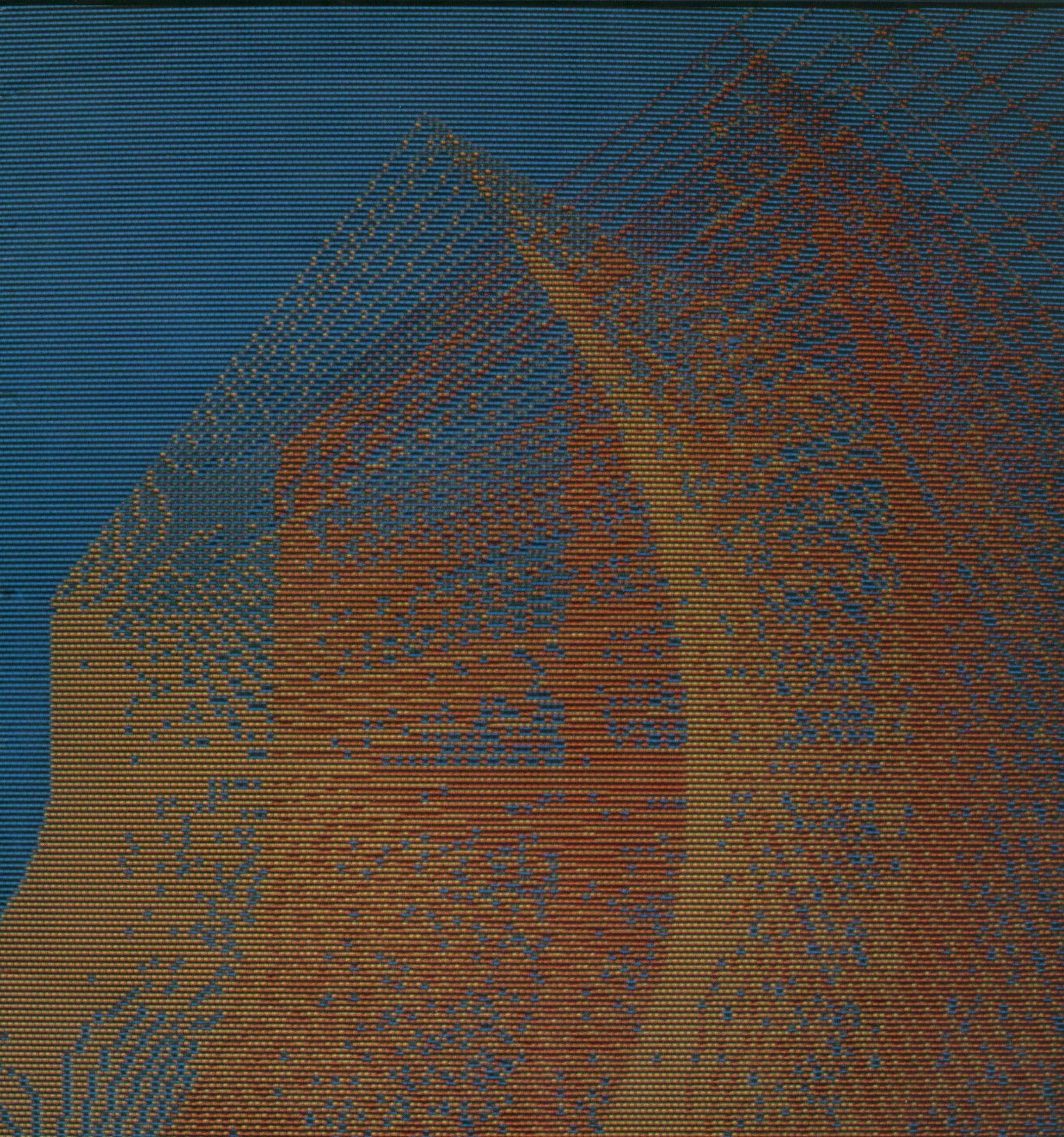
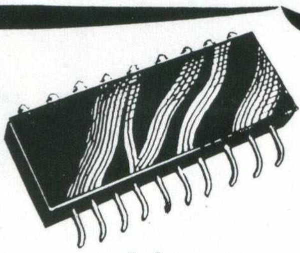
commodore
COMPUTING
international

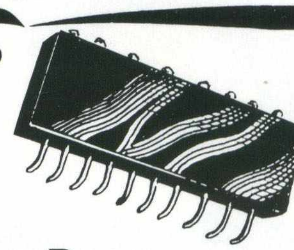
June 1982 £1.00



The independent magazine for Commodore computer users



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)

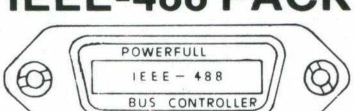
*** EDEX 2.0 only □ EDEX 4.1 only**

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...)

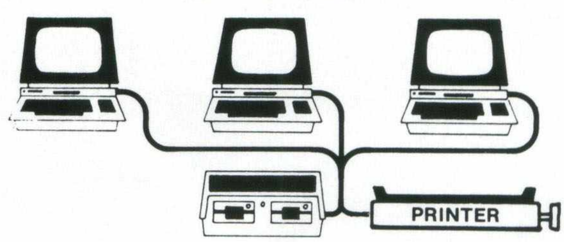
IEEE-PACK allows the use of IEEE-488 Universal Commands:

- 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 AMERICAN EXPRESS BARCLAYCARD VISA

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

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

Contents

Editorial

Editor
Pete Gerrard

Advertising Manager
Peter Chandler, 01-839 1881

Editorial Assitant
Fiona McCormick

Production
Three's Company and
Edwin Snell printers

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 Commodore 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.

The computer industry is in a constant state of flux, with new companies and computers coming onto the market seemingly daily. But a number of the old stalwarts manage to keep rumbling on, and indeed Commodore is one of those. With the ever present PET to stabalise the selling scene, (relative) newcomers like the VIC to enhance matters, and the promise of some superb new machines later this year, Commodore seem set to remain at or near the top for a good few years to come.

Certainly, a company that can afford to put on such an extravagant affair as the Third International Commodore Computer Show can't be doing too badly. At the time of going to press, it looks like the Show will be covering an enormous 20,000 square feet of floor space! For one company to be able to do this in our present economic climate is quite remarkable.

I strongly urge you to go along to the Show. The last two (three really, but no-one, not even Commodore, seems to remember the first, very low key effort) have been eye openers in terms of just what is going on, and the sheer volume of work being done on PETs; increasingly VICs as well. The list of applications, as they say, is endless!

It seems that Commodore is the only company capable of putting on anything of this size. This year, for the first time ever, Apple are attempting something similar. However, 20,000 square feet at the Cunard Hotel in Hammersmith is a long way removed from the Fulcrum in Slough, which is where Apple are having their bash. It will be interesting to see how that goes, but I can't see it being anything like the success that the PET Show (sorry Commodore, the name's stuck!) is virtually guaranteed to be.

As a footnote, goodbye to Clive Booth, ex-Applications Manager at Commodore, as he moves on to new ground. He was chiefly responsible for organising last year's Show, and will have a major part to play in this year's. As an ex-colleague at Commodore, I wish him all the best.

Letters

Dear Sir,

I thought you might like to consider the attached program 'oddity'. The first loop of the program produces the correct result, but the second loop does not. As you will see the calculation only extends to two decimal places, hardly mind-bending? Also, try raising each calculation in line 60 by the power of 1.
It may also interest you to know that besides failing on my Commodore 8032, my Sinclair ZX81 produces exactly the same error.

I would be most interested in your comments.
Yours faithfully
S. J. McFadyen
West Sussex.

