150  NEXT J
160  PRINT "TOTAL IS",T
```

We prompt for the ten numbers with 1? ... 2? ... 3? ... and so on. The prompting number is printed by line 120 - J is stepping from 1 to 10 - and the question mark from the INPUT statement appears behind it because line 120 ends in a semicolon; after printing the number we wait on the same line so that the question mark will appear there. Question: What would happen if line 120 ended with a comma instead of a semicolon? Try it and see.

Number formats

Numbers are printed in a special format. First, there is either a space for positive numbers or a minus sign for negative numbers. Then the number appears, as many digits as required plus a decimal point if needed and perhaps even "E" notation. (Never heard of E notation? Try PRINT 3E2 and see if you can figure it out). Finally, the number is followed by a cursor-right on the screen.

This seems at first to give you two spaces between numbers, but there are one or two fine points that are useful to know. If you type PRINT 2;3;4; you will see two spaces appear between each set of digits. Now try this: Type a bunch of x characters over the answer (a row of xxxxxxx...) and then cursor back to the PRINT statement and press RETURN again. Some of the x's don't go away; that's because a cursor-right skips over that part of the screen without writing there.

There are a couple of ways to eliminate this difficulty if it bothers you. If you change a value to a string before printing, the cursor-right won't be performed. You could type PRINT STR$(2):STR$(3):STR$(4) - the same number will print with at least part of the problem solved. If you

happen to have an 80-column or Fat-40 4.0 system, you may type: PRINT CHR$(16);CHR$(22);2;3;4; and you'll discover the problem is solved quite elegantly.

Here's another exception to the two-spaces rule: Type PRINT 2;-3;-4;5 and look at the result. The minus signs take up one of the two positions, and now there's only one space between some numbers.

Summary

PRINT is handy and versatile. It takes a little while to get used to the formatting of the PRINT statement, but you'll soon have good control over your output.

There are some fascinating things you can PRINT which cause the screen to do unusual things. More about them another time.

Producing Output

When you are producing output, it's good to make it neat. The computer is there to help its human readers, and the more you can do to improve the information, the better job you'll be doing.

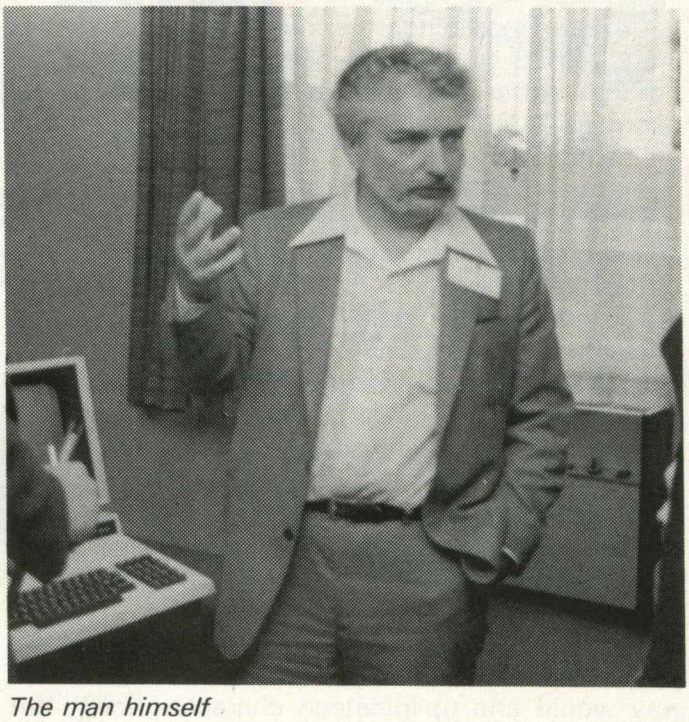
Printing in columns

Beginners often arrange values by using the screen tabulation functions: putting a comma into the PRINT statement, or using the TAB function. These methods work, but they have a pitfall: they won't behave properly if the output goes to other devices. The problem is that the computer always knows exactly where the screen cursor is, but it never knows on what column the external devices are located. It doesn't even try to keep track; so a TAB or a comma directed to the printer or other devices won't behave properly.

It's my feeling that almost everything that goes to the screen can be usefully directed to the printer, or written to a disk file with a view to transferring to the printer later. Once you have a report looking nice on the screen, you don't want to have to reprogram to get it looking nice in print. So ... stay away from TAB and commas - there's a better way.

Redirecting output.

While I'm on the subject of switching output from the screen to the printer, I'd like to share a little coding trick with you. Most programmers know that you can direct output to a printer by performing an OPEN to device number 4 (the printer) and then using PRINT//... That's fine for a finished program; but you can waste a lot of paper while you're checking out a program if you do everything to the printer.



The man himself

Here's the trick: We can OPEN to device number 3 (the screen) and PRINT// to the screen, checking our program and fixing it up. When it's ready to go, all we need to do is to change the OPEN statement so that it names device number 4, and output goes to the printer. We save time and paper. Let's try it: we code:

```
100  OPEN 1,3
110  FOR J=1 TO 10
120  PRINT#1,J;SQR(J)
130  NEXT J
140  CLOSE 1
```

When we run this program, output is delivered to the screen. If everything looks good, we can now change line 100 to OPEN 1,4 ... and output is redirected.

It's not really a trick; it's good coding. We could allow the user to specify what output he wanted by coding something like: 100 INPUT"DEVICE NUMBER";N :OPEN 1,N so that the user could type in 3 or 4 to select the type of output he wants.

Neatness counts

If I'm sternly discouraging TAB and the comma, how can you arrange things in columns? A few simple answers, but first some ground rules. The best way to arrange stuff in columns is to make sure that each "field" is always the same length; that way, each item will be printed neatly in the same place across the page.

Butterfield

How can we rechop two numbers as different as 3 and -32768 so that they occupy the same space? For that matter, how can we take two names as different as BUTTERFIELD and NG and make them the same length?

Let's take the names first. These "strings" could be neatly chopped down to a fixed length by means of the LEFT$(function ... if they were long enough. For example, we could slice out the first eight characters of string X$ with LEFT$(X$,8); but it won't work if X$ is less than eight characters long in the first place. So - pay attention - we must first pad out the name by adding spaces to the end. Sticking extra characters onto the end of a string is called "concatenation" - pronounced with emphasis on the cat - and is done with a plus sign. If we had a short name like M and wanted to tack eight spaces on the end, we'd do it by writing "M"+" " which would create a new string nine characters long. A name like BUTTERFIELD treated the same way would end up nineteen characters long, but this doesn't matter: we're going to chop them both down to the same length with LEFT$(.

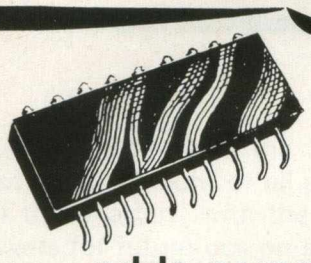
Getting it together

Let's put it all together. If the name is held in variable N$, we code PRINT LEFT$(N$+" " ,8); with a semicolon at the end. First we concatenate, adding the spaces; then we chop (or "truncate"), cutting to a fixed length; finally we print. Both long and short names will be printed as exactly eight characters; the next thing we print will be neatly lined up behind it. We might want to make the field more than eight characters long, since a splendid name like BUTTERFIELD would end up chopped to BUTTERFI - if we do increase the length we must remember to add more spaces, of course.

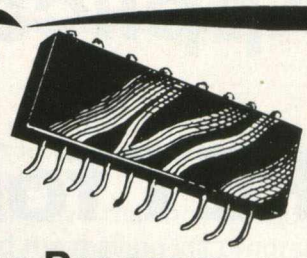
The above procedure is called Left Justification, since the strings are lined up neatly on the left with spaces filling out the right hand side. We can go the other way and produce Right Justification with a small adjustment: try PRINT RIGHT$(" "+N$,8); and you'll see how the the left side fills with spaces and names line up on the right. This is the kind of alignment you will want with numbers; we'll deal with that in a moment. Remember that if you don't allow enough space you'll end up with chopped-off names like TERFIELD, and there's no justification for that...

If the numbers you are using are integers, you'll usually want to line them up with right justification. Once again, this is easy to do once you know the function that changes numbers to strings. If your value is held in variable X, we can change it to a string with STR$(X); now we can do the right justification with PRINT RIGHT$(" "+STR$(X),6); everything will work out neatly. Study this statement and see how X builds up into a neatly justified string of length six.

If your numbers contain fractional values, you may want to try to line up the decimal points. That's much more challenging. Perhaps you'd like to try your hand at it. We'll tackle it here another time.



audiogenic LTD
CHIP SHOP



EDEX 2.0 & 4.1

adds commands to BASIC for use within your Program

IF THEN ELSE ● PLOT ● BEEP ● PRINTUSING ● SWAP MERGE ● HARD COPY ● PLUS A RANGE OF TOOLKIT TYPE FUNCTIONS AND A FAST EDITING SYSTEM

EDEX is an extension to BASIC which considerably enhances the potentialities of the Commodore PET/CBM It consists in a 4K-BYTE ROM which installs inside the PET/CBM.

EDEX is compatible with Commodore disk devices as well as with the DOS Support Program.
EDEX operation is fully transparent towards the Microsoft Basic Interpreter
EDEX is fully compatible with prior programs written without EDEX.

AUTO
Activates automatic line numbering.
APPEND *
Allows the creation of a program with a subroutine library
BEEP
Gives a sound of programable pitch and duration
CALL
Calls a machine language subroutine with transmission of up to 16 arguments
DELETE
Allows multiple line suppression
DUMP
Lists all variables in a program, together with their values
EDITING *
e.g. @ M prints MID$
ERROR
Shows where an error has occurred
FIND
Lists all lines where a given character string is present

HARD COPY □
Dumps screen to printer
IF THEN ELSE
With up to 16 nested conditions
MERGE □
Merge two programs files
PLOT
Plots curves of 50 x 80 or 160 resolution
PRINTUSING
Formats printing on screen or any printer
RENU
Program renumbering
RESET
Suppresses a dot (contrary of PLOT)
SWAP □
Swap one program for another keeping variables
TRACE □
Single line execution (displayed at top of PET)

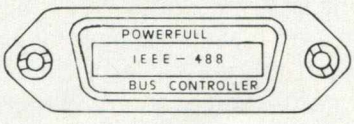
*** EDEX 2.0 only □ EDEX 4.1 only**

EDEX 2.0 for use with BASIC 2 40 Column Pets **£39.50**

EDEX 4.1 for use with 80 Column Pets **£49.50**

Available shortly for BASIC 4 40 Column PETs

IEEE-488 PACK



The end of instrumentation's problems. It resolves all kind of troubles:
- Time-out
- Special characters ("null", and so on...)

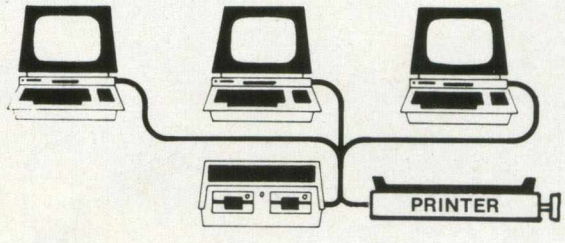
IEEE-PACK allows the use of IEEE-488 Universal Commands:

- DCL (Device clear)
- SPE (Serial poll enable)
- LLO (Local lockout)
- PPL (Parallel poll configure)
- SDC (Selective device clear)
- SPD (Serial poll disable)
- GTL (Goto local
- PPU (Parallel poll unconfigured)

IEEE-PACK also allows BASIC interrupt with functions:
- ONKEY "x", line number
- ONSRQ line number (On Service Request)

IEEE-PACK comes complete with two ROMs. **£89.50**

MULTEX



MULTEX allows several CBM 8032 to work together on the same peripherals.

MULTEX is a ROM which replaces a ROM of the CBM 8032.

Except the substitution of this ROM no other modification is required on the CBM 8032.

MULTEX is much cheaper than any other system.

MULTEX £69.50

Access DINERS

ALL PRICES INCLUDE V.A.T. & P.P.
AVAILABLE FROM ALL GOOD DEALERS OR DIRECT FROM

AMERICAN EXPRESS BARCLAYCARD VISA

AUDIOGENIC, P.O. Box 88, 34-36 Crown Street, Reading, Berks. Tel: Reading (0734) 595269

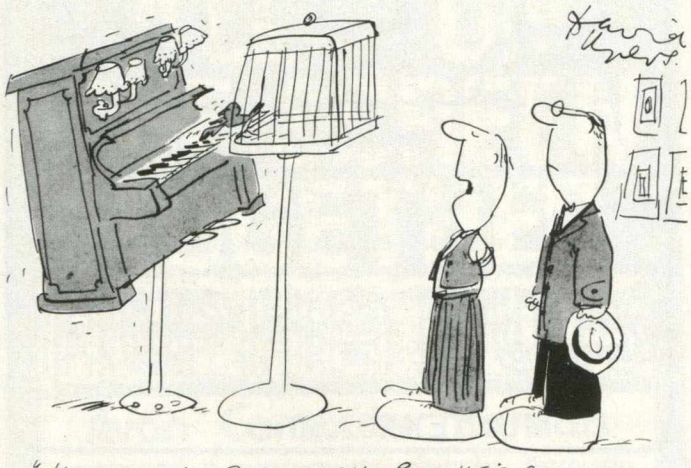
PET for the birds.....

A year ago Richard Williams, Managing Director of a £200,000 company based on birds was thinking about buying a computer.

Today he is an enthusiast, and his Commodore PET 8032 is an integral part of his business - even controlling the feeding programs of the thousands of exotic parrots, which form the core of his business.

Mr. Williams runs Bosley Farm Aviaries of Christchurch, Dorset. At first glance he seems to just buy and sell birds - and that doesn't seem to need the services of a micro computer. But Bosley Farm is an international conglomerate - in miniature. It does buy and sell birds in seven or eight overseas countries. In addition they manufacture special sheds and panels for aviaries; run a retail outlet, and sell services, birds and manufactured items by direct mail throughout the United Kingdom.

Bosley Farm is an efficient operation. The export earnings of the company are around £35,000 per employee - and central to that efficiency is the Commodore PET and its Compsoft DMS, and Wordpro 11 programs. Until the 8032, with its 3022 tractor printer, 3040 floppy disc drive and 8026 typewriter/printer were installed, neither Mr. Williams nor any of his staff, had any experience with computers.



"HE DOESN'T SING MUCH BUT HE'S GREAT ON PIANO...

Because of circumstances, installation was a hurried affair and their instruction in the use of the equipment was sketchy - their skills are largely self-taught. "It wasn't hard. We had the instruction book and we just kept at it. Knowing nothing was an advantage. I still don't know how it works, and I don't care... I can make it do what I want. Compsoft (our Commodore dealer) seem surprised at what we make the DMS program do. In a way they're rather proud of us - but if we have achieved anything it is because the PET and the program are so flexible. We think up a use for it, then we think of a way of bending the system to do what we want it to. It is a 'mops and bucket' type of tool for us. None of us are experts, and while aptitude varies, all of the staff can operate it - and do when they need it, "Mr. Williams said.

The many programs that Bosley Farm Aviaries use on its Commodore PET include one to monitor the feed/cost efficiency of the birds (which can be constantly updated according to the amount of stock) allowing the company to keep a check on both the amount in weight consumed by the birds and its value, per bird. A refinement keeps a check on the consumption variation according to temperature - the lower the temperature, the more feed required.

Stock control is important, and probably consumes more computer time than any other use. It is kept on a daily basis, and as well as keeping a record of stock, ordering replacements, keeping track of bird deaths, and items such as budgie toys, it also allows them to keep a simple running check on turnover. takings, VAT requirements, even allowing comparisons with the same week in previous years.

Financial and VAT management form separate programs, but the way in which Bosley Farm uses their stock control gives them an idea of what is going on - all the time. Useful for impressing the bank manager, according to Mr. Williams!

The company has recently set up its own credit system, and a credit control program used by Mr. Williams not only monitors the state of each account, but also projects cash flow figures from this income source for months ahead.

The manufacturing side of the business, which makes aviary panels, and has just gone into the making of complete sheds as well, uses a similar stock control program. It enables them to forward plan, so that they make the required number of items each month - how many panels, how many doors - the ratio is about 6 to 1, a fact that can be

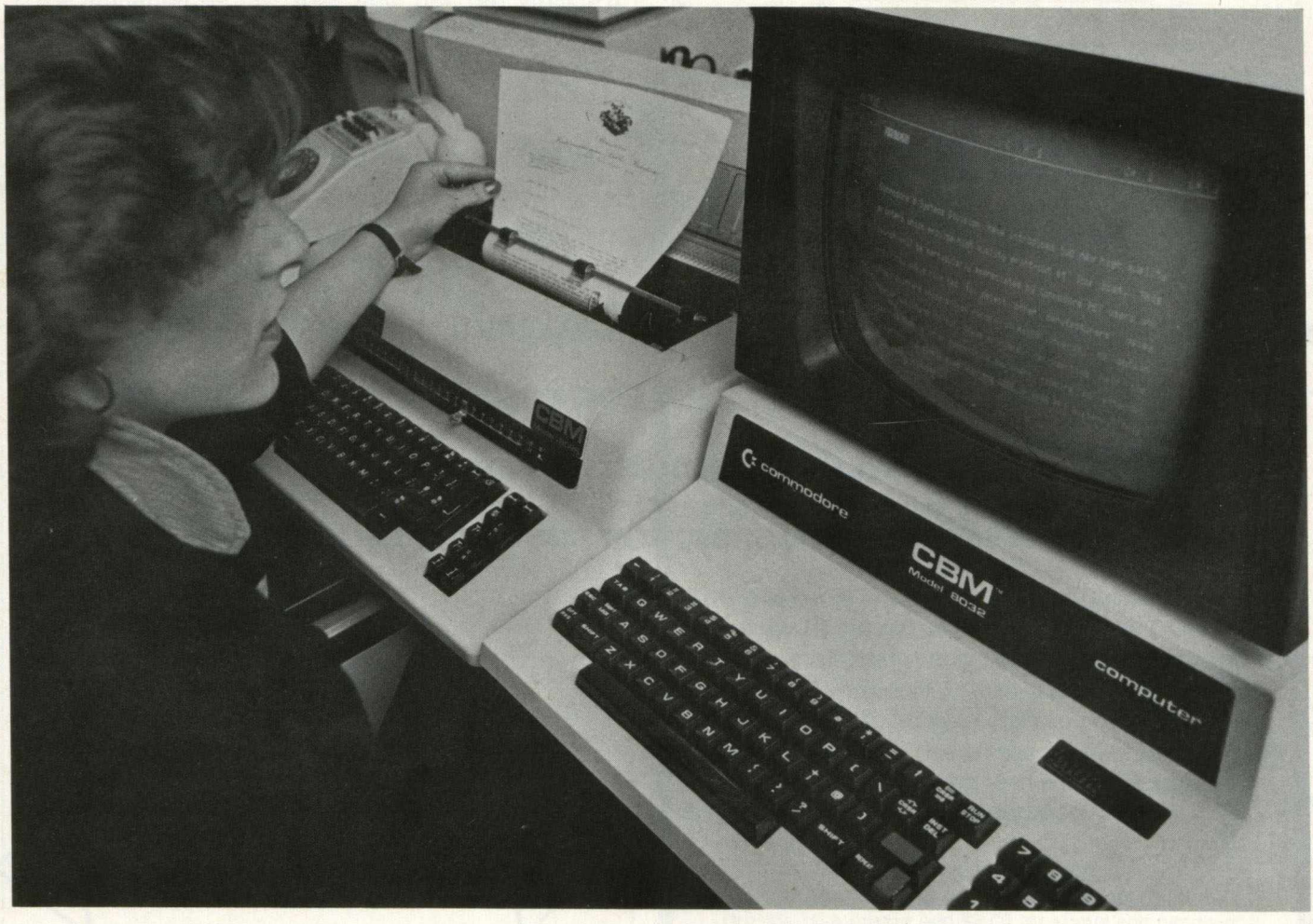
seen at a glance on the computer print-out. They also keep a record of all sales of panels (and birds for that matter), with the names and addresses of buyers for future use on special mailings. To make the maximum use of these names and addresses, Bosley has bought an interface to allow the use of the records in their credit and sales files, for mailings on Wordpro, without the need for transferring them to a special list.

The word processing function of the PET is important to them. They have a special mailing list of people who pay to receive up-to-date information on stock and prices, which is quite separate to any special advertising mailshots that they undertake. Because of the numbers involved, Bosley do not use the Word Processor to type all the lists, but use it instead to cut stencils - a feature that Mr. Williams hasn't been able to find in the instruction book, but which works very well - and because mistakes are corrected on the Word Processor, the stencils are cut without error.

"While the copies are being run on the duplicator, the PET is printing out the address labels on the tractor printer. It does about 8 per minute - it doesn't sound a lot, but it is almost 500 an hour... at that rate it doesn't take us long to prepare a complete mailing of all our clients."

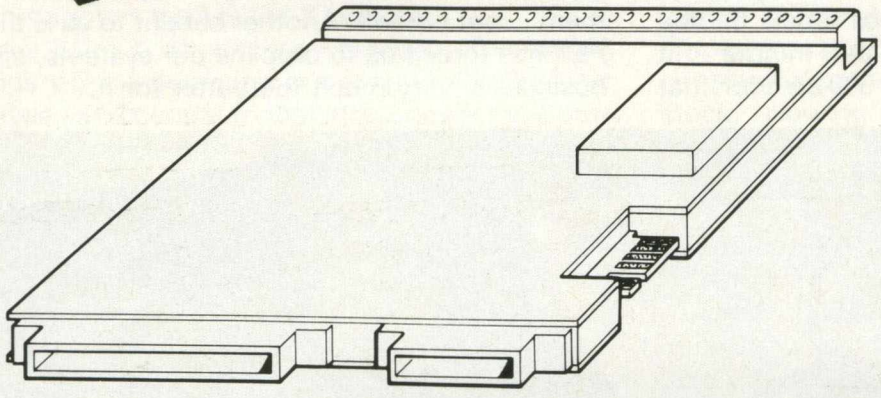
Bosley are able to do selective mailings because of the ability of the PET and the mailing list program to sort the lists according to a number of different criteria. It can select all names of clients who have bought Californian Quails, or, say, live in a specific geographic area, for example - allowing Mr. Williams considerable flexibility when planning his advertising. Beyond that, Wordpro has on store a number of stock letters, price lists, internal stationery, and specifications of manufactured items, which can be altered or up-dated with little effort. This ensures that all material which is dispatched prospective clients reflects the position on the day of dispatch - an important feature in a business where the stock is so varied, and changes as quickly as aviary birds.

According to Richard Williams, "Any small business has some use for a micro computer. Already we can see other uses for it, but have not had the time to devise a way of making the PET do them... but we will. Another benefit to us is that the PET has forced us to disipline our systems, and our business is very much the better for it."



THE PUPI

NEW LOW COST
12 BIT ANALOGUE I/O & DIGITAL I/O
FOR "PET"* USER PORT



The "PET" user interface that everyone can afford. TREAT YOUR "PET" TO A PUPI.

★ 4 ANALOGUE INPUTS (±10V) 12 BIT ★ 2 ANALOGUE OUTPUTS (±10V) 12 BIT ★

★ OPERATING SYSTEM IN EPROM ★ LOW COST ★ 4 TTL INPUTS (5V) ★

★ 4 RELAY CLOSURES (100V 500mA) ★

★ INTERACTS WITH BASIC FOR SIMPLICITY OF USE ★

CIL
MICROSYSTEMS LTD

*"PET" is the registered trademark of Commodore Business Machines.

Interfacing

Additional I/O Interface

Mr. K. Erler of Edson Alberta writes in with:
...a schematic of an interfacing idea of mine. It simply interfaces a second VIA chip to the PET, thus tripling the user's I/O capability. Most of it is direct interfacing .. all but the address lines which had to be decoded.

The circuit uses only 4 three input "AND" gates and one buffer inverter. Once assembled, it connects directly on to the Memory Expansion Port -J4.

After connecting it, operation is very simple. The circuit is designed to use the top 16 bytes of RAM expansion space and since most PETs have only 8K (32K at the most) the very top of the memory would not be used.

The addresses are as follows:

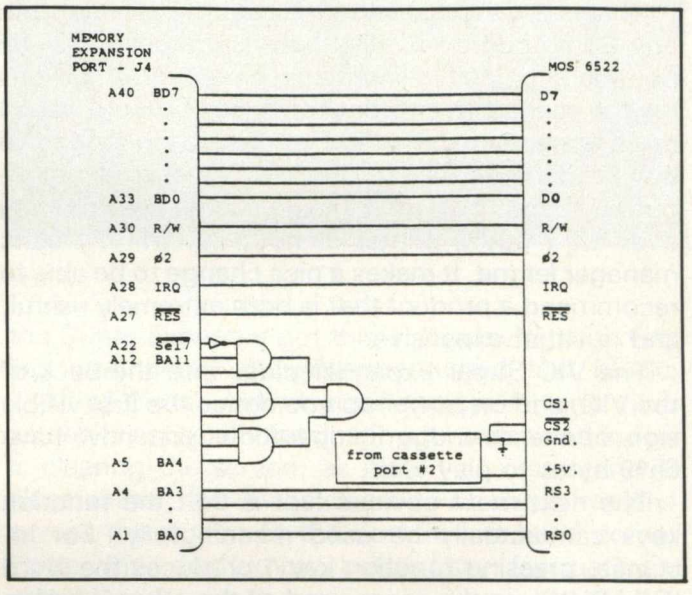
32752 - ORB	32760 - T2L-L T2C-L
32753 - ORA	32761 - T2C-H
32754 - DDRB	32762 - SR
32755 - DDRA	32763 - ACR
32756 - T1C-L T1C-L	32764 - PCR
32757 - TIC-H	32765 - IFR
32758 - T1L-L	32766 - IER
32759 - T1L-H	32767 - ORA (no hand shake)

The advantages are that you get not only PA lines, but also the PB lines and CB1 & CA2 lines.

The operation is as with the other VIA - PEEK and POKE, only with the previously listed addresses.

Output Example

To create a tone on CB2...
POKE 32762,15 (SR)
POKE 32760,155 (Timer 2)
POKE 32763,16 (ACR)
The schematic follows...



Graphics

An interesting double review this month, in that we look at the VIC Super Expander from Commodore, and the 80 column high resolution board from Supersoft. Interesting because both give a high resolution capability to their respective machines, but go about it in a totally different way. First, the VIC Super Expander.

VIC Super Expander

This is a cartridge based product, available from any Commodore VIC distributor, at a cost of 34.95 pounds (inc. VAT). There are so many VIC add-ons on the market at present, and (so it would seem) more appearing daily, that to decide which one to buy is a difficult task for the newcomer to the computing scene. But, after the inevitable heartrending over 'do I buy a printer or not', i.e. will my bank manager let me, it makes a nice change to be able to recommend a product that is both extremely useful, and not that expensive.

The VIC Super Expander plugs into the back of the VIC, and on power up you notice the first visible sign of the cartridge in operation: you now have 6519 bytes to play with.

The next most obvious fact is that the function keys can actually be used meaningfully. For instance, pressing function key 1 produces the word 'GRAPHIC' on the sceen, and all the other function keys have similar commands associated with them. No longer can they only be used for detecting when one of them is pressed! (By using CHR$ (133) to CHR$(140) on a normal VIC, in case you didn't know).

More than this however is the ability to re-assign the commands. I've now got into the habit of assigning function key 1 to be the keyword RUN, plus a carriage return. Thus pressing that key runs your program for you at a single key stroke. The limit to the amount of code you can assign appears to be 4 lines of the VIC screen, or 88 characters (less the 5 needed to set the command in operation).

The main purpose of the Super Expander is in its use of graphics, and the ability to very easily produce quite complex displays.

The Graphics

There are 9 graphic commands in total, plus a further command for controlling sound, and 7 functions which enable you, amongst other things, to read the position of a game paddle, a joystick or a light pen; what a blessing it is to be able to do this easily.

The graphic commands are respectively GRAPHIC, which allows you to set one of four graphics modes; SCNCLR, which clears a graphic screen; COLOR (I wonder where this product came from ?!) for selecting screen, border, character and auxiliary colours; REGION, for selecting a character colour; DRAW, for drawing a line between two points; POINT, which plots a single point; CIRCLE, for drawing a circle, ellipse, or arc; PAINT, which fills an enclosed area around a specified point with a specified colour, and CHAR, which puts text onto the graphics screen.

All of these have to be used in program mode rather than immediate, apart from SCNCLR, which works anywhere.

The most frequently used commands will probably be DRAW, CIRCLE and PAINT, which, when using the high resolution plotting mode of 1024 by 1024 enable you to very simply produce the most intricate of patterns and designs, and are very useful in illustrating a point in an educational program for example. Let me here clear up one fallacy which appears to have spread around; the screen is NOT mapped on a 1024 by 1024 resolution. Points are plotted on that scale and then scaled down to be plotted on the VICs screen.

In conclusion, this is a very handy package, and comes highly recommended.

SUPERSOFT High Resolution Board

Whereas the VIC Super Expander is simplicity itself to fit, this unfortunately isn't!

The High Resolution Board fits inside the PET, so from the ouside it is to all intents and purposes a normal PET. To get it to fit however, requires two pages of instrucitons, removing and installing of chips, fitting of probe hooks and connecting of plugs. I must stress that none of this is that difficult, just time consuming and fiddly; my patience was sorely tried at one point, and I should imagine that a number of other users would feel the same! Having said that, a lot of thought has obviously gone into the design and layout of the board, in order to make it as inexpensive to produce as possible, and certainly cheaper than similar products on the market.

Still, once it's there, it's there, and you're free to explore the wonders of high resolution graphics.

Supersoft have gone for a different command mode for using their graphics than Commodore; in view of the fact that they're using an 80 column PET and have to interface to BASIC, they've chosen the most sensible route. Instead of commands like COLOR, PAINT and so on, every graphic function is accessed by a SYS 59650 call. This was chosen to maintain compatability with any software that adds its own commands to BASIC (e.g. toolkits, basic aid etc.), and to avoid slowing down of BASIC execution, which is inherent when any wedge is placed in the CHRGET routine.

The screen is mapped on a 320 x 200 grid, and in-

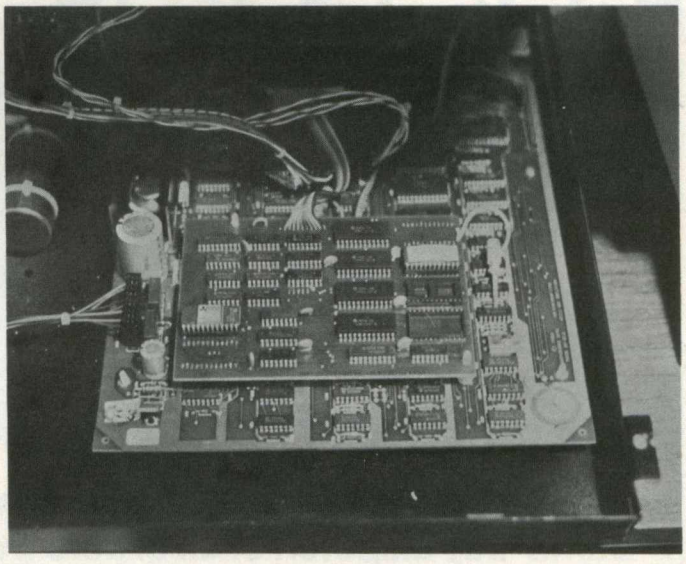
deed that is the resolution that the board gives you. The bottom left hand corner is 0,0 and the top right is 319,199. Thus all plotting is done with those co-ordinates in mind.

The board gives you essentially the same commands as the VIC Super Expander, but in the form of SYS rather than new keywords. No memory is taken from the PET; equally, none is added. However, since that is not what you're paying for, we can't quibble: with the VIC it's just a bonus.

Now onto a common gripe: documentation. This can look technically superb, as in the case of The Manager (reviewed elsewhere), but be lacking in actual content: it can look technically bad, as in the case of the preliminary documentation supplied with the VIC Super Expander, but be quite good in terms of content. There is a third category, which I'm afraid Supersoft's documentation for the board falls into. It doesn't look very good, and leaves a lot to be desired in terms of information presented. The bare bones are there, in that you're told what the commands are, but there is no real explanation of some of the less obvious points e.g. you're told that you can have text blocks in memory, but you'd have to be rather clever to figure out how! Even worse, there is a mistake in the section explaining how to connect the board up: I'm told that has now been corrected.

Despite all that, it's the hardware you're paying for and with that I have no complaints. Once it's fitted correctly, and the documentation is understood, you are left with a very good product indeed.

The basic graphic commands are Set, Reset, Flip and examine a point; Set, Reset and Flip a line; Set and Reset points on a continous straight line; Fill and Erase an area around a specified point; Clear the screen and Invert the screen (and why hasn't the Super Expander got this?); Displaying a text string, and finally the aforementioned text blocks.



All of these work at gratifying speed, and invert screen is especially fascinating to watch. As indeed is clearing the screen, as it isn't an instantaneous clear: rather, the image clears downwards (rapidly!) across the screen to the bottom left hand corner. Quite why I don't know, but as I say: fascinating to watch!

One major difference between this board and the VIC one is that all the commands can be entered in direct mode; this is useful in writing a program when you want to check on one or two lines. Rather than having to go through the whole lot again, or typing RUN 40 when it should have been 60, you just enter the code and see what happens. Another difference, and purists can argue the merit of this one, is that with the VIC, when you're in graphics mode you only see the graphics display, and going back to text mode clears it, whereas here you can have both displays on the screen at the same time. In other words, with a short program you can actually look at the listing whilst the program is being executed. Scrolling down moves one screen but not the other.

To conclude, despite the limitations of the documentation and the relative difficulty of assembly, for 149 pounds (ex. VAT) this is a well thought out, useful piece of hardware. The ability to have variables passed to the graphics commands means you can have very graphic illustration of programs. There are versions available for the 40 column machines as well: Supersoft, on 01 861 1166 have all the details.

Next month we'll be featuring listings of programs implementing high resolution graphics on the VIC.

SEARCH TECHNIQUES

This article looks at the different techniques which can be used to search for a particular item of data in a file or table. Linear and binary searches are considered, together with an introduction to the powerful method known as hash tabling.

One of the central problems encountered in writing computer programs is how to store and then retrieve data. In practice, it is often found that this latter operation, recovering previously-acquired data from storage, presents the greater problem, to such an extent that the algorithms used to store the data initially are determined by what will happen to it later.

Many examples of this data storage/retrieval problem are to be found in the current crop of microprocessor-based programs.

A typical problem would be faced by a business package writer. Important information is stored under the name of a person or company to whom an invoice, receipt or some goods must be sent, or from whom something is expected.

To access the record of this person or organisation within a file the name must be entered with a request for the desired details. The program will then search through all the records until the name input matches one in the file; then the information stored in the record under that name may be processed and printed.

Primary Interest

While it is true that circumstances will determine the best search strategy, and a number of well-known techniques exist, three of them will be looked at in this article. They all have certain requirements. They must be short in terms of computer time and program length. They must guarantee to find a record if it exists and must stop if no record is found to match the key. Failure to do this would put the program in an infinite loop if no such record exists, or the keys were mis-typed.

In each of the three cases it will be assumed that the key, the name of some data, is to be stored in a list, and associated with each key will be some further information, which is the thing of primary interest.

In the business example the list of keys would be stored in the main memory of the computer and each entry will have a pointer stored with it. This