The listing:—

```
 5 REM *** LOGIC ERROR
PROGRAM ***
 10 LET A=15
 20 LET B=100
 30 LET C=1020
 40 LET D=12
 50 LET E=12.75
 60 PRINT "CALCULATION =
";((D*E)-((A/B)*C))
 70 LET X=D*E
 80 LET Y=(A/B)*C
 82 PRINT "X=";X;"
Y=";Y;" X-Y=";X-Y
 85 IF E=8.65 THEN GOTO
130
 90 LET E=8.65
100 LET C=692,00
110 GOTO 60
130 STOP
```

Dear Mr. McFadyen,

The error that occurs when the program is run is quite a simple one to explain, but an annoying one to deal with. The PET stores all its numbers not in decimal form as you type them in, but in binary. So the PET sees fractions like .10 or .68 as endlessly repeating fractions. To fit the fractions in memory, it must first trim it, so many fractions are adjusted slightly before storage and calculation. You will find that this error will occur on any microcomputer. To get round the problem, just alter all calculations as in the following example:—

```
 70 X=INT ((D*E)*100+.5)
```

and then

```
 82 PRINT "X=";X/100 etc
.....
```

This will then produce the desired results.

Dear Sirs

Please could you send me details about the PET Users Group, and the kind of work that they do.

Yours faithfully
Miss S. M. Oakes
London

Dear Miss Oakes,

A short query, but a longer answer! In this issue of the magazine you can read about the sort of work that one of the PET User Clubs, ICPUG South East, do, and how they've evolved over the years. Commodore's own PET User Club, which never really amounted to much anyway, other than as a body producng the old CPUCN, died a death when the magazine left Commodore to come to Nick Hampshire publications. Consequently, Commodore are now quite content to let the Independents do the work. Their main contact as a source of information is:

Mrs. Eli Pamphlett
7 Lower Green
Tewin, Welwyn
Hertfordshire

Dear Sir,

I enclose a listing of a routine for use when debugging machine code programs. It causes a jump to the machine code monitor whenever the stop key is pressed. As such it is the software equivalent of the 'Butterfield reset', with the advantage that the stack pointer does not need to be modified when restoring operations.
The routine modifies the interrupt routine. It is designed for PET Basic 2, and the jump vectors may need to be modified for other versions.
It is fully relocatable, but like all modifications to the interrupt system it is best located so that the first instruction starts with the low order byte of its address at $xx2E (Basic 2 roms). It can then be enabled by poking the high order address byte ($xx) into $0091 without special routines to 'suspend' the interrupt system. Don't forget to disable it before attempting input/output operations on cassettes.
I do my programming with the Supermon assembler, and rather than guess at forward branches I put them into a closed loop (e.g. $6000 BEQ $6000) and resolve them later. If I forget, 'Stop Key' gives the location of the closed loop in the PC register.
Yours sincerely

Donald Skene
Maidstone.

Dear Donald,

Thanks for an interesting letter, and an interesting routine!

The listing :—

```
7F2E  AD 10 E8   LDA $E810
7F31  29 F0      AND #$F0
7F33  09 08      ORA #$08
7F35  8D 10 E8   STA $E810

7F38  AD 12 E8   LDA $E812
7F3B  CD 12 E8   CMP $E812
7F3E  D0 F8      BNE $7F3B

7F40  49 FF      EOR #$FF
7F42  29 21      AND #$21
7F44  D0 22      BNE $7F68

7F46  EE 10 E8   INC $E810

7F49  AD 12 E8   LDA $E812
7F4C  CD 12 E8   CMP $E812
7F4F  D0 F8      BNE $7F49

7F51  49 FF      EOR #$FF
7F53  29 10      AND #$10
7F55  F0 11      BEQ $7F68

7F57  AD 12 E8   LDA $E812
7F5A  CD 12 E8   CMP $E812
7F5D  D0 F8      BNE $7F57

7F5F  49 FF      EOR #$FF
7F61  29 10      AND #$10
7F63  D0 F2      BNE $7F57

7F65  6C 92 00   JMP ($0092)

7F68  4C 2E E6   JMP $E62E

To enable  :    POKE 145,127

To disable :    POKE 145,230
```

New Product News

New Range of Computers

At the recent Hanover Fair, Commodore announced a whole new range of micros, covering the entire spectrum of the market. The VIC 10 and Commodore 64 are covered in our hardware review this month : here we'll take a brief look at the other newcomers.

A new printer was amongst the new arrivals. Branded as the 8300, it is a letter quality daisy wheel printer originally from Diablo. At 1395 pounds it is a trifle expensive, but has a number of interesting features as compensation. Print speed is 40 characters a second (bi-directional), with 2 baud data transfer rates, three pitch settings, and a number of detectors built in (paper out, end of ribbon etc.). Available now, according to Commodore.

Other peripherals announced were a collection of new disk drives, starting with the 8250. This is essentially a superior version of the 8050, having a total storage capacity of 2 megabyte : relative record size has been extended to 1 megabyte. The beast is read/write compatible with existing 8050 Discs. At 1295 pounds, this is expected to become available very shortly.

The next two up, for reasons best known to Commodore named the 9060 and the 9090, are respectively 5 megabyte and 7.5 megabyte drives. Retail prices are 1995 and 2495 pounds, with availability expected to be around the end of May. Apart from this, there is not an awful lot known about these drives: we'll try and find out for next time.

And so onto the machines themselves, with a look at the VIC 30. Not too much to say here, as it's basically a 16K version of the Commodore 64. All the graphics and sound facilities are there, Basic is built in (unlike the VIC 10), and it has an estimated retail price of 250 pounds, although in light of the Spectrum this may well drop. Anticipated delivery is January of next year, but again this could well be speeded up.

The 500 Series

The first of the new 'big' machines from Commodore. It utilises the same graphics and sound capabilities as the VIC 10, but a significant advance on that particular machine. Designed to connect up to a standard television, the colour screen display is a conventional 40 x 25 characters. On board RAM come at 64K, but this is expandable up to 256K, with the possibility of going ultimately to more than 750K! The keyboard is reminiscent of the 8032, but with 10 programmable function keys along the top.

With an IEEE port, RS232C port, the now familiar User Port, and a second processor slot built in, the 500 has exciting communications facilities. With these, it is capable of running any of the existing Commodore peripherals, and the RS232 allows access to a number of specialised devices. Also, of course, using the second processor slot means that you can run the machine under CP/M and operating system : a significant advance.

The price of the 500 is expected to be 695 pounds. Remember the first 8K PET, which came out at a very similar price? Things have certainly advanced since then!

The Commodore 720

This is the star of the show. With a built–in 80 x 25, tilt and swivel black and white monitor, keyboard very similar to the 500 series but detachable, built in twin disk drive with a storage capacity of 340K, and on–board RAM of 256K, the Commodore 720 has got everything going for it. It also has the traditional IEEE 488 port and 8 bit user port, as well as an RS232C port and (as in the 500) a second processor port. Is this finally to be the emulator machine that so many rumours have floated around.? There is (in addition to everything else) a DMA

The new look Commodore 720

which enables you to access large capacity disk drives: Winchester for instance.

Estimated price of this is 1595 pounds, with a delivery scheduled for September of this year.

We'll endeavour to bring you a more in-depth look at all of these machines in a future issue.

Stack Motherboard

The cheapest motherboard on the market, selling at well below the price of Commodore's own board, or the Arfon board. A retail price of just 24.99 pounds (plus VAT) makes this a very good bargain for the Vic user who wants to expand his system. If Stack can do it for this price, why can't everyone else?

But there is a snag. In order to use this motherboard, you have to buy a product known as the Storeboard, available from ... guess who? Stack, is the surprising answer.

The motherboard fits into the port at the rear of the Storeboard, and enables 4 additional cartridges to be hooked up to the Vic. Each cartridge slot is switch selectable, so any one cartridge, all four, or any other combination can be used at the same time. Thus your Vic can quite happily be expanded up to its maximum addressable memory size of 32K.