```
 LOAD
0001 DATA ABA      0019 DATA BPL      0037 DATA INC      0055 DATA SBA
0002 DATA ADC      0020 DATA BRA      0038 DATA INS      0056 DATA SBC
0003 DATA ADD      0021 DATA BSR      0039 DATA INX      0057 DATA SEC
0004 DATA AND      0022 DATA BVC      0040 DATA JMP      0058 DATA SEI
0005 DATA ASL      0023 DATA BVS      0041 DATA JSR      0059 DATA SEV
0006 DATA ASR      0024 DATA CBA      0042 DATA LDA      0060 DATA STA
0007 DATA BCC      0025 DATA CLC      0043 DATA LDS      0061 DATA STS
0008 DATA BCS      0026 DATA CLI      0044 DATA LDX      0062 DATA STX
0009 DATA BEQ      0027 DATA CLR      0045 DATA LSR      0063 DATA SUB
0010 DATA BGE      0028 DATA CLV      0046 DATA NEG      0064 DATA SWI
0011 DATA BGT      0029 DATA CMP      0047 DATA NOP      0065 DATA TAB
0012 DATA BHI      0030 DATA COM      0048 DATA ORA      0066 DATA TAP
0013 DATA BIT      0031 DATA CPX      0049 DATA PSH      0067 DATA TBA
0014 DATA BLE      0032 DATA DAA      0050 DATA PUL      0068 DATA TPA
0015 DATA BLS      0033 DATA DEC      0051 DATA ROL      0069 DATA TST
0016 DATA BLT      0034 DATA DES      0052 DATA ROR      0070 DATA TSX
0017 DATA BMI      0035 DATA DEX      0053 DATA RTI      0071 DATA TXS
0018 DATA BNE      0036 DATA EOR      0054 DATA RTS      0072 DATA WAI
```

pointer will provide sufficient information to allow the program to locate the rest of the record immediately.

It might, for instance, be the disc/track/sector number on a floppy disc file store, where the bulk of the information is to be found. Further examples where searching is important may be found in the writing of compilers and assemblers.

The great advantage of an assembly language, like the one produced by Motorola for the MC6800 microprocessor, is that symbolic names can be used for the instructions, for labels (machine location) and in the operands (like variable names and named constants).

Key Words

This means that the user can have alphabetic names instead of numbers. Similarly, a compiler or interpreter such as Basic has a number of special keywords - often called reserved words - which are recognised by the compiler or interpreter.

In BASIC, READ, IF-THEN, GOTO, DATA, FOR-NEXT INPUT and PRINT are all reserved words.

A table of them will be stored in the interpreter program. Whenever a word is extracted from the program during interpretation, the table is searched. If the word is found and identified as a reserved word, some code will be called to interpret the program text which follows.

The reserved word INPUT will jump to the code which expects input from a keyboard. Since each time a reserved word must be searched, efficiency is obviously of great importance if the source program is to run quickly.

A first approach to designing all these systems is to store the words in the list in any arbitary order and then to search from the top, checking each one in turn. If the target word is found the search stops; if the bottom of the list is reached a "not-found" error message is generated.

The DATA list in Program 1 shows all the mnemonic opcodes for the M6800 assembler. They are equivalent to the reserved words of the Basic interpreter. Every machine code instruction will contain one of these opcodes, so for each assembly code program line - excluding comments and some assembler directives - this list must be searched by matching the target opcode against the members of the list.

In a real assembler further data would be stored, with each of the nmemonics providing such information as the instruction class and some pointer into a further table containing the various hex values of the opcodes in all the modes.

The list is in alphabetical order. It should be noted, however, that this ordering bears little rela-

```
KEY STRING ?
                PROGRAM 1


 500 PRINT "TYPE - 1 FOR LINEAR SEARCH DEMONSTRATION"
 510 PRINT "       2 FOR BINARY SEARCH DEMONSTRATION"
 520 PRINT "       3 FOR AUGMENTED LINEAR SEARCH DEMO"
 530 INPUTQ
 540 ON Q GOTO 1000, 2000, 3000
 1000 REM LINEAR SEARCH
 1010 GOSUB 5000
 1020 INPUT "TARGET";T$
 1030 FORI=1TOL
 1040 IFT$=L$(I)THEN1100
 1050 NEXTI
 1060 PRINTT$;" NOT FOUND AFTER ";I;" ITERATIONS"
 1070 INPUT "TRY AGAIN";T$
 1080 IFLEFT$(T$,1)="Y"THEN1020
 1090 END
 1100 PRINTT$;" FOUND AFTER ";I;" ITERATIONS"
 1110 GOTO1070
 2000 REM BINARY SEARCH
 2010 GOSUB 5000
 2020 INPUT"TARGET";T$
 2030 T=1
 2040 R=L+1
 2050 M=T+INT((R-T)/2)
 2055 PRINTM;
 2060 IFT$<>L$(M)THEN2130
 2070 PRINT "FOUND ";T$;" AT ";M
 2080 GOTO2160
 2090 IFT=MTHEN2150
 2100 IFT$>L$(M)THEN2130
 2110 R=M
 2120 GOTO2050
 2130 T=M
 2140 GOTO2050
 2150 PRINT"ERROR - ";T$;" NOT FOUND"
 2160 INPUT"TRY AGAIN";T$
 2170 IFLEFT$(T$,1)="Y"THEN2020
 2180 END
 3000 REM AUGMENTED LINEAR SEARCH
 3010 GOSUB 5000
 3020 INPUT"TARGET";T$
 3030 X$=LEFT$(T$,1)
 3040 IFX$<"S"THEN3070
 3050 G0=55
 3060 GOTO3140
 3070 IFX$<"J"THEN3100
 3080 G0=40
 3090 GOTO3140
 3100 IFX$<"C"THEN3130
 3110 G0=24
 3120 GOTO3140
 3130 G0=1
 3140 FORI=G0TOL
 3150 IFT$=L$(I)THEN3210
 3160 NEXTI
 3170 PRINT"ERROR - ";T$;" NOT FOUND AFTER ";I-G0+1;" ITERATIONS"
 3180 INPUT"TRY AGAIN ";T$
 3190 IFLEFT$(T$,1)="Y"THEN3020
 3200 END
 3210 PRINTT$;" FOUND AFTER ";I-G0+1;" ITERATIONS"
 3220 GOTO3180
 5000 REM READ IN TABLE
 5005 RESTORE
 5010 L=72
 5020 DIML$(L)
 5030 FORI=1TOL
 5040 READL$(I)
 5050 NEXTI
 5060 RETURN
READY.
```

tion to the functional ordering, the one which affects the miroprocessor chip directly. INX is no more related to JMP than it is to ADD, even though in the first case they are lexically adjacent and in the second they are wildly-separated.

We take pot-luck when searching the list about how many matches must be tried between target and stored members of the list before one succeeds.

The Basic program from 1,000 to 1,110 demonstrates how such a search is coded. Subroutine 5,000 reads in the table. The target mnemonic is read-in from the keyboard and a FOR loop is entered to compare each member of the list to the target in turn.

When the target matches, the program jumps out of the loop, the current value of the FOR loop counter I is the position in the list which held the target value. This printed value is the same as the BASIC DATA statement number.

Test Run 1 shows this program in operation. If we search for ABA, the first in the list, it is located very rapidly the first time round the loop. If we search for WAI, the last loop, it takes 72 iterations around the loop before the match is made. Intermediate targets, such as BVS, CBA and NOP are found after 23,24 and 47 iterations respectively.

The time taken to locate any item is linearly almost proportional to the number of iterations required to find it. Timing the performance of this program when coded in SWTP 8K Basic shows that it takes about three seconds to find WA1.

The list of opcodes in Table 1 shows how the program can be speeded by taking advantage of the MC6800 structure. It transpires that about 10 percent of the instructions are used 90 percent of the time. Table 1 was produced by counting the number of times each opcode appeared in a large program. JSR and LDX are clear winners, followed closely by CMP, LDA, BEQ, BSR, STA, STX, BNE, JMP, BRA and TRS.

This shows that subroutine calls (JSR, BSR and RTS) are the most common operations followed by pointer manipulation (LDX and STX), then tests and branches (CMP, BEQ, BNE, JMP and BRA), with accumulator stores and loads being about equal (LDA and STA).

Then there are many instructions which are used occasionally, including the arithmetic ones. About one-third of the instructions were not used at all in this program.

This new ordering is probably not too far from the "average best" ordering, although counting with a wide range of programs might make the average search time slightly shorter. Moving the first few to the top, however, and leaving all the others alone would produce great savings in overall search time.

No Special Ordering

When there is no special ordering, the average search time for an item is approximately 1/2n, where there are n items in the list. Little can be said about special ordering cases as it depends on the effects the ordering will cause in the individual instance.

Special-case ordering is not limited to assembler design. A business system could sort its customer list according to how often an order is placed. A periodic shuffle would be required to update the lists. The disadvantage of this special-case shuffling scheme is that it leaves the list in a very unhelpful order for any other task, such as printing a list of customers in alphabetical order. Nevertheless, it is a possibility not to be overlooked, particularly if a few entries make up the bulk of the search task.

A second technique is demonstrated by statements 2000 to 2180 in program 1 which exploits lexical ordering and reduces search time in the binary search, to find the target item a point is chosen halfway between the two ends of the list. A check is first made to see if the target matches the middle item in the list (st.2060); if it does, the target succeeds and the program exits, with the middle point value being printed as the result (st.2079).

Table 1

If the match fails, the list being searched will be divided into two, and one of the halves will be sear-

```
TYPE - 1 FOR LINEAR SEARCH DEMONSTRATION
       2 FOR BINARY SEARCH DEMONSTRATION
       3 FOR AUGMENTED LINEAR SEARCH DEMO