Similarly, because of the switch selectability you can leave a number of games cartridges sitting in the motherboard and just select the one you want. This solves the problem of constantly inserting and removing cartridges, and the corresponding wear and tear that occurs on the edge connectors.

The first public viewing of this board will be at the Third International Commodore Computer Show (The Pet Show to you and me) in early June, and available from dealers after that.

Stack Storeboard

A new idea in Vic peripherals, the Storeboard is a boxed printed circuit board which can take up to 27K of RAM fitted internally beneath a detachable cover. Well designed (and well built) the unit slots into the memory expansion port on the Vic, and stands at the same level. Using gold edge connectors, it ensures good contact with the Vic, and a further point in its favour is that it doesn't require any extra power when in use.

Coming with 3K of RAM of its own on-board, this immediately allows you to properly implement high resolution graphics. To upgrade further, 8K RAMpacks are available (again from Stack, at 29 pounds), and these fit inside the Storeboard. By using the connector at the back of the Storeboard, Stack's motherboard can be slotted on to allow you to use existing 3K, 8K, or 16K expansion RAM cartridges.

Also on board is a socket which allows games ROMs, or something like VICKIT, to be used. This is positioned outside the cover, to allow easy access to, and exchange of, any eproms you may care to use.

The Storeboard costs 49.00 pounds (plus VAT), and is available from any VIC dealer.

Vickit II

Another Stack product, this is a Vic equivalent of the Toolkit, which proved so popular in the early days of the Commodore PET. It adds a number of commands to the existing built-in commands of Commdore Basic, including Auto, Delete, Dump, Find, Help, Renumber, Step and Trace. This makes it significantly easier to develop and de-bug Basic programs on the Vic.

In addition to this there are an additional nine graphics commands built in, to help in using high resolution graphics. These make the use of hi-res much easier, and allows you to produce quite intricate patterns and designs on the screen. Consequently your programs can now start making proper use of the facilities of the Vic.

The product costs 29 pounds, and comes in ROM form. Thus it can be used with the Storeboard, for instance.

This is available from any Commodore Vic dealer.

Word Processors

Space precludes a detailed review this month, but two new word processing packages deserve at least brief mention. The first is called Superscript, available from the Independent Commodore Products User Group (ICPUG for short), at the remarkable price of 30 pounds to ICPUG members, 35 to non-members. With all the features of Wordpro and Wordcraft, plus a lot more, the ability to have up to 240 columns of text in memory at any one time, this is going to cause quite a stir amongst the manufacturers of word processing programs. More next month, and indeed more for Wordform 2, the latest from Landsoft. We carried a detailed review of his first program in an earlier issue, but this one is a definite step forward. Possibly the nicest feature is the ability to move a window of text, and then move that window to wherever you decide to put it. Again, we'll carry more details next month.

Club News

ICPUG : South East Region

The Independent Commodore Products Users Group (recently renamed from the Independent PET Users Group) has a number of regional ''off-spring''. Here we take a look at the history and development over the years of one such users group: the South Eastern Regional branch of ICPUG.

IPUG national sent Mick Ryan in Sevenoaks down in Kent a list of IPUG member names and addresses from which he extracted 57 names living in Kent, Sussex and south of the Thames London. Mick was already on the committee of the North Kent Amateur Computer Club, which held its meetings at a school in Biggin Hill. As the hire of the room from the Adult Education Centre was only one pound per night, and it was reasonably near to the centre of gravity of the 57 IPUG members, it was decided to use Biggin Hill for the first exploratory meeting, which took place on the 13th of May 1980.

30 of the nearest members were telephoned, as this was the cheapest and quickest form of contact. The meeting was an open evening for NKACC, where PET Users were showing their machines, and about 25 people attended.

Encouraged by this success, a more formal and separate evening was arranged for the 4th June 1980: again based on a PET program and ideas swapping evening. All 57 members were sent a circular, and about 35 attended. More names than were on the original list were now appearing, and many of that original list never responded. At this meeting members were asked what they wanted from the group. Many offered help, and most were content to accept suggestions for the way ahead.

Jim Butterfield

Thanks to the excellent co-operation of Helen Elsam from Commodore, Mick was able to meet Jim Butterfield at the Cafe Royal PET Show (those were the days!) and he kindly agreed to address the first proper meeting. A good start! Jim stayed the night at Mick's place, and Mick delivered him to Gatwick Airport en route to his next appointment in Paris. The evening, June 25th 1980, was of course a resounding success. It was opened up to the NKACC and the Croydon ACC, and about 60 people attended.

By this time they had been lucky enough to have Bill and Maureen Coles join them, with offers of help. Bill and Maureen were amateur, but at the same time very professional, printers in their spare time. They offered typing, printing and addressing at cost price, and took over the newsletter production from the Butterfield evening circular onwards. By this time the club had about 55 members who had formally registered an interest. Many were from the original NKACC. One regular used to travel all the way across Kent from Sandwich!

First Newsletter

Jim Butterfield proved difficult to follow. However, members main interest seemed to be obtaining help with their programming problems. Commodore seemed very helpful, so for the cost of a return fare, and a supper presented by Mick's long suffering wife, Commodore's resident software expert Paul Higginbottom readily agreed to run a programmers clinic on the 16th July. This was announced in the first formal newsletter. Members were asked for any queries they might have; although only two responded one provided a marvellous list of questions which was of general interest to everyone anyway. These were given to Paul in advance, and the rest of the points came up on the night. Paul was very helpful, and it was a popular evening.

Behind the scenes there were dramas over room availability, as the Adult Education organisation closed early for the summer holidays. This convinced Mick that the time was right for some help from the organising committee. Committee members needed to be reliable, willing, useful, and living close to Sevenoaks. It was no use asking for votes at this early stage in the club's development: ''military democracy'' had to be executed in 30 seconds before Paul began his clinic. ''You, you and you'' were volunteered by Mick for the committee, and agreed by the 40 or so members present before they could object! (Actually, they had been asked beforehand). Bill and Maureen Coles were naturals, for all their work on the printing and distribution of the newsletter. Reg Ivory, a Bank Manager with Lloyds International, readily 'volunteered' along with his son John (who at 14 had already sold his programs commercially), to represent the youth in the club.

Kevin Viney, the resident PET machine code expert in the NKACC committee, and then the proud owner of Wordpro 3, was another natural. Ron Plater was the only member brave enough to volunteer from the floor of the meeting. Their greatest coup was in persuading Harry Broomhall to join the committee as technical adviser. Harry is the U.K.'s 'Jim Butterfield', and acknowledged by Commodore to know more about the PET disk system (Ed. and everything else!) than anyone. At the same time the 'meeting' agreed that three pounds was not excessive for a subscription which included 6 newsletters a year.

First Committee Meeting

Mick held the first committee meeting at his house on the 15th August 1980. It was a long affair, but covered all the necessary details for administering the group and planning the rest of the program until the end of the year.

IPUG national were getting nervous at this stage that with so much enthusiasm in the South East they were going to do a 'SUPA' on them and break away. The committee hurriedly assured them that this was not considered in the best interest of PET users, and was certainly not the intention. The IPUG constitution was adopted, and IPUG national kindly agreed to the use of their logo (with the addition of South East) to be used on the newsletter and letterheads

Growing Membership

There were now 60 names on the mailing list. Total set-up costs amounted to twenty five pounds, and this was refunded by Commodore through IPUG national.

The crucial decision was whether or not to have a newsletter. Mailing was neccessary to give members notice of meeting dates, venue and subject, so it was decided to add news items as they became available. It would also have the plan of a permanent meeting location and a list of officials. The newsletter has attracted a great many members well outside the South East region.

At about this stage it was decided that members so enjoyed the opportunity of having informal discussions during the 30 minute coffee break that they would probably like a second meeting each month in which to bring their own machines and talk about computing. The meetings then settled to the third and fourth Thursday of each month.