TYPE - 1 FOR LINEAR SEARCH DEMONSTRATION
       2 FOR BINARY SEARCH DEMONSTRATION
       3 FOR AUGMENTED LINEAR SEARCH DEMO
?  1
TARGET ? ABA
ABA FOUND AFTER  1  ITERATIONS
TRY AGAIN ? YES
TARGET ? WAI
WAI FOUND AFTER  72  ITERATIONS
TRY AGAIN ? YES
TARGET ? BVS
BVS FOUND AFTER  23  ITERATIONS
TRY AGAIN ? YES
TARGET ? CBA
CBA FOUND AFTER  24  ITERATIONS
TRY AGAIN ? YES
TARGET ? NOP
NOP FOUND AFTER  47  ITERATIONS
TRY AGAIN ? YES
TARGET ? AAA
AAA NOT FOUND AFTER  73  ITERATIONS
TRY AGAIN ? YES
TARGET ? ZZZ
ZZZ NOT FOUND AFTER  73  ITERATIONS
TRY AGAIN ? NO
```

ched. If the target is less than the mid-point, the same process is repeated with the bottom of the list taken as the mid-point used on the previous occasion (st.21110).

If the target is lexically greater than the item at the mid-point, the top of the new sub-list is subjected to the same procedure until one of the mid-points generated equals the target.

If the top or bottom point becomes equal to the middle point without the match (st.2090), the whole list has been searched and no entry equal to the target exists, and an error message is printed (st.2150).

This algorithm obviously fails if the list is not in lexical order but the search time is guaranteed to be short. The overall search time is also kept down. Average search length will be close to log2N, where N, is the length of the table. In this example log2N, is 6.16 for 72 entries.

Tested with Targets

Compare this to the linear search average of 36. The situation is not so promising, as two comprisons between the target string and those in the list are made per trial (st.2060 and st.2100), whereas there was only one per iteration in the previous example, and there is more calculation to be performed, mainly caused by statement 2050, which calculates the new mid-point (variable M) by bisecting the top (T) and the bottom (B) of the list.

Trial Run 2 shows this algorithm in action. It takes about seven bisections to locate an entry - for instance RTS, ABA and WAI. On only one case, INC, the first middle point matches the target. In two cases, of which BPL is one, only two bisections ar required.

CLV is one of the four cases where three attempts are needed, eight with four tries (BVS) and so on. It is important to test extremes to ensure that the algorithm locates them correctly. Both ABA, the first, and WAI, the last, present no problem and are located correctly in seven attempts.

It is also tested with targets which do not appear in the search list. PQR fails after seven attempts, as does AAA, which is before any of the entries. The worst case with the current program design is ZZZ which fails after eight attempts. Note that 123 is lexically before AAA because ASCII "1" is before ASCII "A" Similarly " AA" comes after ZZZ because the opening square bracket is after "Z".

If data containing these characters is used it should be remembered that the digits come before colon, semi-colon, commercial at "@" etc, and that they come before the capital letters. These in turn are before the square brackets and the lower-case letters. Space is the first ASCII "printing" character, so any string beginning with a space comes before all the rest.

TABLE 1

1	USR	43	37	STS	2
2	LDX	43	38	AEA	1
3	CMP	36	39	ADC	1
4	LDA	30	40	BLE	1
5	BEQ	28	41	BLS	1
6	BSR	26	42	CLC	1
7	STA	23	43	RTI	1
8	STX	22	44	SBC	1
9	BNE	18	45	BGE	0
10	JMP	18	46	BHI	0
11	BRA	15	47	BIT	0
12	RTS	13	48	BLT	0
13	CLR	11	49	BPL	0
14	DEC	7	50	BVC	0
15	PSH	7	51	BVS	0
16	PUL	7	52	CBA	0
17	SUB	7	53	CLI	0
18	ADD	6	54	CLV	0
19	INX	6	55	DAA	0
20	TST	6	56	DES	0
21	AND	5	57	EOR	0
22	DEX	5	58	INS	0
23	ASL	4	59	NEG	0
24	ASR	4	60	CRA	0
25	BCC	4	61	ROL	0
26	BCS	4	62	ROR	0
27	INC	4	63	SBA	0
28	LSR	4	64	SEC	0
29	TAB	4	65	SEI	0
30	COM	3	66	SEV	0
31	TSX	3	67	SWI	0
32	BGT	2	68	TAP	0
33	BMI	2	69	TBA	0
34	CPX	2	70	TPA	0
35	LDS	2	71	TXS	0
36	NOP	2	72	WAI	0

The numbers printed-out in Run 2 show the mid-point of the sub-lists at each state (st.2055). The first attempt is always 37, the next either 19 or 55, and the next 10, 28, 46, or 64 and so on. With the SWTP 8K Basic, the worst case search took about 1.5 seconds, including printing-out the intermediate mid-points on a high speed video terminal.

With tables this length the gains over a linear table are marginal. When the table has 1,000 entries, the linear search average has risen to 500, but the binary search is somewhat less than 9.97.

Perhaps the least desirable aspect of any ordered list of this type is the difficulty of adding new items. Any new item must be added in the correct place or the search algorithm will fail, this means that all the items below the one being added must be shifted down one place to make room for the addition. So binary search is best for applications where the complete list is known from the start, or where few additions have to be made.

When additions must be made, they can all be done in a batch. While on the subject of alphabetical ordering, statements 3000-3200 show how the linear search can be speeded by starting it part of the way down the list.

If the first letter of the target is "S" there is no point in checking those entries beginning "A" to "R", so we start at entry 55 - SBA. Start at 40 if the target begins with "J", 24 if "C", one otherwise.

Run Three shows ABA still takes one iteration and

BVS still takes 23, the next one in the list, CBA, now takes only one try, WAI, our previous worst case, now takes only 18 iterations. Non-existent entries now take a variable time, the worst being AAA and the best ZZZ, because the search is mostly finished before it starts.

Clearly what is required is a technique for storing and retrieving information which is easy to extend by adding more entries; that allows quick access to the previously-stored information, and also uses minimum program and data storage space. There is a number of contenders for the title. One is the hash table technique and the second is linked lists, which will be left for another time. The hash table allows independent insertion and recovery.

In either case, the key string is subjected to a "hash-function", which should randomise the input completely. Program Two shows such a function. It takes a string as input and returns a number, which should be totally unrelated to the input - except, of course, that if the function is applied a second time it returns the SAME number, and continues to return the same number every time the same key-string is entered. The listing after Program Two shows the effect on a number of data strings.

```
TYPE - 1 FOR LINEAR SEARCH DEMONSTRATION
       2 FOR BINARY SEARCH DEMONSTRATION
       3 FOR AUGMENTED LINEAR SEARCH DEMO
? 2
TARGET ? RTS
 37  55  46  50  52  53  54 FOUND RTS AT  54
TRY AGAIN ? YES
TARGET ? ABA
 37  19  10  5  3  2  1 FOUND ABA AT  1
TRY AGAIN ? YES
TARGET ? INC
 37 FOUND INC AT  37
TRY AGAIN ? YES
TARGET ? BPL
 37  19 FOUND BPL AT  19
TRY AGAIN ? YES
TARGET ? CLV
 37  19  28 FOUND CLV AT  28
TRY AGAIN ? YES
TARGET ? BVS
 37  19  28  23 FOUND BVS AT  23
TRY AGAIN ? YES
TARGET ? ROR
 37  55  46  50  52 FOUND ROR AT  52
TRY AGAIN ? YES
TARGET ? PSH
 37  55  46  50  48  49 FOUND PSH AT  49
TRY AGAIN ? YES
TARGET ? NOP
 37  55  46  50  48  47 FOUND NOP AT  47
TRY AGAIN ? YES
TARGET ? WAI
 37  55  64  68  70  71  72 FOUND WAI AT  72
TRY AGAIN ? YES
TARGET ? PQR
 37  55  46  50  48  49  48 ERROR - PQR NOT FOUND
TRY AGAIN ? YES
TARGET ? AAA
 37  19  10  5  3  2  1 ERROR - AAA NOT FOUND
TRY AGAIN ? YES
TARGET ? ZZZ
 37  55  64  68  70  71  72  72 ERROR - ZZZ NOT FOUND
TRY AGAIN ? YES
TARGET ? 123
 37  19  10  5  3  2  1 ERROR - 123 NOT FOUND
TRY AGAIN ? YES
TARGET ? [AA
 37  55  64  68  70  71  72  72 ERROR - [AA NOT FOUND
TRY AGAIN ? NO
```

HASH Function

Program Two works by translating each of the first five characters in the string into their decimal equivalent (A = 65. Z =), retaining only the last digit (A = 5 Z = 0) and forming a decimal number from the (NEXT = 78,69,88,84,32 = 89842). The two is added to the end because the strings are always padded up to five characters with spaces (32), and only the first five characters are used.

This number is then divided by a prime number and the remainder is extracted (89842/53 = 1695:13207). The remainder is then manipulated by the table length to give the position of the table entry, in this case 6 (0.13207*50).

To add an item to the table, the hash function is applied to the key-string and an attempt is made to store both the input string and the data to be associated with it in the Nth entry point, where N is the value returned from the hash function. The key string will be stored in the Nth entry of the key-table, and the data in the corresponding Nth entry of the data table.

If the Nth location in the two tables is empty, then it is stored. If something is already stored in that entry - the hash function cannot return a value for each key-string - then an attempt is made to store it in the next location N + 1.

The listing from Program Two shows that both HASH (VALUE) and HASH (OCODE) return the value 42. If location N + 1 is empty the data is stored there. If the end of the table is reached before a free space is found, it wraps round to the beginning of the table and continues from entry 1.

If M attempts are made to store the data, where M is the length of the table, the table is full and the size of the table must be increased. This is done most easily with a 're-hash'. Entries from the full table are extracted, fed through a different hash function and placed in a bigger table. Hence the new hash function is always used.

To locate an item already stored in the table, the input key is headed, the search starts at the location pointed to by the returned value HASH(VALUE) and HASH(OCODE) both still equal 42. The search for either VALUE or OCODE would start at 42. Since VALUE was stored first, it would be located immediately.

When searching for OCODE, the first match at 42 would fail, because it contains VALUE, the next location; 43 is matched which succeeds, because

OCODE has been stored there. If we search for a third item which hashes to 42, and the match with both VALUE and OCODE fails and location 44 contains null string (no entry), we know that the third item was never stored.

Hash Tables

Hash tables are ideal for search/store applications in which data is often being saved and located. The two operations may be mixed and there is no need to build the table first. The technique is often used in compiler and assembler symbol tables.

Each time a label or variable is mentioned the table is searched, if the item is already there, its value is extracted from the data field and returned: if it has not been defined previously, it is added into the next free location.

The run after Program Two shows that hash function is equally happy with peoples' names (perhaps for an address list), variable names (from a high level language program) such as SYMBOLNAME and MODE, reserved words from the Basic language, and even assorted ASCII strings.

Since only the first five characters are significant, HEXOUT and HEXOUT1 both return two. AAAA, AAAB and AAAC, however, produce very different table entries. One disadvantage of the hash table is that a slight typing error in the key-string leads to a gross error in the table entry point. A second disadvantage is that the size of the table is fixed, and must be somewhat larger than the total number of entries expected.

Hash functions can also be somewhat tempermental and should be chosen with care, the one used in Program Two was by no means my first attempt. Clumping, where many entries all map on to a single location, leaving the majority clear, can occur, and will increase search time dramatically.

This happens when I used both digits of the decimal equivalent of the string divided by a small prime number: SYMBOL = 8389776679 divided by 727 produced 11540270.5, with only a singe digit remainder in each case, because this was at the limit of the arithmetic accuracy in Basic.

```
1000 REM HASH FUNCTIONS FOR STRINGS
1010 REM HASH TABLE SIZE
1011 S=50
1020 REM A PRIME NUMBER
1021 P=53
1030 INPUT"KEY STRING";Q$
1040 Q$=Q$+"    "
1050 Q$=LEFT$(Q$,6)
1060 R$=""
1070 FORI=1TO5
1075 T=ASC(MID$(Q$,I,1))/10
1080 R$=R$+STR$(INT((T-INT(T))*10))
1085 NEXTI
1090 R=VAL(R$)
1100 H=R/P
1110 O=INT((H-INT(H))*S)
1120 PRINT"HASH(";Q$;")= ";O;" [";R$;",";H;"]"
1130 GOTO1030
READY.
```

When a larger prime was chosen to combat the clumping around the multiples of five (894581), similar strings began to appear at the same location. This would be undesirable for an address list.

It seems that choosing a prime just larger than the size of the table gives the best performance (prime P = 52 for a table of size S = 50). It might help when using this function to reverse the order of the characters as they are converted into digits if any trouble is encountered.

To give some idea of how much better over linear and binary search the hash table can be, here are a few performance figures. When the table is only 10 percent full, the average search length is 1.053; at 90 percent full it is about 5.5. A completely full table is a pathological case and the search length increases to about 15-16.

It would be noted, however, that these search lengths are essentially independent of the table size. So for problems involving continuous storage and retrieval of data, in a hurry, the hash table algorithm may be the best solution.

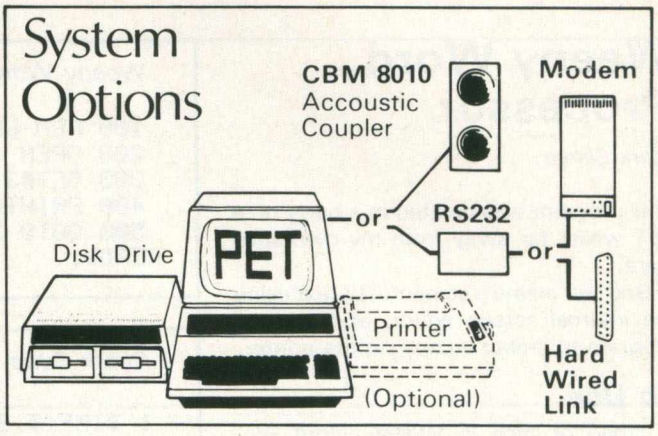
```
KEY STRING ? AAAA
HASH(AAAA  )=  7  [ 5 5 5 5 2, 1048.15094 ]
KEY STRING ? AAAB
HASH(AAAB  )=  7  [ 5 5 5 5 2, 1048.15094 ]
KEY STRING ? AAAC
HASH(AAAC  )=  16  [ 5 5 5 6 2, 1048.33962 ]
KEY STRING ? ANDERSON
HASH(ANDERS)=  7  [ 5 8 8 9 1, 1111.15094 ]
KEY STRING ? BACKUS
HASH(BACKUS)=  4  [ 5 5 6 5 5, 1050.89434 ]
KEY STRING ? BROOKER
HASH(BROOKE)=  1  [ 5 1 9 9 5, 981.037736 ]
KEY STRING ? CHOMSKY
HASH(CHOMSK)=  5  [ 6 1 9 6 3, 1169.11321 ]
KEY STRING ? FLOYD
HASH(FLOYD )=  49  [ 0 5 9 8 8, 112.981132 ]
KEY STRING ? HOROWITZ
HASH(HOROWI)=  28  [ 1 9 1 6 3, 361.566038 ]
KEY STRING ? KNUTH
HASH(KNUTH )=  17  [ 5 8 5 3 1, 1104.35849 ]
KEY STRING ? NAUR
HASH(NAUR  )=  21  [ 8 5 5 1 2, 1613.43396 ]
KEY STRING ? ZAND
HASH(ZAND  )=  49  [ 0 5 8 8 2, 110.981132 ]
KEY STRING ? SYMBOLNAME
HASH(SYMBOL)=  20  [ 3 8 6 5 9, 729.415094 ]
KEY STRING ? MODE
HASH(MODE  )=  25  [ 6 9 8 9 2, 1318.71698 ]
KEY STRING ? VALUE
HASH(VALUE )=  48  [ 6 5 5 5 9, 1236.96226 ]
KEY STRING ? OPCLASS
HASH(OPCLAS)=  23  [ 9 0 6 5 5, 1710.4717 ]
KEY STRING ? OCODE
HASH(OCODE )=  49  [ 9 6 9 8 9, 1829.98113 ]
KEY STRING ? HEXOUT
HASH(HEXOUT)=  18  [ 1 9 8 9 5, 375.377358 ]
KEY STRING ? HEXOUT1
HASH(HEXOUT)=  18  [ 1 9 8 9 5, 375.377358 ]