The club purchased a second hand ILER moniter which made demonstrations easier. It also took out insurance for members equipment to the value of 5,000 pounds (now 10,000 pounds).

New Developments.

During 1981, the club went from strength to strength. One of the business members of the club, who was so pleased with the benefits he had obtained through his membership of IPUG South East, gave the club 100 pounds to help finance future projects. These included the machine code course and the hardware add-ons project. Both these are run by Fred Offler on a correspondence course basis, for the price of one pound plus six self-addressed and stamped envelopes.

Kevin Viney and Tom Cranstoun then produced an excellent pair of utility chips (BASMON and PLUSDOS) for twenty pounds each, and these subsequently proved to be very popular retail items, thus helping the club even more. By May membership was up to 70 fully paid members, and they manned a stand alongside the national group at the PET Show in June of last year. Jim Butterfield, a complimentary member, came down to a club night during the Show, and this (not suprisingly) attracted a large number of people.

Membership was now growing rapidly, and by November they had 105 people enrolled (and around seven hundred pounds in the bank). Towards the end of 1981 Simon Tranmer produced his superb word processing package, known as Superscript. Naturally enough, IPUG South East think this is better than the other two main word processors: our view is shown in the New Product News section! What we can't argue about is the price: at thirty pounds it's about a tenth the price of those other two. Simon is adamant that he does not wish to market this as a private commercial venture, and is being outstandingly generous in that half of the thirty pound cost (thirty five to non-members) should go to IPUG South East.

This has already allowed the group to buy a set of 8050 disk drives for use at Club nights, for Superscript production, and for Simon's development work. The group now also has its own compiler available for compiling any members private programs.

New Committee Members

They've recently taken stands at the North London Polytechnic London Computer Fair and the Earl's Court Computer Fair, and are appearing in the future at the Commodore Computer Show and the PCW Show. They've given the first showing of the 40 column 64K VIC at a club night on the 29th of April, and would like me to pass on their thanks to Commodore for that opportunity. The future can only see them grow and grow.

Well, that's how one group have done it. How about you?


Now you can do all
accounting with...
without...

the filing, typing and

Silicon Office is the latest microcomputer software program from the Bristol Software Factory.

Designed specifically for use with the Commodore PET 8096, it'll help you run your office with the minimum amount of effort and maximum efficiency.

Think of it like three normal software packages in one, each separate package totally interactive with the other.

For around £4,500, you can have the complete electronic office, the solution to practically all your business problems. The price includes Commodore hardware, a high quality daisy wheel printer and Silicon Office software.

Silicon Office is made up from a flexible information management system which lets you create and maintain an extensive filing arrangement. Allowing you to search quickly through your records, making cross references between files in order to gain the facts you require.

A highly sophisticated word processing program allows you to generate letters, documents and reports. Letting secretaries get on with the more important tasks.

And a fully comprehensive calculator means you can handle all the number crunching you're ever likely to do in a business situation. Leaving the accounts department to concentrate on more profitable things.

But that's not all by any means. Silicon Office also has a special programmability feature which means you or your dealer can expand and tailor the Silicon Office program to your business.

When Silicon Office is used in an everyday business situation, certain command sequences are inevitably repeated. By writing short, very simple programs which are entered into the computer's memory, Silicon Office can perform the necessary tasks, automatically.

And last, but by no means least is an optional communications facility.

It doesn't take much imagination to see the potential of Silicon Office in virtually any line of business.

So to get a better grasp, send away for our brochure. It'll only cost you a stamp. And it could save you a fortune. Or talk to your local Commodore dealer who has all the facts at his fingertips.

You'll soon see how you're much better off with Silicon Office. Than without.

commodore
COMPUTER

I can't wait to get my hands on a free copy of the Silicon Office brochure.

Name______________________________

Position______________________________

Company______________________________

Address______________________________

______________________________ Telephone_______________

I own a Commodore PET (Please tick box) YES ☐ NO ☐

BSF Send to: Bristol Software Factory, PO Box 14, Horley, Surrey.

SILICON OFFICE

Telecomputing

TELECOMPUTING QUESTIONNAIRE

Modem

(a) What facilities do you require for a modem?

1. 300 baud full duplex
2. 75/1200/1200 baud full and half duplex
3. Auto – dial
4. Auto – answer
5. Acoustic
6. Direct connect

(b) How much are you willing to spend on a modem?

1. Up to £ 50
2. Up to £ 75
3. Up to £150
4. More than £150

Service

Which of the following proposed services would be of most interest to you? (Place a number 1 —17 to indicate order of importance, ie. 1 = highest.)

(a) Technical support information for:

1. Commodore hardware
2. Commodore software
3, OEM hardware
4. OEM software

(b) Sales information for:

Commodore hardware
Commodore software
OEM hardware
OEM software

(c) Electronic point-to-point and point-to-multipoint mail.

(d) Software distribution:

1. Public domain
2. Commodore
3. OEM
4. Uploading user – originated software

(e) Selling technical information.

(f) Selling secondhand equipent.

(g) Computer dating.

(h) Other (please specify): ....

....

....

What kind of software interests you most?: Educational Business Home System

Tariff

The more cash flow an information utility system such as is being proposed by Commodore can sustain in its early years, the better its chance of satisfactory growth and development. Please indicate how you think it would be best to apportion charges for this service.

0% 5% 10% 15% 20% 50% 100% other

(a) Connect time

(b) Commodore commission on cash flow

(c) Processing time

(d) Storage time

(e) First time
(Cost of software to acess system as percentage of total expenditure on system for first year.)

You may have read in the computer press that we have proposed a storage charge in the region of 30p per kilobyte per month with a commission payable to Commodore for all cash transactions on the system, effectively this means zero percent connect time charges, zero percent first time charges, zero percent processing charges.

Traffic

How much do you think you would use such a system per week?

5 mins 10 mins 15 mins 30 mins 1 hr over 1hr

Remarks

Please make any comment you have in the space provided below:

....

....

....

Send your completed form to:

Jean Frost
'PETNET' Project
CBM
675 Ajax Avenue
Slough, Berks SL1 4BG

With your name, address and occupation.
Best entries will receive complete collection of public domain software from Software Workshop.

FORTH Part Two

In the first article I gave a very brief introduction to some of FORTH's features: here I am going to run through a program which will read the directory from drive 0 and display it on the computer's screen.

Since FORTH is a compiler it is not possible to have only one piece of code which both the computer and the user can work with, as with BASIC. The source code, (that which the user sees), and the compiled code (which the computer uses) have to be kept separate.

When writing a FORTH program an editor is used. The editor works with a screen of 64 characters by 16 lines, on the VIC a horizontally scrolling screen makes writing programs far easier than using the standard VIC editor. Any screen from 1 to 255 can be edited, and when a screen is not in use it is compressed to reduce the use of RAM. Commands are built into the editor to move lines around the editing screen and even between screens. There are other commands to delete or copy whole screens, and because FORTH is easy to extend other commands can be added when necessary. For example a command called DUMP was added to print out a program listing.

Reading the disc directory

The Commodore disc system keeps its directory in the form of a program so that when the command: LOAD ''$0'',8 is given, the directory is read into the computer's RAM. When the command LIST is given it is displayed like a normal program.

So, if a program is going to be written which will read the directory directly, the manner in which programs are saved to disk must be known.

The first two bytes of a program file give the address where the program will be put into RAM. Then comes the program itself where each line has a standard format.
Two bytes for the link address to the next line.
Two bytes for the line number.
The line itself which is terminated by a 0. The end of the program can be found either by noting the change in the status, the method used here, or by checking for three consecutive zero's.

FORTH reads the directory.

Given the information above it is quite a simple task to translate it into FORTH. FORTH is a top-down language, in that you start with the main aim of the program and then flesh it out with the details. This allows very clear trains of thought to be followed through to conclusion before starting on another.