KEY STRING ? PRINT
HASH(PRINT )=  4  [ 0 1 3 8 3, 26.8943396 ]
KEY STRING ? RESTORE
HASH(RESTOR)=  44  [ 1 9 3 3 9, 364.886793 ]
KEY STRING ? NEXT
HASH(NEXT  )=  47  [ 8 9 8 3 2, 1694.9434 ]
KEY STRING ? 1234
HASH(1234  )=  16  [ 9 0 0 1 2, 1698.33962 ]
KEY STRING ? 1024
HASH(1024  )=  14  [ 9 8 0 1 2, 1849.28302 ]
KEY STRING ? 1025
HASH(1025  )=  33  [ 9 8 0 3 2, 1849.66038 ]
KEY STRING ? @%%&
HASH(@%%&  )=  15  [ 5 5 7 7 2, 1852.30189 ]
KEY STRING ? AAAA
HASH(AAAA  )=  7  [ 5 5 5 5 2, 1048.15094 ]

TYPE - 1 FOR LINEAR SEARCH DEMONSTRATION
       2 FOR BINARY SEARCH DEMONSTRATION
       3 FOR AUGMENTED LINEAR SEARCH DEMO
? 3
TARGET ? ABA
ABA FOUND AFTER  1  ITERATIONS
TRY AGAIN ? YES
TARGET ? WAI
WAI FOUND AFTER  18  ITERATIONS
TRY AGAIN ? YES
TARGET ? BVS
BVS FOUND AFTER  23  ITERATIONS
TRY AGAIN ? YES
TARGET ? CBA
CBA FOUND AFTER  1  ITERATIONS
TRY AGAIN ? YES
TARGET ? NOP
NOP FOUND AFTER  8  ITERATIONS
TRY AGAIN ? YES
TARGET ? AAA
ERROR - AAA NOT FOUND AFTER  73  ITERATIONS
TRY AGAIN ? YES
TARGET ? ZZZ
ERROR - ZZZ NOT FOUND AFTER  19  ITERATIONS
TRY AGAIN ? NO
```

COMPASS NEW
The PET Communication Package £125·00

+ VAT

COMPASS, a COMPrehensive Asynchronous Software System for Commodore PET microcomputers, allows communication with any other type of mainframe, mini and micro computer system. The system has been developed to meet the needs of the rapidly developing communications market. No specialist knowledge is required to configure or use COMPASS.

The system communicates through an RS232 interface (except CBM 8010) using telephone modems, accoustic couplers or direct wired links. Users are able to undertake interactive processing and access remote data bases, with the added advantage of local processing power to run a wide range of business software.



COMPASS has widespread application. The system transmits information over the link in true ASCII form, so that PET's can be used as remote terminals for other systems which are not otherwise compatible. A selection option allows interchange of file data between host and PET or PET to PET.

COMPASS is available on disk for 2031/3040/4040/8050 drives, to run on Commodore 8032, 8096 and SuperPET models.

The "Ultimate" PET/VIC Software Development Tool (including EPROM Programmer)

A comprehensive and powerful system comprising of software and an EPROM programmer that fits directly to the user and cassette ports on the PET. All functions are handled by a single screen display, and include:

FETCH Any part of PET's RAM or ROM can be displayed on the screen in Hex, Disassembled Mnemonics and ASCII form

READ Contents of EPROM fitted in programmer socket can be displayed as above

GET Allows contents of disk file to be displayed as above

PUT Writes displayed data to disk in loadable program format

LIST Allows displayed data to be listed on printer

CHANGE Displays can be scrolled and modified directly

WRITE Writes contents of display to EPROM in programmer socket

Other facilities allow the movement of data into different work areas to provide, for instance, the facility to write the data from two 2K EPROMS into a 4K, verification of written EPROMS and changing type of display.

DEVELOP A PROGRAM ON YOUR PET — WRITE AN EPROM AND RUN ON THE VIC.

Supports all single rail EPROMS — 2716 (2K), 2532 (4K) and equivalents.

PRICE: £125.00 inc VAT — Inclusive of Software, EPROM Programmer and Manual.

Used by COMMODORE USA and UK.
Distributed in 4 continents.

Davidson-Richards Ltd.

14 Duffield Road, Derby DE1 3BB Telephone: Derby (0332) 366803

Basic Programs

Weeny Word Processor

Mark Clarke

This program was created in a hurry on a PET whilst far away from my own software.

One can create a screenful of text using the internal screen editor, and run the program to create a copy on the printer.

To Use

1/ Load or type in Weeny Word Processor, but do not run it.

2) Create a page of text using the normal screen editor, but use SHIFT/RETURN instead of RETURN at the end of a line, and avoid using the top line.

3) Mark the end of the text with a back arrow, and any following characters will not be printed.

4) When the text is in the form required, move the cursor to the top empty line, type RUN and hit RETURN. The text between RUN and the back arrow will then be printed. When printing is finished the cursor reappears, allowing another page of text to be created, starting from step 2.

Weeny Word Processor

```
 100 REM WEENY WP  MTC 82
 200 OPEN 3,3 :OPEN 4,4 :PRINT#4,"■";
 300 GET#3,A$ :IF A$="←" THEN PRINT#4 :CLOSE 3,4  :END
 400 PRINT#4,A$; :IF A$=CHR$(13) THEN PRINT#4,"■";
 500 GOTO 300
READY.
```

Rubik Cube

John Billingsley

Have you been sitting in front of The Cube for hours on end wondering what to do next? Or are you one of those geniuses who can solve it in one minute flat? (Ed. note: I tried for six months and eventually threw the thing away!)

Well, just for you, we have made it possible so that when you go wrong you can start with a new cube.

Just type in the program, make a copy, and then RUN. Then type in how ever many random twists you want to give the cube. Then, type the number of the face you want turned, and the program will duly respond. To turn it backwards, type '-' before the turn.

If you type in -1 when it asks for random turns it will allow you to set up strings of turns to do again and again and ....

Good luck: there are only Five times Ten to the power twenty nine turns!

PRIZE!!!

We will be offering a ten pound prize for the best implementation of this, or any other games program on a VIC 20. Colour, sound, what are you waiting for?!

Rubik Cube

```
 1 DIMF(5,8)
 2 DIMG(5,12)
 3 INPUT"■NUMBER OF TWISTS TO MIX";P
 4 M=1
 5 PRINT"■"
 20 PRINT"          ┌─────┐"
 25 PRINT"          │5    │"
 30 FORI=1TO6
 40 PRINT"          │     │":NEXT
 50 PRINT"  ┌───────┼─────┼─────┼─────┐"
 55 PRINT"  │4      │1    │2    │3    │"
 60 FORI=1TO6
 70 PRINT"  │       │     │     │     │":NEXT
 80 PRINT"  └───────┼─────┼─────┼─────┘"
 85 PRINT"          │6    │"
 90 FORI=1TO6:PRINT"          │     │":NEXT
 95 PRINT"          └─────┘■"
 150 F(0,0)=32768+13+(12*40)
 160 F(1,0)=32768+21+(12*40)
 170 F(2,0)=32768+29+(12*40)
 180 F(3,0)=32768+5+(12*40)
 190 F(4,0)=32768+13+(4*40)
 200 F(5,0)=32768+13+(20*40)
 210 C(0)=86:C(1)=90:C(2)=102:C(3)=127:C(4)=42:C(5)=160
 220 FORI=0TO5
 230 K=F(I,0)
 240 F(I,1)=K-82
 250 F(I,2)=K-80
 260 F(I,3)=K-78
 270 F(I,4)=K+2
 280 F(I,5)=K+82
 290 F(I,6)=K+80
 300 F(I,7)=K+78
 310 F(I,8)=K-2
 320 NEXT
 330 FORI=0TO5
 340 FORJ=0TO9STEP3
 350 READN
 360 FORK=1TO3
 370 READN2
 380 G(I,J+K)=F(N,N2)
 390 NEXT
 400 NEXT
 410 NEXT
 420 FORJ=0TO5
 430 FORI=0TO8
 440 POKEF(J,I),C(J)
 450 NEXT
 460 NEXT
 465 IFP<=0THEN1000
 470 GOTO710
 480 REM
 490 S1=PEEK(F(I,1))
 500 S2=PEEK(F(I,2))
 510 FORJ=1TO6
```

Rubic Cube continued

```
 520 POKEF(I,J),PEEK(F(I,J+2)):NEXT
 530 POKEF(I,7),S1
 540 POKEF(I,8),S2
 550 S1=PEEK(G(I,1))
 560 S2=PEEK(G(I,2))
 570 S3=PEEK(G(I,3))
 580 FORJ=1TO9
 590 POKEG(I,J),PEEK(G(I,J+3))
 600 NEXT
 610 POKEG(I,10),S1
 620 POKEG(I,11),S2
 630 POKEG(I,12),S3
 640 RETURN
 650 DATA4,7,6,5,1,1,8,7,5,3,2,1,3,5,4,3
 660 DATA4,5,4,3,2,1,8,7,5,5,4,3,0,5,4,3
 670 DATA4,3,2,1,3,1,8,7,5,7,6,5,1,5,4,3
 680 DATA4,1,8,7,0,1,8,7,5,1,8,7,2,5,4,3
 690 DATA2,3,2,1,1,3,2,1,0,3,2,1,3,3,2,1
 700 DATA0,7,6,5,1,7,6,5,2,7,6,5,3,7,6,5
 710 FORX=1TOP:I=INT(RND(1)*6):GOSUB480:NEXT
 720 PRINT"■■MOVE:1-6":PRINT"      OR-"
 725 IFM>2THENPOKE32768,45
 730 IFM<2THENPOKE32768,32
 740 GETA$
 750 IFA$=""THEN740
 760 IFA$="-"THENM=M+2:M=MAND3:GOTO720
 770 IFA$=" "THENRUN
 780 A=VAL(A$):IFA=0THEN740
 790 IFA>6THEN740
 800 FORN=1TOM:I=A-1:GOSUB480:NEXT
 810 M=1:GOTO720
 1000 IFP=0THEN720
 1010 INPUT"■GROUP CYCLE- STRING";G$
 1020 INPUT"■NUMBER OF CYCLES              ■■■■■■■■■";NC
 1030 IFNC<1THEN1010
 1040 FORCY=1TONC:FORGG=1TOLEN(G$):I=VAL(MID$(G$,GG,1))-1:GOSUB480:NEXT:NEXT
 1050 GOTO1020
READY.
```

Moving Maze *Jim Butterfield*

```
  100 REM MOVING MAZE    JIM BUTTERFIELD
  110 PRINT"[CLR,2CD,2CR]MOVING MAZE
  120 PRINT"[CD]   JIM BUTTERFIELD
  130 PRINT"[2CD]THE MAZE CHANGES AS YOU MOVE THROUGH IT
  140 PRINT"[2CD] .. YOU MAY PUSH THROUGH A WALL BUT IT
  150 PRINT"WILL COST YOU 10 MOVEMENT POINTS.
  160 PRINT"[CD] MOVE WITH KEYS:
  170 PRINT"           (UP)
  171 PRINT"             8
  172 PRINT"    (LEFT) 4   6 (RIGHT)
  173 PRINT"             2
  174 PRINT"          (DOWN)
  180 PRINT"[CD] [RVS]PRESS ANY KEY WHEN READY
  190 GETX$:IFX$=""GOTO190
  200 PRINTCHR$(142)"[CLR]":PRINT" ":FORJ=0TO90:IFPEEK(32768+J)=32THENNEXTJ:POKE5
 9468,12
  210 L=J:L1=(L-10)/2:L$=" ┌":R$="| ":M$="─":FORJ=1TO9
  220 PRINTL$;:FORK=1TOL-8:PRINTM$;
  230 NEXTK:PRINT"[CL]";R$
  240 IFJ=9GOTO280
  250 PRINT" | ";:FORK=1TOL-8:PRINT" ";:NEXTK:PRINT"[CL]";"| "
  260 L$=" | ":R$="| ":M$=" "
  270 NEXTJ
  280 PRINT"[CU] ─";:FORK=1TOL-8:PRINT"─";:NEXTK:PRINT"[CL]";" "
  290 FORV=0TO6:FORH=0TOL1-1:H%=V+H:IFINT(H%/2)=H%/2GOTO320
  300 GOSUB520:V%=RND(1)*2
  310 GOSUB530
```

continued over page

Moving Maze continued

```
320 NEXTH:NEXTV
330 M=0:V=0:H=-1:GOSUB520
340 C=X+1-L:POKEC,126
350 V=6:H=L1:GOSUB520:POKEX-L,96
360 GETX$:IFX$=""GOTO360
370 X=ASC(X$)-49:IFX<0ORX>8GOTO360
380 V%=X/3:H%=X-V%*3
390 V%=1-V%:H%=H%-1:IFV%*H%<>0GOTO360
400 M=M+1:PRINT"[HOME]";M;"[CL] MOVES"
410 IFV%+H%=0GOTO460
420 M1=C+L*V%+H%
430 M2=C+2*(L*V%+H%)
440 IFPEEK(M1)<>32GOTO470
450 POKEC,32:C=M2:POKEC,126
460 GOSUB720:GOTO360
470 IFPEEK(M1)<>96THENGOSUB770:GOTO360
480 POKEC,32:C=M2:POKEC,126
490 PRINT"[HOME]HOME IN";M;:INPUT"MOVES!  [RVS] ANOTHER GAME[OFF]";Z$
500 IFASC(Z$)<>78GOTO200
510 PRINT"[CLR]":END
520 X=(V+2)*L*2+(H+2)*2+32767:RETURN
530 IFV%>0GOTO670
540 GOTO590
550 FORJ=X-L*2TOX+LSTEPL:K=PEEK(J):Z=32:IFK=101GOTO580
560 Z=99:IFK=79GOTO580
570 STOP
580 POKEJ,Z:NEXTJ
590 FORJ=X-2TOX+1:K=PEEK(J):Z=99:IFK=32GOTO620
600 Z=79:IFK=101GOTO620
610 STOP
620 POKEJ,Z:NEXTJ:RETURN
630 FORJ=X-2TOX+1:K=PEEK(J):Z=32:IFK=99GOTO660
640 Z=101:IFK=79GOTO660
650 STOP
660 POKEJ,Z:NEXTJ:GOTO670
670 FORJ=X-L*2TOX+LSTEPL
680 K=PEEK(J):Z=101:IFK=32GOTO710
690 Z=79:IFK=99GOTO710
700 STOP
710 POKEJ,Z:NEXTJ:RETURN
720 V=INT(RND(1)*7):H=INT(RND(1)*L1):H%=V+H:IFINT(H%/2)=H%/2GOTO720
730 GOSUB520:K=PEEK(X)
740 IFK=101GOTO550
750 IFK=99GOTO630
760 STOP
770 M=M+10:Q=M1-32767-4*L-4
780 V2=INT(Q/L):H2=Q-V2*L
790 V=INT(V2/2):H=INT(H2/2)
800 IFV=V2/2GOTO840
810 IFH=H2/2GOTO870
820 STOP
830 V=V1:H=H1:GOTO730
840 H=(H2+1)/2:GOSUB910
850 IFETHENH=(H2-1)/2
860 GOTO890
870 V=(V2+1)/2:GOSUB910
880 IFETHENV=(V2-1)/2
890 IFV>=0ANDV<7ANDH>=0ANDH<L1GOTO730
900 RETURN
910 H%=V+H
920 E=(INT(H%/2)=H%/2):RETURN
READY.
```

Datamaker — *Courtesy of Jim Butterfield*

```
 0 QV=0:IFPEEK(54321)<>0THENQV=1:QK=525-QV*367:QB=526+QV*96:REM>DATAMAKER-NP
 1 S=826:F=1023:L=9:REM>S&F=ADRES
 2 PRINT"[CLR,2CD]":FORI=STOS+47STEP6:IFI>FTHENNEXT:L=3:PRINT"QK="QK":QB="QB":GO
TO6":GOTO5
 3 PRINTI;"DATA";:FORJ=0TO5:PRINTPEEK(I+J)"[CL],";:NEXTJ:PRINTCHR$(20):NEXTI
 4 PRINT"QK="QK"[CL]:QB="QB"[CL]:S="S+48"[CL]:F="F"[CL]:L="L"[CL]:GOTO2"
 5 POKEQK,L:FORK=1TOL:POKEQB+K,13:NEXTK:PRINT"[HOME]":END
 6 PRINT"[CLR,2CD]":FORM=0TO7:PRINTM:NEXTM
 7 POKEQK,8:FORK=1TO8:POKEQB+K,13:NEXTK:PRINT"[HOME]":END
 800 FOR ADRES=826TO1023:READ DATTA:POKE ADRES,DATTA:NEXT ADRES
READY.
```

Landmine

```
5 REM SET UP MINEFIELD IN ARRAYS
7 NM=10:REM NO MINES = 10 INITIALLY
10 DIM A$(15,16)
15 GOSUB 4000
20 FOR I=1 TO 15
30 FOR J=1 TO 15
40 A$(I,J)="."
50 NEXT J
60 NEXT I
70 REM SET UP MINES AT RANDOM
80 FOR I=1 TO NM
90 X%=RND(1)*15+1
100 Y%=RND(1)*15+1
110 A$(X%,Y%)="M"
115 NEXT I
117 A$(1,1)="."
119 A$(15,15)="."
120 PRINT"■■■■■■■■■■■■■■■■L A N D M I N E"
130 PRINT"■■■■■■■■■■■■■■■■────────────────"
135 PRINT"NO OF MINES = ";NM
140 REM PRINT MINEFIELD ON SCREEN
200 CL=15:NO=15:Z$=".":T$="H"
210 FOR RW=5 TO 19
220 GOSUB 1000
230 NEXT RW
240 CL=14:NO=17:RW=4:Z$="▒":T$="H"
250 GOSUB 1000
260 RW=20
270 GOSUB 1000
280 CL=14:NO=17:RW=4:Z$="▒":T$="V"
290 GOSUB 1000
300 CL=30
310 GOSUB 1000
320 X%=1:Y%=1
330 GOTO 540
340 GOSUB 3000
342 PRINT"●";
345 GET GO$:IF GO$="" THEN 340
350 GO=VAL(GO$)
360 IF GO<1 OR GO>9 THEN 340
370 ON GO GOTO 410,420,430,440,450,460,470,480,490
410 XX%=X%-1:YY%=Y%+1:CR$="■■"
415 GOTO 500
420 XX%=X%:YY%=Y%+1:CR$="■"
425 GOTO 500
430 XX%=X%+1:YY%=Y%+1:CR$="■■"
435 GOTO 500
440 XX%=X%-1:YY%=Y%:CR$="■"
445 GOTO 500
450 GOTO 340
460 XX%=X%+1:YY%=Y%:CR$="■"
465 GOTO 500
470 XX%=X%-1:YY%=Y%-1:CR$="■■"
475 GOTO 500
480 XX%=X%:YY%=Y%-1:CR$="■"
485 GOTO 500
490 XX%=X%+1:YY%=Y%-1:CR$="■■"
500 IFXX%<1 OR XX%>15 THEN 340
510 IFYY%<1 OR YY%>15 THEN 340
520 X%=XX%:Y%=YY%
530 REM NOW CHECK FOR MINES IN VICINITY
540 IF A$(X%,Y%)="M" THEN 830
545 A$(X%,Y%)="●"
550 IF X%=15 AND Y%=15 THEN 800
560 REM CHECK FOR 1 MOVE AWAY
570 MS%=1:GOSUB 2000
580 IF MN<>0 THEN 620
590 REM CHECK FOR 2 MOVES AWAY
600 MS%=2:GOSUB 2000
620 REM RESET CURSOR WITHIN MINEFIELD
630 GOSUB 3000
690 GOTO 340
700 REM MINEFIELD CROSSED SUCCESSFULLY
800 CL=1:NO=1:RW=23:T$="H"
810 Z$="MINEFIELD SUCCESSFULLY CROSSED      "
815 GOSUB 1000
820 GOTO 850
830 REM LANDMINE EXPLODES
835 CL=1:NO=1:RW=23:T$="H"
837 Z$="B A N G !!!!!!!                     "
840 GOSUB 1000
850 A$(X%,Y%)="X"
870 FOR Y=1 TO 15
875 X%=1:Y%=Y:GOSUB 3000
880 FOR X=1 TO 15
890 PRINT A$(X,Y);
900 NEXT X
920 NEXT Y
930 GET GO$:IF GO$="" THEN 930
935 NM=NM+2
940 GOTO 20
1000 REM SUBROUTINE TO DRAW LINE ON SCREEN
1005 REM CL=START COLUMN,RW=START ROW
1007 REM NO=NO OF CHARS,T$= V(VERT),H(HORIZ)
1009 REM Z$=TEXT TO PRINT
1010 PRINT"■"
1020 FOR I=1 TO CL-1:PRINT"■";:NEXT I
1030 FOR I=1 TO RW-1:PRINT"■";:NEXT I
1040 IF T$="H" THEN I$=""
1050 IF T$="V" THEN I$="■■"
1060 FOR I=1 TO NO
1070 PRINTZ$;I$;
1080 NEXT I
1090 RETURN
2000 REM CHECK FOR MINES IN VICINITY
2005 REM MS%=1 CLOSE SEARCH,MS%=2 WIDE SEARCH
2010 MN=0
2020 FOR I=X%-MS% TO X%+MS%
2030 FOR J=Y%-MS% TO Y%+MS%
2032 IF I<1 OR J<1 THEN 2050
2034 IF I>15 OR J>15 THEN 2050
2040 IF A$(I,J)="M" THEN MN=MN+1
2050 NEXT J
2060 NEXT I
2070 Z$="                                   "
2080 IF MN>0 THEN Z$=STR$(MN)+" MINES(S) "+STR$(MS%)+" MOVES AWAY"
```

continued over page

Landmine continued

```
2090 CL=1:NO=1:RW=23:T$="H"
2095 GOSUB 1000
2097 RETURN
3000 REM CURSOR ADDRESS TO WITHIN MINEFIELD
3005 REM X%=X AXIS, Y%=Y AXIS
3010 PRINT"■";
3020 FOR I=1 TO X%+14:PRINT"■";:NEXT I
3030 FOR I=1 TO Y%+4:PRINT"■";:NEXT I
3035 PRINT"■";
3040 RETURN
4000 REM INTRODUCTION & INSTRUCTIONS FOLLOW
4010 PRINT"■■■■■■■■■■■■■■■■L A N D M I N E"
4020 PRINT"■■■■■■■■■■■■■■■■┌──────────"
4030 PRINT"THE OBJECT OF THIS GAME IS TO"
4040 PRINT"TRAVERSE A MINEFIELD. YOUR MINE"
4050 PRINT"DETECTOR GIVES SOME WARNING"
4060 PRINT"OF NEARBY MINES. BUT CAN'T"
4070 PRINT"GIVE THEIR EXACT LOCATION."
4080 PRINT"SO BE CAREFUL OR :-"
4090 PRINT"■   B A N G !!!   ■"
4110 PRINT"YOU START AT THE TOP LEFT CORNER"
4120 PRINT"AND FINISH (HOPEFULLY) AT THE          BOTTOM RIGHT."
4130 PRINT"USE THE NUMERIC PAD TO MOVE,"
4140 PRINT"WITH YOUR POSITION BEING RELATED "
4150 PRINT"TO THE 5 KEY.■"
4162 PRINT"        7 8 9
4164 PRINT"         \|/
4166 PRINT"        4-●-6
4167 PRINT"         /|\
4168 PRINT"        1 2 3
4170 PRINT"■■PRESS ANY KEY WHEN READY.■"
4180 GET GO$:IF GO$="" THEN 4180
4190 RETURN
READY.
```

ALPHA SORTS

Here's the Shell-Metzner Alpha-Sort I announced last time.

But first a couple of notes and errata on the Shell-Sort issued in October.

You can fool the starting controls and most likely crash the system if you try to sort an integer-array. Same doom if you sort more elements than you actually dimensioned. So, please don't.

Now, suppose you have to sort an array made up of a double code; i.e. a six digits's normal code and a date (which has six digits too, in the form YYMMDD).

Even so, you get a ten digits' code, which isn't yet short enough. Or is it? Why, "everybody knows" a Pet has a nine digits' precision. That's true if you print or display them. Perhaps less people know that CBM floating-point variables offer up to ten decimal digits accuracy. Hurray!

Thus, our sort would be OK ... if ... if you only rush and correct a blunder that recurred a cool three times on the listings (blame and shame on me).
Line 32689 (7FB1) was :
CPY.IMM 04 (4)
should be instead:
CPY.IMM 05 (5)
Same correction on lines 32444 (7EBC) and 32465 (7ED1) of the second routine.
The short had only 8 digits of precision in the former (and wrong) version.

Due to the particular way in which string-variables are organized on the Microsoft-BASIC of CBM, it is possible to create a string-array with different lengths elements. That's most convenient in many circumstances and although, with BASIC 2.0 it causes the known "garbage collection" delay, it offers greater versatility than with other languages (COBOL, even other BASICS etc.).

The string-variables are split into two parts. The first part uses three bytes: one of these specifies the length of the variables (i.e. how many characters it contains) and the remaining two, in the usual form of LO-BYTE and HI-BYTE address, point to the second part in HIGH-RAM-MEMORY, containing the ASCII values of its characters. In fact, the RAM area starting from the TOP-OF-MEMORY and regressing to the END-OF-ARRAYS area is used as storage of the ASCII characters values of the string-variables.

A string-array consists exactly of a series of three bytes (length, LO and HI pointers) elements.

If two elements are to be compared, we have to get their ASCII values in high-memory: compare the first character of both of them, then the second, etc. and repeat the comparisons till we get to the last character of the shorter element. That's why we must compare the two lengths before any other thing, to know which one is shorter.

In many applications partial precision sorts are needed (for instance; only the first 5 characters of a 15 characters' code). In a more versatile version, one could ask any number of characters precision, starting at any position of the code. In our routine, it is possible to specify only the number of characters that are to be compared, by pokeing it into location 203.

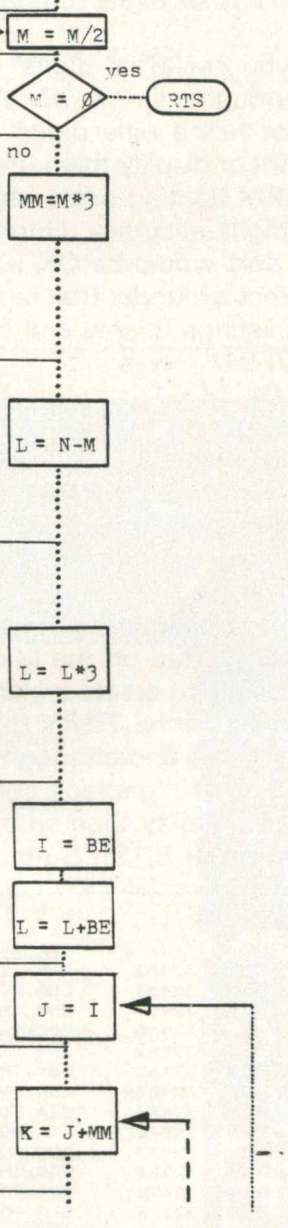
Poke 0 to location 203 if you need the greatest accuracy.

Nothing else to say, I guess, but renew the notes and hints given last time.

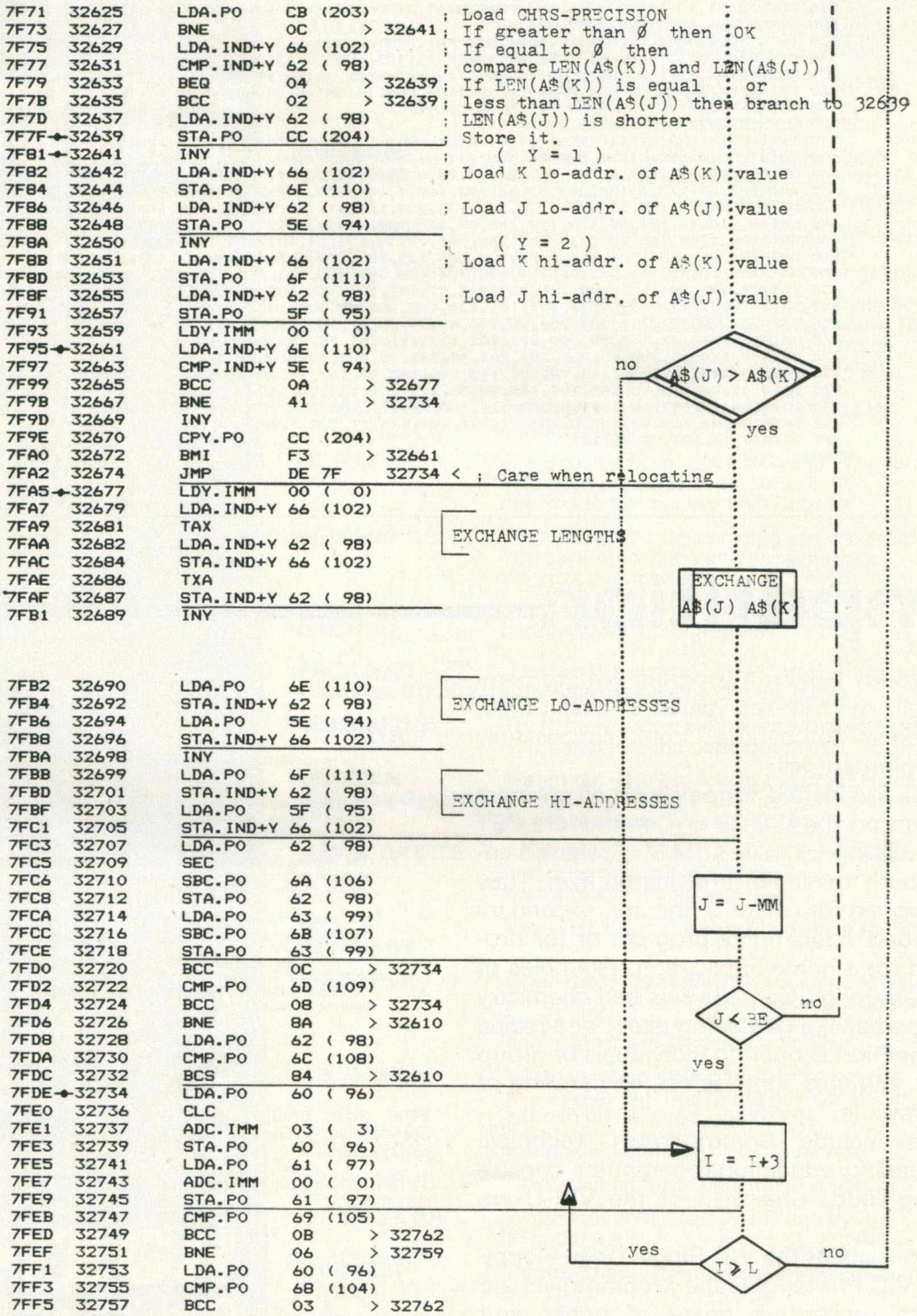
Fulvio Rissitano
V.le Oscar Sinigaglia, 15
00143 ROME
ITALY