When the program is compiled by giving the command LOAD each of the commands starting with a colon will be compiled into the directory, (nine in all). Only the command $o is of real value to the user, the other eight are just flesh around the bones, but are given logical names to help with readability.

Here is how $0 works:-

OPENDIR as the name suggests opens the directory. The file is first CLOSEd to avoid 'file open error'. To open the file five parameters are required. These are the logical file number (1), the device (8), the secondary address (O), the address of the file name and the number of characters it contains (DRIVEO). The word DRIVEO leaves the address of the text ''$0'' and the number of characters it contains (2). The file is then OPENed. The IEEE bus is set to send characters from the disc by telling it that logical file 1 is for input (HPIN)

DROPTWO. This command will read two bytes from the disc and then drop them from the stack. The command 0 KEY accepts any value between 0-255 to the stack (the other command KEY will wait until a non-zero value is given). DROP simply removes the top item from the stack.

NUMBERS called DROPTWO to remove the link address and then reads in the line number and converts the two bytes into a 16 bit number (O KEY reads in the low and hi-bytes respectively, by multiplying the hi-byte by 256 and adding the lo-byte the conversion is made).

TITLE. The title line of the directory is special in that the line number is not used so DROPTWO is called twice to remove both the link and the line number. 3 SPACES simply pad the disc title to match the rest of the directory. CR forces a carriage return.

READ introduces some structured programming. FORTH will repeat the operations between BEGIN and UNTIL as long as the value on the top of the stack at the end of the loop remains false. O KEY EMIT will print out each letter as it is received from the disk. However we need to know if the value being taken from the disc is 0 to see if the end-of-line (EOL) has been reached.

The check is made by duplicating the value before it is printed ie. O KEY DUP EMIT the test 0= is then applied and leaves 1 if the test is true and 0 if the value on the stack is non-zero. STATUS leaves the current status of the IEEE on the top of the stack. This will usually be 0 indicating everything is ok: an error is non-zero. By ORing together the status and the EOL check a double test is being made ie. if either the status or the EOL have been reached then stop reading. UNTIL removes the top stack item while performing its test.

DIRECTORY is the routine which reads a full line of directory and prints it out. NUMBERS prints out the line number. A check on the STATUS is then made and only if this is zero (O=) will the number be printed (IF. THEN): the command '.' prints the top stack value. READ prints out the rest of the line and finally a check is made to see if the shift key is being pressed. The test BEGIN 152 C@ 0= UNTIL will loop continually if the shift key is down. On the VIC the address 152, which is 1 if the shift key is down, should be changed to 653. C@ (pronounced cee-fetch) replaces the address on the top of the stack with the byte of the address.

WAIT prints the message and will clear the screen when a key has been pressed.

$0 is the main calling word for reading the directory. The screen is cleared and the directory opened on drive 0. The start address is dropped and the disks title printed. The main body of the directory is then printed until either the status changes or the stop key (?TERMINAL leaves 1 if the stop key is pressed or 0 otherwise) is pressed. The IEEE is switched off, the directory closed and the message asking for a key to be pressed is printed, and when this is done the screen is cleared and control returns to the FORTH editor.

The program would be slightly more complex if the directory from either drive was to be read. The OPENDIR command would need a logical test to select either DRIVE or DRIVE 1 prior to opening the file. The main calling command $0 or $1 would have to leave a value on the stack before going to OPENDIR e.g,

: $0 0 DIR ; :$1 1 DIR ;

Where DIR is at present called $0.

The directory is read from the disk as fast as the inbuilt BASIC4 command CATALOG, thus showing the speed of FORTH. BASIC is noticeably slow at reading in the directory.

Software Review

ALIEN

Available from any Commodore VIC dealer, at 19.95 pounds.

Like many of the games cartridges available for the VIC, this one bears remarkable resemblance to an existing arcade game. It's a fairly accurate reproduction as well, making good use of the VIC programmable characters. One thing missing (lack of space?) was one of the more delightful features of the original game, namely that when you lose a life, an angel appeared from the top of the screen and carried you off. However, you can't have everything.

The game itself. This is activated simply by plugging the cartridge into the back of the VIC, and after an initial period of centering the image on the screen (why does this always appear to the case: you need to play with the cursor keys in order to get the picture centered up) you're ready to go. You control a little man around a maze, being chased by (at first) four alien beings. The idea is to kill the aliens off, and this is where the charm of the game becomes apparent.

Something different.

Rather than the usual space war scenario, where you're merrily blasting down spaceships raining down on you from above, this one is pleasantly different. To eliminate the aliens, you have to dig a hole in the floor of the maze, and wait until one of them blunders into it. Then you have to frantically fill the hole in again before the beast can clamber out. The keyboard takes quite a hammering at this point!

As well as having to avoid the monsters, and digging holes everywhere, you have another enemy to contend with: time. There is a three minute limit on your achieving success, namely killing all of them off. This is not too bad when there's only four of them, but once you've succesfully got rid of all of those, another six take their place to do battle once more. I've never got beyond this stage!

Scoring

Scoring is done, as you might imagine, by actually succeeding in burying an alien. Points seem to be given out on a random basis, but are usually around 2 or 3 hundred : the faster you catch them the more points you get. The description of the game given to you hints at a monster worth one thousand points, but he seems very shy and doesn't come out very often. You have three lives in total: they go very rapidly at first, but you gradually begin to get the hang of things, and games last a little longer.

One note of complaint is the manner of playing the game. It was designed for a right handed person (I'm not!), and needs four keys to move the man around the maze. A further two keys are necessary to dig the holes and fill them up again, which for a right handed person would be all right, but a left handed person could have problems, with fingers like wet spaghetti wandering over the keyboard attempting to find the correct keys in time. True, the program can be used with a joystick, making life a lot easier, but why should a left handed person have to pay out an extra 7.50 pounds?

Summary

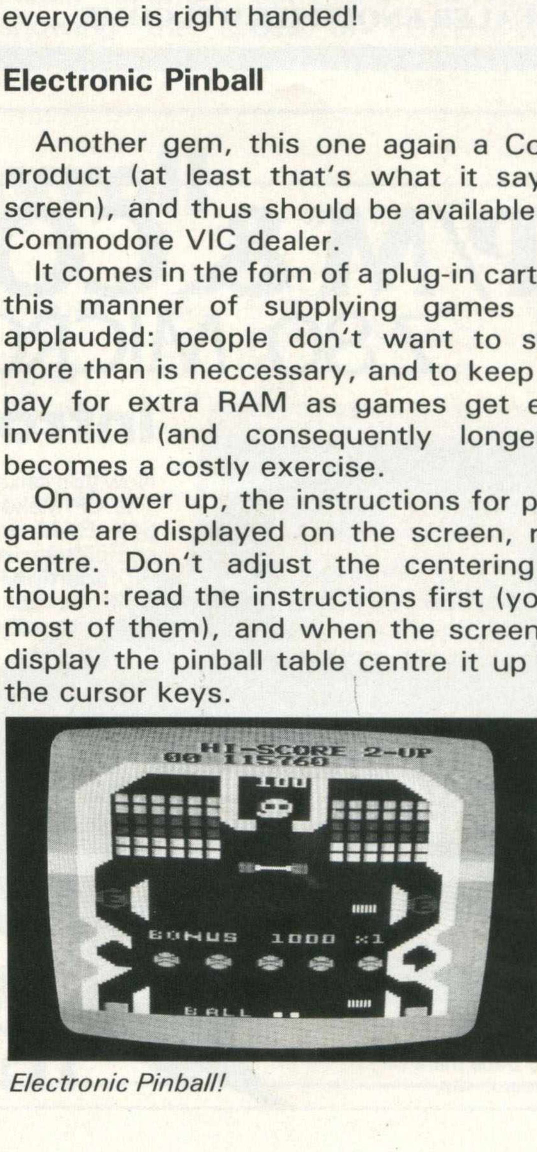
An original and interesting game, well packaged and presented. As stated earlier, however, programmers should realise that not everyone is right handed!

Electronic Pinball

Another gem, this one again a Commodore product (at least that's what it says on the screen), and thus should be available from any Commodore VIC dealer.

It comes in the form of a plug-in cartridge, and this manner of supplying games is to be applauded: people don't want to spend any more than is neccessary, and to keep having to pay for extra RAM as games get ever more inventive (and consequently longer) rapidly becomes a costly exercise.

On power up, the instructions for playing the game are displayed on the screen, rather off-centre. Don't adjust the centering just yet though: read the instructions first (you can see most of them), and when the screen clears to display the pinball table centre it up then with the cursor keys.



Electronic Pinball!

Pressing function key 1 sets the ball rolling, and then you press function key 1 or 3 depending on whether you want a one or two player game respectively. You will need to go into some additional expenditure to play this game however: it relies on the use of the VIC paddle to play the game. But at 13.50 pounds this is not too bad. It also makes the game significately easier to play, as it means you don't have to sit poised over the keyboard all the time. You can go into relaxed mode whilst attempting to rattle up the highest score.

Use of graphics

The game itself is very good, and the use of the VICs high resolution graphics, colour and sound is excellent. Once the ball is released, you have two blocks with which to stop the ball disappearing down the centre of the 'table'. At the top of the screen is a wall of small coloured blocks, which you have to knock out in 'breakout' fashion. If you manage to do this, a strange alien being appears in their place, and by killing this being (just by hitting it) a random bonus score is achieved.

In the top centre of the screen various other aliens appear from time to time, and again bonus points are awarded for hitting those. One of the very nice features of this game is a row of 'faces' towards the bottom of the screen. As the ball passes over them they turn from a frown to a smile, and making all five of them beam happliy away gives you yet more bonus points. This is not as easy as it sounds, because if the ball passes over a particular face again, it reverts from a smile to a frown, so to get all five smiling at the same time is rather difficult.

A bonus ball is achieved if you reach a score of 50,000, which is not often done. The highest score recorded at the recent Hanover Fair, for instance, was just over 35,000 (I think), by well known graphics designer and clairvoyant extraordinaire Myles Hewitt.

Summary

A very good, and very addictive, game, making full use of the programming facilities available on the VIC. Seeing games like this, and Alien as well for that matter, make you wish that more writers of games software for the VIC would adhere to this high standard.

Hardware Review

VIC 10

The rationale behind the renaming of this particular machine is somewhat unclear. The Ultimax was a good name, suggesting as it did 'The Ultimate'. Still, the VIC is the name that Commodore have established in the minds of the public, my only concern is the inevitable confusion that will arise, since the 2016 has been similarly renamed the VIC 30. Incidentally, the 2064 has now become known as the Commodore 64, just to add to the fun.

A first look at the VIC 10 is somewhat unimpressive. The now familiar PET and VIC keyboards have been replaced by a membrane keyboard, and the machine has an overall 'plastic' appearance. All the usual keys are still there, along with the famous PET graphics characters, although the function keys of the VIC 20 have been removed. It is designed to connect into a standard television, and if you're intending to use it with a monitor make sure it has sound : more of that later.

Commodore are stating a delivery date in the U.K of September this year, and a selling price of around 100 pounds (including V.A.T.). Let us hope that these can be stuck to, or even improved, especially in light of Clive Sinclair's new machine, the Spectrum. Their track record is not encouraging in this area, but perhaps this time they will be spurred into action!

Specification

The VIC 10 has a very impressive specification. Some raw details first of all: high resolution colour graphics, with a full resolution of 320 × 200 pixels. It has a 40 column by 25 row screen (Teletext!), a full keyboard music synthesis with three voices, programmable tones, wave shapes, filters and a noise generator. There is the ability to connect it up to plug-in cartridges, joysticks and a light pen, and there is a cassette port on board, thus destroying the earlier rumour that it couldn't be interfaced with a cassette deck. On the minus side, it only has 2.5K of User RAM available, and a mini-version of PET Basic (no dimension statements, no trigonometric functions, for instance). Looking at the technical overview there appears to be no facility for linking it up to printers or disk drives, but for the game end of the market, where most of the programs will be coming on cartridges, that is perhaps not such a drawback.

The high resolution graphics are very good indeed. Everything revolves around characters known as 'Sprites', each of which measures 24 × 21 pixels. On any given horizontal line there can be up to 8 of these sprites, but careful use of the interrupt capabilities of the video controller chip allows up to 256 of these to be displayed simultaneously on the screen. One can have multi-colour background and multi-colour sprites (all of this controlled from Basic I might add). The commands governing movement of sprites are similarly impressive: positioning is done just by specifying an X—Y register, and there are also routines for expanding sprites, collision detection, and so on. This should mean that we'll be seeing some very good arcade games for this machine shortly after it's appearance (or, if Commodore have any sense, AT it's appearance).

In terms of colour, 16 different colours can be displayed on the screen at any one time. At full resolution, 2 colours can be displayed per 8 × 8 pixel area, and at half resolution (160 × 200) 4 colours can be displayed per 8 × 4 pixel area. Facilities also exist for user programmable characters.

Sound

Also on board the VIC 10 is something called SID (Sound Interface Device?), which is the sound synthesiser chip. Now this is something of a revolution. You have control of 3 independent voices, each of which has the following capabilities : —1) a nine octave range from 0.059 Hz to 3.9 Hz, in steps of 0.059 Hz, 2) four waveforms (triangle, sawtooth, variable pulse and noise), 3) amplitude modulation, 4) ring modulation, 5) programmable addressable envelope generator, and 6) oscillator synchronisation. There is a programmable filter, independently selectable for each voice, and finally a master volume control. The result of all this is that you can achieve some staggering musical results, with remarkable ease. Certainly no other 'all-purpose' micro has attempted anything like this, on this sort of scale, before. Clive Sinclair's BEEP comand pales into insignificance.

Summary

If, and it's a big if, this machine comes out on time, to the specifications given, and if it comes out at the right price, Commodore are onto a winner. Despite incompatability with the VIC, the market is a completely different one, so that doesn't present any major problems: it's a very different area they're going into with this one.

Timing really is crucial though: with Sinclair reputedly well into production on the Spectrum, the VIC 10 needs to put in an appearance very quickly. Perhaps The PET Show will give us the answer.

Commodore 64

This is a larger version of the VIC 10, having the same sound and graphics capabilities, but with a lot more besides.

Pricing seems to be in a state of flux at the moment, but the initial suggestion was somewhere around the 500 pounds mark. This will have to fall in light of new developments. Their promised delivery date is January 1983, but again this will have to come forward quite a lot.

The machine has 64K of RAM on board, of which 40K is available directly to the user in Basic, the rest being accessed by machine code. Basic is contained in 20K of ROM, rather than in the form of a plug-in cartridge as with the VIC 10. It supports all of the VIC 20 peripherals, such as disk drives, printers etc., but items like the Beeline become redundant, as it has a 40 × 25 screen capacity built in. Compatible with Ultimax cartridges, so you games freaks can still relax, it will also support two joysticks or four paddles.

Memory mapping is quite ingenious, as you can redefine the memory map for more convenient input/output, or video location. I can't think of another machine where this is possible (which no doubt means hundreds of you will write in and tell me of one!). There is also an EPROM on board: a nice point.

Apart from that, the Commodore 64 is virtually the same as the VIC 10. One major difference is the keyboard: it has the same keyboard as the 'new' VIC 20s, which allows for touch typing.

Summary

The same comments apply to this as apply to the VIC 10. If it comes out at the right time, and at the right price, it has every prospect of being a very big seller indeed. But Commodore need to move fast: the market is beginning to catch up with them rapidly!