```
                                                                  Notes for
                                                                  BASIC 2.0
7EC5  32453   JSR          2B C1     49451 <; Search N% address   6D CF on 2.0
7EC8  32456   LDA.PO       08 (  8)        ; Is it an integer?
7ECA  32458   CMP.IMM      80 (128)                                 M = N%
7ECC  32460   BEQ          03        > 32465  ; If it is  then OK  N = N%
7ECE-•32462   JMP          00 BF     48896 <; else  SYNTAX ERROR   03 CE on 2.0
7ED1-•32465   LDY.IMM      01 (  1)
7ED3  32467   LDA.IND+Y    44 ( 68)        ; Load hi-value of N%
7ED5  32469   STA.PO       00 (  0)        ; and store it.
7ED7  32471   STA.PO       CE (206)
7ED9  32473   DEY
7EDA  32474   LDA.IND+Y    44 ( 68)        ; Load lo-value of N%
7EDC  32476   STA.PO       01 (  1)        ; and store it.
7EDE  32478   STA.PO       CF (207)
7EE0  32480   JSR          2B C1     49451 <; Search A$(1) address   See above
7EE3  32483   LDA.PO       07 (  7)        ; Is it a string?
7EE5  32485   CMP.IMM      FF (255)                                 BE=SOA
7EE7  32487   BNE          E5        > 32462  ; If not then ERROR
7EE9  32489   LDA.PO       CB (203)        ; Load N.characters precision
```

```
7EEB  32491      LDY.IMM    03 (  3)          ; Compare it with the
7EED  32493      CMP.IND+Y  44 ( 68)          ; length of A$(2)
7EEF  32495      BEQ        04      > 32501   ; If equal then OK
7EF1  32497      BCC        02      > 32501   ; If less  then OK
7EF3  32499      LDA.IMM    00 (  0)          ; If greater, make it Ø
7EF5-●32501      STA.P0     CB (203)          ; OK,  store it.
7EF7  32503      STA.P0     CC (204)
7EF9  32505      LDA.P0     45 ( 69)          ; Is the 2nd parameter
7EFB  32507      STA.P0     6D (109)          ; inserted with the SEARCH
7EFD  32509      CMP.P0     2D ( 45)          ; an array?
7EFF  32511      BCC        CD      > 32462   ; If not then SYNTAX ERROR
7F01  32513      BNE        06      > 32521   ; Else store its address.
7F03  32515      LDA.P0     2C ( 44)
7F05  32517      CMP.P0     44 ( 68)
7F07  32519      BCS        C5      > 32462
7F09-●32521      LDA.P0     44 ( 68)
7F0B  32523      STA.P0     6C (108)
7F0D-●32525      LSR.P0     CF (207)
7F0F  32527      ROR.P0     CE (206)
7F11  32529      BNE        05      > 32536
7F13  32531      LDA.P0     CF (207)
7F15  32533      BNE        01      > 32536
7F17  32535      RTS
7F18-●32536      LDA.P0     CF (207)
7F1A  32538      STA.P0     6B (107)
7F1C  32540      LDA.P0     CE (206)
7F1E  32542      ASL.A
7F1F  32543      ROL.P0     6B (107)
7F21  32545      CLC
7F22  32546      ADC.P0     CE (206)
7F24  32548      STA.P0     6A (106)
7F26  32550      LDA.P0     6B (107)
7F28  32552      ADC.P0     CF (207)
7F2A  32554      STA.P0     6B (107)
7F2C  32556      LDA.P0     00 (  0)
7F2E  32558      SEC
7F2F  32559      SBC.P0     CE (206)
7F31  32561      STA.P0     68 (104)
7F33  32563      TAX
7F34  32564      LDA.P0     01 (  1)
7F36  32566      SBC.P0     CF (207)
7F38  32568      STA.P0     69 (105)
7F3A  32570      TAY
7F3B  32571      TXA
7F3C  32572      ASL.A

7F3D  32573      ROL.P0     69 (105)
7F3F  32575      CLC
7F40  32576      ADC.P0     68 (104)
7F42  32578      STA.P0     68 (104)
7F44  32580      TYA
7F45  32581      ADC.P0     69 (105)
7F47  32583      STA.P0     69 (105)
7F49  32585      LDA.P0     6C (108)
7F4B  32587      STA.P0     60 ( 96)
7F4D  32589      CLC
7F4E  32590      ADC.P0     68 (104)
7F50  32592      STA.P0     68 (104)
7F52  32594      LDA.P0     6D (109)
7F54  32596      STA.P0     61 ( 97)
7F56  32598      ADC.P0     69 (105)
7F58  32600      STA.P0     69 (105)
7F5A-●32602      LDA.P0     60 ( 96)
7F5C  32604      STA.P0     62 ( 98)
7F5E  32606      LDA.P0     61 ( 97)
7F60  32608      STA.P0     63 ( 99)
7F62-●32610      LDA.P0     62 ( 98)
7F64  32612      CLC
7F65  32613      ADC.P0     6A (106)
7F67  32615      STA.P0     66 (102)
7F69  32617      LDA.P0     63 ( 99)
7F6B  32619      ADC.P0     6B (107)
7F6D  32621      STA.P0     67 (103)
7F6F  32623      LDY.IMM    00 (  0)
```



```
7F71  32625      LDA.PO     CB (203)        ; Load CHRS-PRECISION
7F73  32627      BNE        0C      > 32641 ; If greater than Ø  then :OK
7F75  32629      LDA.IND+Y  66 (102)        ; If equal to Ø  then
7F77  32631      CMP.IND+Y  62 ( 98)        ; compare LEN(A$(K)) and LEN(A$(J))
7F79  32633      BEQ        04      > 32639 ; If LEN(A$(K)) is equal  : or
7F7B  32635      BCC        02      > 32639 ; less than LEN(A$(J)) then branch to 32639
7F7D  32637      LDA.IND+Y  62 ( 98)        ; LEN(A$(J)) is shorter
7F7F-●32639      STA.PO     CC (204)        ; Store it.
7F81-●32641      INY                        ;    ( Y = 1 )
7F82  32642      LDA.IND+Y  66 (102)        ; Load K lo-addr. of A$(K) value
7F84  32644      STA.PO     6E (110)
7F86  32646      LDA.IND+Y  62 ( 98)        ; Load J lo-addr. of A$(J) value
7F88  32648      STA.PO     5E ( 94)
7F8A  32650      INY                        ;    ( Y = 2 )
7F8B  32651      LDA.IND+Y  66 (102)        ; Load K hi-addr. of A$(K) value
7F8D  32653      STA.PO     6F (111)
7F8F  32655      LDA.IND+Y  62 ( 98)        ; Load J hi-addr. of A$(J) value
7F91  32657      STA.PO     5F ( 95)
7F93  32659      LDY.IMM    00 (  0)
7F95-●32661      LDA.IND+Y  6E (110)
7F97  32663      CMP.IND+Y  5E ( 94)
7F99  32665      BCC        0A      > 32677
7F9B  32667      BNE        41      > 32734
7F9D  32669      INY
7F9E  32670      CPY.PO     CC (204)
7FA0  32672      BMI        F3      > 32661
7FA2  32674      JMP        DE 7F   32734 < ; Care when relocating
7FA5-●32677      LDY.IMM    00 (  0)
7FA7  32679      LDA.IND+Y  66 (102)
7FA9  32681      TAX
7FAA  32682      LDA.IND+Y  62 ( 98)
7FAC  32684      STA.IND+Y  66 (102)
7FAE  32686      TXA
7FAF  32687      STA.IND+Y  62 ( 98)
7FB1  32689      INY

7FB2  32690      LDA.PO     6E (110)
7FB4  32692      STA.IND+Y  62 ( 98)
7FB6  32694      LDA.PO     5E ( 94)
7FB8  32696      STA.IND+Y  66 (102)
7FBA  32698      INY
7FBB  32699      LDA.PO     6F (111)
7FBD  32701      STA.IND+Y  62 ( 98)
7FBF  32703      LDA.PO     5F ( 95)
7FC1  32705      STA.IND+Y  66 (102)
7FC3  32707      LDA.PO     62 ( 98)
7FC5  32709      SEC
7FC6  32710      SBC.PO     6A (106)
7FC8  32712      STA.PO     62 ( 98)
7FCA  32714      LDA.PO     63 ( 99)
7FCC  32716      SBC.PO     6B (107)
7FCE  32718      STA.PO     63 ( 99)
7FD0  32720      BCC        0C      > 32734
7FD2  32722      CMP.PO     6D (109)
7FD4  32724      BCC        08      > 32734
7FD6  32726      BNE        8A      > 32610
7FD8  32728      LDA.PO     62 ( 98)
7FDA  32730      CMP.PO     6C (108)
7FDC  32732      BCS        84      > 32610
7FDE-●32734      LDA.PO     60 ( 96)
7FE0  32736      CLC
7FE1  32737      ADC.IMM    03 (  3)
7FE3  32739      STA.PO     60 ( 96)
7FE5  32741      LDA.PO     61 ( 97)
7FE7  32743      ADC.IMM    00 (  0)
7FE9  32745      STA.PO     61 ( 97)
7FEB  32747      CMP.PO     69 (105)
7FED  32749      BCC        0B      > 32762
7FEF  32751      BNE        06      > 32759
7FF1  32753      LDA.PO     60 ( 96)
7FF3  32755      CMP.PO     68 (104)
7FF5  32757      BCC        03      > 32762
7FF7-●32759      JMP        0D 7F   32525 < ; Care when relocating
7FFA-●32762      JMP        5A 7F   32602 < ; Care when relocating
```



Machine Code