The New Boys

Book Review

Programming the PET/CBM

Since the original 8K PET first made its appearance back in 1978, there have been a vast number of books published about it. This is due principally to the at times truly appalling standard of Commodore's own documentation, and people were quick to realise that there were many gaps to fill.

It has to be admitted that a number of those books were themselves not terribly impressive. One had to filter through a whole array of publications before finding one that was worthwhile looking at, let along buying. Even today, that regrettable situation remains with us.

Consequently, it is a pleasure to be asked to review such a book as Ray West's "Programming the PET/CBM". Some of the books that get sent to me are only useful for stopping unstable desks from falling over. This one, as they say, is something completely different!

Technical details first of all. The book, in paperback form only as far as I know, sells at 14.90 per copy (including post and packaging), or 13.99 pounds inclusive for quantities of five or more. This is available from Level Limited, PO Box 438, Hampstead, London NW3 1BH. It covers an impressive 503 pages, and is broken down into 17 main chapters, plus a host of appendices and an excellent index and glossary.

Introduction

The first chapter is basically an introduction to the book itself, and it would be instructive here to reproduce part of that introduction: it gives as good a clue as any to the reason for producing it, and (inherently) for buying it. Mr. West writes:-

"The purposes of this book are to teach competent programming and provide a comprehensive reference text on the PET/CBM range of microcomputers".

Essentially the book makes the transition from BASIC to machine code, stopping off along the way to pay tribute to the more popular Commodore peripherals: the disk drives and the printers. Newer items like the VIC, MMF (or SuperPet, or whatever it's known as these days), and the Commodore modem, are not really mentioned in any great detail. The author took the quite valid theory that these were not in the mainstream of use, and that most users would have mainly a PET, and/or a disk drive and printer. I think that this is indeed the case.

One of the features of the book, and this is hinted at in the general introduction, is the number of extremely useful listings. For example, one that is actually mentioned in the introduction, is a program to convert the PET cursor control characters into words like HOME, DOWN, RVS, and so one. Most of these programs are quite short, thus the likeliehood of mis-entry from the keyboard is reduced, and are usually in machine code when speed is of the essence. The explanations for the programs are to be commended.

The first three chapters of the book concern themselves with an introduction to programming the PET in Basic. Owing to the amount of material that this book is covering, the explanations of how Basic and the PET work are necessarily brief, but perfectly adequate for even the beginner to programming. There is a chapter on program and system design, which is the element of programming that a lot of people seem to miss out on. The fourth chapter is a short guide to effective programming in Basic, and how to get the best out of the programs that you develop yourself.

Basic Commands

It is in the fifth chapter however that the book comes into it's own, with a section which as far as I know has never been printed before. 86 Basic commands are explained in great detail, with at least a page on each one. Nothing special about that? Not only does Raeto West take all the PET Basic commands, but also other Basic commands that are not implemented on the PET, together with a description of how to go about getting them on your system.

For the existing commands, there is an explanation of how they work, together with examples of their use, and short (but concise and perfectly understandable) notes on how to effectively use them when programming. This is followed by the rom entry points for those commands, thankfully given for each version of Basic that has so far appeared.

The list of new commands presented here goes a long way to making add-ons such as toolkits, basic aids etc. redundant, as almost all of the commands that they give you are also contained in this book. Of course, there is the inconvenience of having to enter them in the first place, but once they're there, and you have of course the ability to blow your own roms ....

An introduction to just some of the commands should give you an idea of the extra power that you can give your system. For instance, we have the now standard merge, dump, renumber and so on, but there are also newcomers such as OLD (recovers a program accidentally deleted by NEW), UNLIST (to prevent listing of programs), SORT (and a variety are presented here), and many more.

Peripherals

The book then moves onto three chapters about peripherals, and in particular the various disk drives that Commodore have produced over the years. Examples of sequential and direct access files are given, together with an introduction to machine code programming on disk drives. The Computhink unit is also given a mention.

From there we have a brief section covering such items as printers, cassette decks, modems and other anciliary hardware. These are dealt with fairly briefly, but then as we stated earlier this book is aimed at the mainstream user.

The final chapter to concern itself mainly with Basic is devoted to the use of graphics and sound, and contains quite a number of detailed program listings and examples of how to perform animation and so on. He also takes a quick look at pen plotters.

And so onto machine code, where the format is roughly the same as his introduction to Basic. Sections on programming in machine code, and how to make the transition from Basic to 6502 language are followed by a chapter on effective 6502 programming, in much the same way as West earlier looked at effective Basic programming. Similarly we then have at least a page devoted to each of the 6502 opcodes (all existing ones this time!), followed by a description of how to modify the inherent rom routines.

The final part of the book is the usual selection of memory maps, a listing of Supermon for all versions of Basic, to conclude an excellent book.

Summary

Possibly the best book available for the Commodore range of computers at present. My one complaint is that the text is not spaced out enough, and that sometimes one wishes that he'd supplied a magnifying glass to go with the book. However, this must not detract from what is, overall, a most useful addition to anyone's computing library.

Guest Expert

F. J. Townsend looks at using an 8050 Commodore Disk Drive on a Old ROM PET

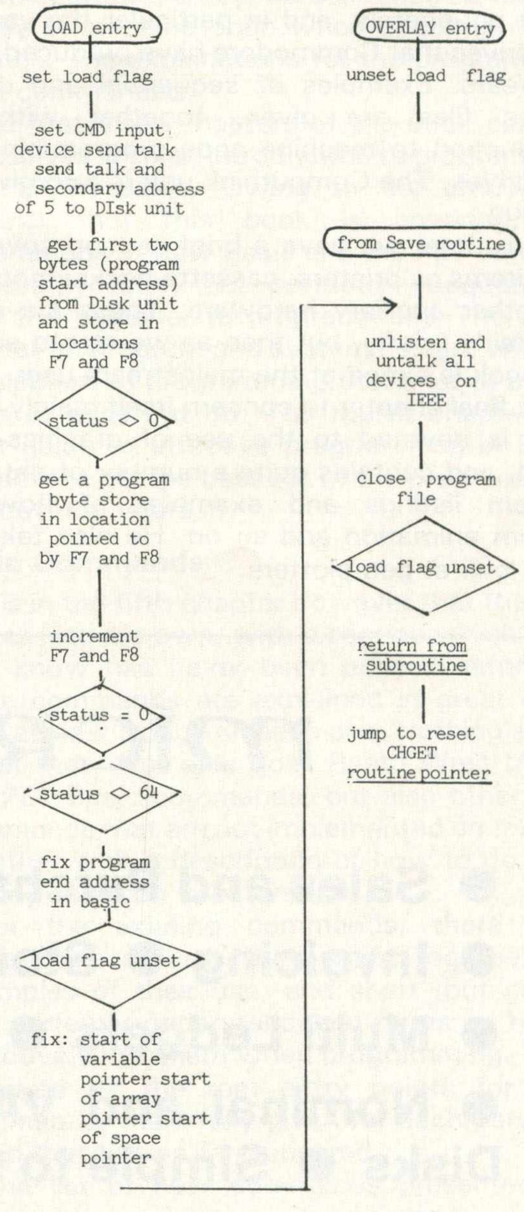
No doubt many owners of the original 2001 series PETs have hankered after the addition of a Disk unit to their system. In the past one way was to fit a set of later ROMs, but this is only easy if the version of the PET uses the 24 pin IC ROMs. The author was in the unfortunate position of having 28 pin IC ICs so this course was abandoned. The alternative was to examine the possibility of providing a machine code routine to interface to a Disk unit.

The main problem in driving a Disk unit with the Old ROM Basic is that the functions of Load, Save and Verify will not work with Disk. In their original form these functions were designed with the Cassette Tape only in view. As the method used for storing programs on Tape is different to Disk, later ROMs contained considerable changes in these routines. In the area of data handling the old ROMs are able to Open, Input, Get, Print and Close Disk files. This now narrowed the missing requirements to Loading and Saving programs only, as it was decided that the Verify function was not absolutely necessary.