```
500 REM       -  TESTING THE SHELL-METZNER M.L. ALPHA-SORT  -
502 POKE52,255:POKE53,125:CLR:X$=""+"                ":I=0:T=0:U=0:N=0:K=0:N%=0:A%=0
504 IFPEEK(32504)=204ANDPEEK(32723)=109THEN530
506 FORI=32453TO32764:READA%:POKEI,A%:NEXT:IFPEEK(48896)=PEEK(49451)THEN530
510 POKE32454,109:POKE32455,207:POKE32463,3:POKE32464,206:REM <<< BASIC 2.0
520 POKE32481,109:POKE32482,207
530 INPUT"HOW MANY ELEMENTS";N:PRINT:IFN>2085THEN530
540 DIMA$(N):A$(0)=X$:T=3+PEEK(42)+PEEK(43)*256:U=PEEK(T)+PEEK(T+1)*256
550 FORI=1TON:FORT=0TO8:POKEU+T,RND(I)*26+65:NEXT:A$(I)=X$:PRINTX$,:NEXT
560 PRINT:PRINT:INPUT"HOW MANY CHARACTERS PRECISION";A%:POKE203,A%
570 PRINT:PRINT" S  O  R  T  I  N  G ":K=TI
580 N%=N:SYS32453 N% A$(1)
590 T=TI:PRINT:PRINT"JIFFIES:";T-K:PRINT:FORI=1TON:PRINTA$(I),:NEXT:PRINT
600 PRINT:PRINT"THE TIME NEEDED WAS";(T-K)/60;"SEC.":END
610 DATA32,43,193,165,8,201,128,240,3,76,0,191,160,1,177,68,133,0,133,206,136
620 DATA177,68,133,1,133,207,32,43,193,165,7,201,255,208,229,165,203,160,3,209
630 DATA68,240,4,144,2,169,0,133,203,133,204,165,69,133,109,197,45,144,205,208
640 DATA6,165,44,197,68,176,197,165,68,133,108,70,207,102,206,208,5,165,207
650 DATA208,1,96,165,207,133,107,165,206,10,38,107,24,101,206,133,106,165,107
660 DATA101,207,133,107,165,0,56,229,206,133,104,170,165,1,229,207,133,105,168
670 DATA138,10,38,105,24,101,104,133,104,152,101,105,133,105,165,108,133,96
680 DATA24,101,104,133,104,165,109,133,97,101,105,133,105,165,96,133,98,165
690 DATA97,133,99,165,98,24,101,106,133,102,165,99,101,107,133,103,160,0,165
700 DATA203,208,12,177,102,209,98,240,4,144,2,177,98,133,204,200,177,102,133
710 DATA110,177,98,133,94,200,177,102,133,111,177,98,133,95,160,0,177,110,209
720 DATA94,144,10,208,65,200,196,204,48,243,76,222,127,160,0,177,102,170,177
730 DATA98,145,102,138,145,98,200,165,110,145,98,165,94,145,102,200,165,111
740 DATA145,98,165,95,145,102,165,98,56,229,106,133,98,165,99,229,107,133,99
750 DATA144,12,197,109,144,8,208,138,165,98,197,108,176,132,165,96,24,105,3
760 DATA133,96,165,97,105,0,133,97,197,105,144,11,208,6,165,96,197,104,144,3
770 DATA76,13,127,76,90,127
READY.
```

Competition

Commodore today launch a program writing competition - to all - with prizes valued at over £1500, and the chance to benefit from successfully marketed winning entries.

Submissions are invited for any type of program designed to run on the VIC-20 or Commodore PET 4000 series - utilising up to 32K RAM. Preferred entries will be both inventive and instructive. They should be designed for use as a primary, secondary or special schools educational program or for programs devised for a home application. Examples of program content range from physics and chemistry packages to, perhaps, a gardening diary, or a recipe file. The competition is open to individuals or group school entries. Multiple submissions on cassette or disk are premissable.

The judges include Commodore's Technical Manager, a leading educational computer consultant and Mike Todd, Chairman of the VIC-Users Group.

First prize consists of the VIC Single Drive Floppy Disk unit, the VIC Printer plus the programmers aid cartridge. With runners-up prizes of books and magazines about the VIC-20 Colour computer and composite disks of the winning entries, the total prize fund exceeds £1500. All entries should be sent to Commodore Software Competition, 35 Garway Road, London W2 4QD no later than 30th June 1982.

RULES

1. The competition is open to all individuals and school groups resident in the U.K. and Ireland.
2. The closing date for entries to be received by Commodore is 30th June 1982. All entries must be accompanied by the magazine coupon and sent to the Commodore Software Competition, 35 Garway Road, London W2 4QD.
3. The winning entries will be decided at the discretion of the judges and their decision will be final. No correspondence will be entered into. Entries will only be returned upon written request.
4. The winners will be notified in writing. Names and addresses of winners and runners-up will be supplied upon request.
5. The prizes will be presented at a formal presentation during August, 1982.
6. Commodore Business Machines (UK) Ltd reserve the right to market entries in association with the originator(s).

Send your program writing entry to: Commodore Software Competition, 35 Garway Road, London W2 4QD. Entries only valid with this coupon.

NAME: .....

ADDRESS: .....

.....

.....

.....

ENTRY TITLE: .....

VIC GAMES

You've tried the others now have the best!

Now available - the super-value fun games, exclusively for the VIC 20 Computer

SUPER-BREAKOUT	A brilliant version of the 'Arcade' game	**£7.50**
HEAD ON	Drive round a track – try to avoid the Vic car hitting you head on!	**£8.75**
VIC CUBE	Can you solve the Rubic Cube? A great test of skill.	**£8.75**
CRAZY BALLOON	Guide the swaying balloon through a prickly path. An addictive game.	**£8.75**
MOLE ATTACK	A great game. Quick reflexes are needed to shoot the moles as they pop out of their holes.	**£8.75**
BALLOON BOMBER	Destroy the air ships before they drop their bombs on you. An exciting game.	**£8.75**
SKIER	Ski down the Olympic run. The better the course you take the faster you go. Try to improve your record each time.	**£8.75**
TREASURE CARRYING	Your man wants the loot! – Get him back alive as often as possible avoiding the hail of arrows. Very funny and addictive game.	**£8.75**
SUPER MOON LANDER High Res Graphics	Make a soft landing on the moon. Watch out for the mountains! You control thrust direction etc. As good as the arcade game.	**£8.75**
OTHELLO	This is an 8 x 8 board game played with black and white counters. Pit your wits against the computer. Excitement mounts with each move! The last move can win the game!	**£8.75**
SPACE SHOOT	Defend the earth from being invaded by fiendish aliens!	**£8.75**
I.C.B.M.	Fire a rocket from your battleship! Navigate it through the attacking anti-ballistic missiles to hit your enemy's targets.	**£8.75**
INDY 500	Your Grand Prix car is racing against others. You control steering and acceleration. Very good graphics.	**£8.75**
SLOT MACHINE	Go for the Jackpot!! Just like one-arm bandits! Very good graphics.	**£8.75**
PACK MAN High Res Graphics	Just like the arcade game! Excellent graphics. Currently the best selling Vic game.	**£8.75**
SUBMARINE	Try to blow up the submarines before they sink you. Great fun and good graphics.	**£8.75**
GUNMAN High Res Graphics	Watch out – He's quick on the draw! Have a gunfight against the gunman controlled by the computer. Great graphics.	**£8.75**
NAVAL BATTLE	An exciting game! You have submarines and and guns against you. Try to win! It's tough but it can be done!!	**£8.75**
SPACE INVADERS	Yes–THE Space Invaders!! Need we say more?	**£8.75**
FIRE TREK	Open the force screen on your ship to search for the enemy. When located the ship's computer works out range, energy etc. to help you annihilate them. Amazing graphics!	**£8.75**
ALIEN WARS	When in range, your on-board computer locks on target. This game has 'got to be seen to be believed' graphics!!	**£8.75**
VIC COVER	Keep your keyboard clean. Dust free.	**£3.95**
CARE DRIVE	Drive your car through a forest track.	**£7.50**
3-D MAZE (U)	Walk through a maze in 3-D and try to find the exit.	**£7.50**
INVADER FALL(U)	Destroy the invaders before they destroy you.	**£7.50**
DRAGON MAZE (U)	Can the dragon eat you before you get out of the maze?	**£7.50**
ALIEN MAZE (U)	Build your force field against the 3 aliens after you.	**£7.50**
MISSILE COMMANDER (U)	Fire your A.B.M. at I.C.B.M. coming in to destroy your cities.	**£8.75**
3K RAM PACKS	are required for (U) Games. We are giving a FREE GAME with every purchase.	**£29.50**

Buy direct from us or your local dealer

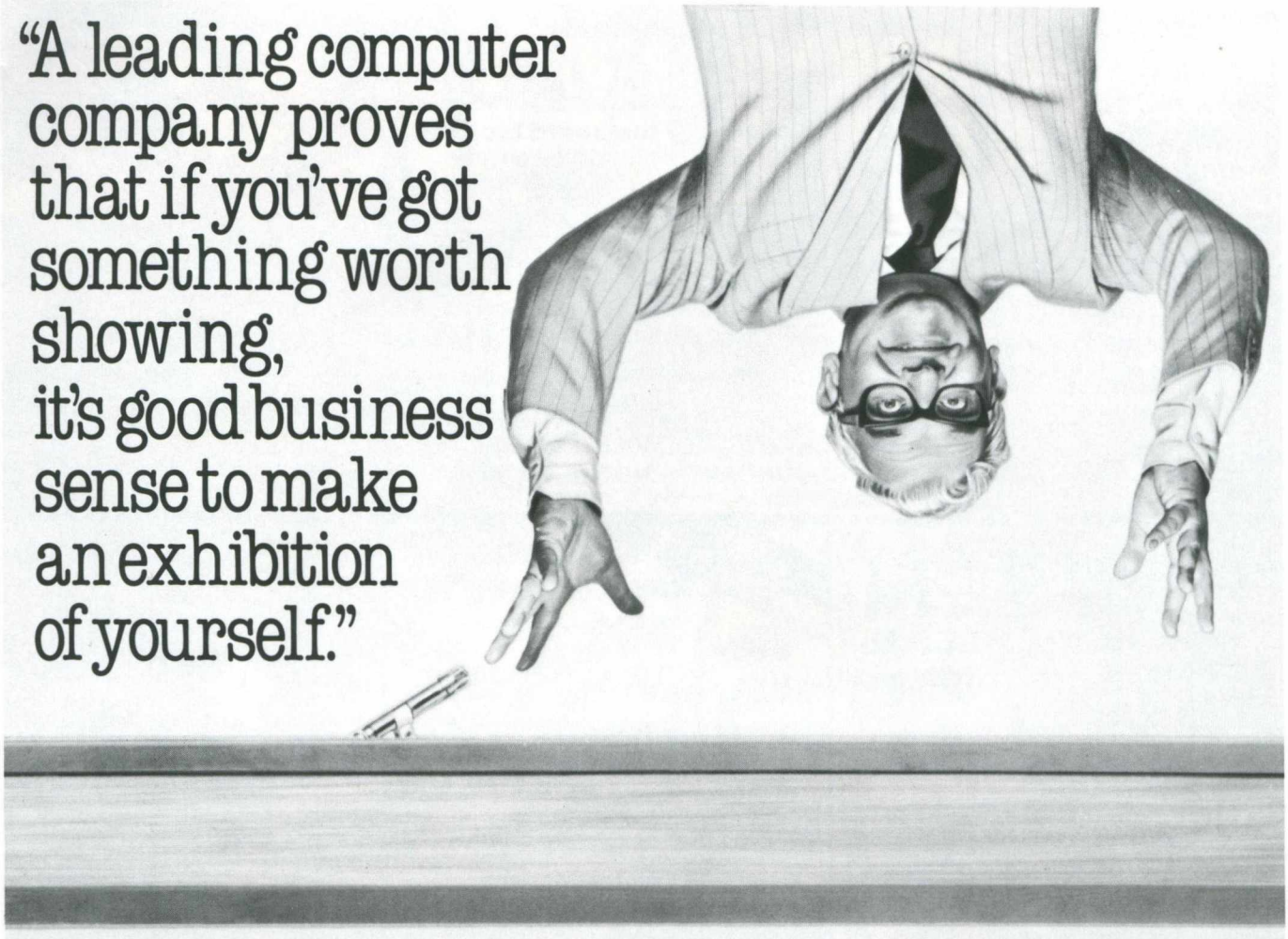
No extra costs - the price includes VAT, post & packing

Barclaycard/Visa Access Welcome

HI-TECH COMPUTERS LTD

DEPT B
7 Queensway, Hemel Hempstead
Herts. Tel: **(0442) 50450**

"A leading computer company proves that if you've got something worth showing, it's good business sense to make an exhibition of yourself."



THE THIRD INTERNATIONAL COMMODORE COMPUTER SHOW OPENS JUNE 3RD

The Commodore Show is one of the finest opportunities to see the best of today's micro-computer systems in action. A wide range that covers our home colour computer and our selection of sophisticated business systems as well as the latest in software and related products.

Also we'll demonstrate just how they can improve efficiency, whatever line of business you're in.

There'll be seminars on education, communications and a wide variety of business applications.

Guests include knowledgeable and interesting people like Jim Butterfield, who's the foremost authority on the PET and its capabilities.

All in all, it's the biggest and best Commodore Show yet, and definitely not to be missed.

See us at:
The Third International Commodore Computer Show, Cunard Hotel, Hammersmith, London.

Thursday June 3rd	12am - 6pm
Friday June 4th	10am - 6pm
Saturday June 5th	10am - 5pm

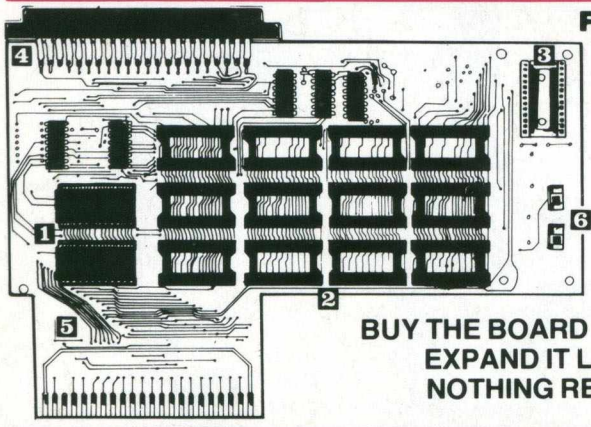
For further details ring
SLOUGH (0753) 79292
675 Ajax Ave, Slough, Berks.

commodore COMPUTER

Quite simply, you benefit from our experience

THIS SHOULD BE THE ONLY MEMORY BOARD YOU WILL EVER NEED FOR YOUR VIC!

£49 +VAT



Features of the board include:

1. 3K Memory hi-resolution area
2. 24K Expansion area, just plug in the chips to expand memory!
3. Socket for VicKit or other roms
4. Memory port reproduced
5. High quality gold plated edge, plated thro' hole, glass fibre board
6. Switch to isolate 3K memory area

Board shown with covers removed

*Expandable board fully socketed with 3K RAM fitted.

NO NEED FOR A MOTHERBOARD OR EXPANSION CHASSIS

BUY THE BOARD WITH THE MEMORY YOU NEED NOW!
EXPAND IT LATER! ADD VICKIT OR VICKIT II
NOTHING REDUNDANT! NOTHING WASTED

£25

VICKIT

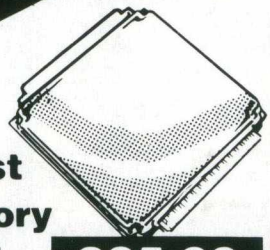
offers HELP to programmers

it also offers AUTO, DELETE, DUMP, FIND, HELP, OFF, RENUMBER, STEP, TRACE
Example:-

```
RUN

?DIVISION BY ZERO ERROR IN 500
READY.
HELP
 500 J = SQR(A*B/C)

READY
```

HELP How many times have you wanted to scream "HELP!" when your VIC couldn't interpret your program and all it would say was **?SYNTAX ERROR**? Fret no longer: Now just type in **HELP** The line on which the error occurs will be shown and the erroneous portion of the line will be indicated in reverse video on the screen. Truly a great help in any learning or school situation.

NOW AVAILABLE



VIC- Low cost 3k Memory

£25.99

Lowest Cost!
Compact Size!
Gives 6½k user Ram!
Allows high resolution graphics to be programmed!
Complete user instructions included!

VIC- Light Pen

OFFICIALLY APPROVED BY COMMODORE

This high quality light pen works in both normal and Hi-Res modes on the Vic allowing simple interaction with the Vic without keyboard entry. Easy to program and easy to use. e.g. Menu selection, Non-keyboard entry, Teaching, Games.
Feature: Touch sensitive "Enter" contacts to eliminate accidental entry.

£25.00

VICKIT II

£29

NEW

At last, Hi-resolution Graphics Made Easy
In addition to all the features of VICKIT, VICKIT II has.....

TEXT-	swap between the hi-resolution mode and normal text mode
GRAPHICS-	(TEXT automatically set on ERROR)
LINE-	for drawing lines and boxes (filled or unfilled)
CLEAR-	to clear the graphics screen
DRAW-	to draw lines (more than 1 at a time)
PUT-	save and restore the contents of a rectangular area on the hi-res screen into and from an array
FILL-	fills an area on the screen
SET-	sets a point on the hi-res screen
POINT-	tests whether a point is set on the hi-res screen

VICKIT or VICKIT II plugs into ROM Switchboard or Stack Memory Expansion Board

---

SPECIAL INTRODUCTORY PRICE Only £64.00

Fully socketed board (with cover)
plus **3K hi-res area memory** plus **VICKIT**

plus £1.50 p&p or £5.00 Securicor
(Prices and p&p subject to VAT)

PRICES:-

BOARD with 3K	**£49.00**
BOARD with 8K	**£69.00**
BOARD with 8K plus 3K	**£85.00**
BOARD with 16K	**£99.00**
BOARD with 16K plus 3K	**£115.00**
BOARD with 24K	**£129.00**
BOARD with 24K plus 3K	**£145.00**

SPECIAL
Add £15 plus VAT for VICKIT
if ordered with any memory
(Usually £25 plus VAT if supplied separately)

Additional Memory
8K (4 chips) **£39.00** 3K (2 chips) **£20.00**

Stack Computer Services Limited

290-298 Derby Road, Bootle, Liverpool 20
Telephone: **051-933 5511**
Telex: **627026 (STACK G)**

Please add £1.50 p&p or £5.00 Securicor plus VAT
Prices exclude VAT.
Please make cheques payable to Stack Computer Services Ltd.

24hr answerphone service.
Please ring us day or night on
051 933 5511

VISA

STACK COMPUTER SERVICES LTD

ASK YOUR LOCAL COMMODORE VIC DEALER FOR DETAILS OF STACKS VIC PERIPHERALS