On the basis of this, the author took the plunge and purchased a 8050 Disk unit. The choice of a 8050 was made so that all the latest DOS 2.5 facilities would be available as well as the increased storage capacity. The main point abut the Commodore Disk unit is that programs are stored in the same format as data files, the main difference being that the file type is 'Program'. As the Old ROM PET can read Sequential files there was no reason why it could not read Program files so all that was needed was a routine able to open a program file, read it into the RAM store, adjust the various Basic pointers and then return to Basic. With the above in mind and a further constraint that the routine to be produced must fit in a Cassette Buffer, the following program was written.

The Disk Load Routine

Figure 1 shows the flowchart of the program load routine. Two points will be noticed, these being that there is no mention of opening the program file and there are two entry points. In the first case opening files at machine code level is complicated and can consume considerable code so it was decided to open the required program file as a direct Basic command prior to entering the program load routine. This is simply done by typing:— OPEN 5,8,0,''program-name,P,R''. The two entry points allow for either just loading a program and then returning to Basic command level or loading and immediately running a program in 'Overlay Mode' retaining the data area from the previous program.



The Disk Save Routine

Figure 2 shows the flowchart of the program save routine. As in the load routine the program file opening is done prior to entering the program save routine. This is done by typing:— OPEN 5, 8, 1, ''program-name,P,W'' The jump to the load routine on completion of save is purely to reduce code by using a common terminating routine.

SUPERSCRIPT WORD PROCESSOR

It is the opinion of very many discerning CBM users that SUPERSCRIPT is superior to either of the other two serious contenders for Commodore honours, namely 'Wordpro' and 'Wordcraft'. SUPERSCRIPT supports extremely full type, edit, print and store commands, using any combination of machine or disk system. It will even read files already created by either of its rivals mentioned above!

The superb SUPERSCRIPT specification includes:-

* Use of screen as a 'window' on text with full scrolling in **all** directions.
* Enough memory space for 20,000 text characters at a time (equivalent to 250 lines, 80 cols. wide)!
* Text widths from screen size up to max of 240 characters.
* Character string search or replace in local or global mode.
* Block transfer or append function.
* Auto or manual variable block fill from disk files (ASCII or SUPERSCRIPT).
* Foolproof spooling modes - single page or continuous (disk to printer whilst computer freed for other tasks).
* Output options include video preview, continuous or single sheet, auto fill, global and multi-copy.
* Full control of both margins **and** lines per page, auto line feed, line spacing, no. of text lines per page, forced paging, etc.
* Pause in printing command (with message if desired), also ability to insert unprinted comments into text.
* Printer routines, where applicable, provide automatic underlining, enhancement, bold print, superscripts, subscripts, ribbon colour change, variable line and character pitch, etc.
* Horizontal **and** vertical tab settings (saved on disk if req.).
* Powerful and comprehensive insert, transpose and delete modes.
* Erase all, remainder, paragraph or sentence.
* Justification (with equal white-spacing on letter-quality printers).
* Centering and right alignment, even with mix of enlarged and normal characters.
* Secondary address o/p of data (for control of interfaces etc.).
* Headers and trailers available on every page, including auto page numbering if required.
* Any special printer functions fully usable including; self defined characters, backspace, escape codes, proportional printing, etc.

SUPERSCRIPT supports MX80, CBM, Spinwriter, Qume and Diablo printers. There are facilities for saving printer files to disk and specifing the use of ASCII format. The latter allows full access by other protocols. ASCII files can also be loaded directly into SUPERSCRIPT, eg. as sequential files. Even BASIC programs can be loaded and manipulated if they are first saved on disc. All standard disk operations are available including directory viewing which can be accomplished without the need to exit from the program and text is not at risk.

SUPERSCRIPT is available only through ICPUG SE
(Specify 2/3/4040 or 8050 format).

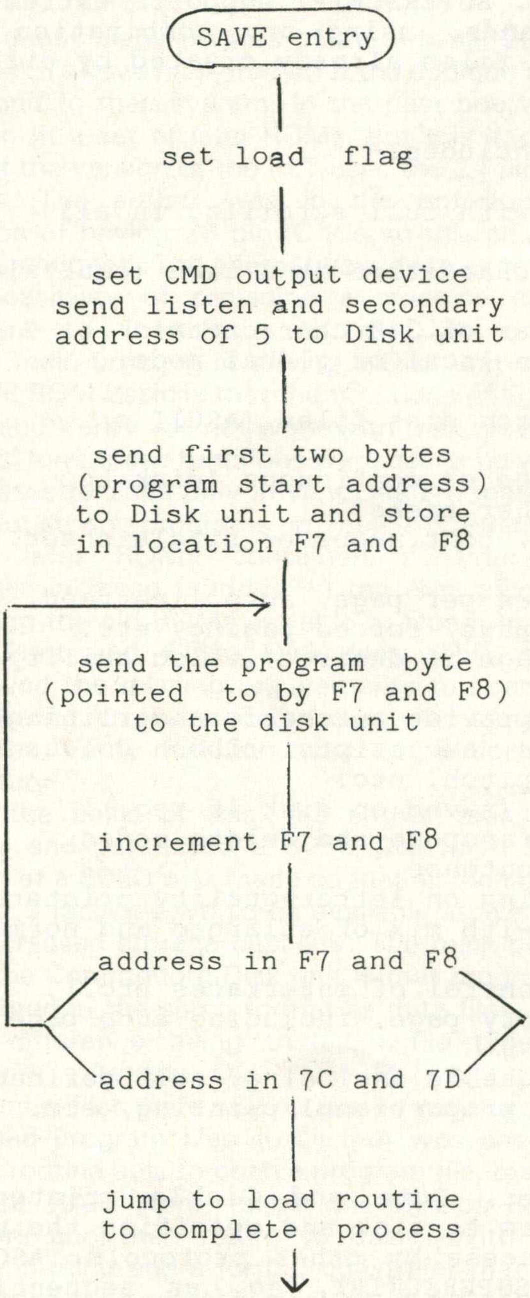
Price £30.00 - all ICPUG members (£35.00 - non ICPUG members)
Includes disk and comprehensive instruction folder.

Orders enclosing a cheque/postal order (made payable to ICPUG SE) to:-

Tom Cranstoun, Flat 7, 10 Lancaster Road, London SE25 4AQ.
Telephone (01) 771 3525.

Guest Expert

Figure 2 - The Save Routine



The Basic Disk Load/Save Program

The Basic Disk Load/Save program is shown in figure 3 and is coded to reside and be used from Cassette Buffer 2. If the user wishes to use Buffer 1 then the changes shown at the begining of the program must be implemented. Naturally after the changes are done the resultant Basic Disk Load/Save program will, when run, locate the program in the appropriate buffer. For those interested, a disassembled print of the program is shown in figure 4.

BASIC DISK LOAD/SAVE PROGRAM

```
60 REM   CHANGES FOR BUFFERS NO 1 OR NO 2
61 REM
62 REM   BASIC    VALUES FOR BUFFERS
63 REM   LINES      NO 1      NO 2
64 REM
65 REM   110        223,2     159,3
66 REM   150        223,2     159,3
67 REM   165        223,2     159,3
68 REM   173        223,2     159,3
69 REM
70 REM   ALSO FOR BUFFER NO 1
71 REM   CHANGES LINE 100 TO :=
72 REM
73 REM   FOR I = 634 TO 784
74 REM
100 FORI=826TO976
102 READDT:POKEI,DT:NEXT:PRINTCHR$(147):NEW
105 DATA 169,1,208,2,169,0,141
110 DATA 159,3
115 DATA 162,5,32,139,247,32,204
120 DATA 241,133,247,173,12,2,208,59,32,204
 ,241,133,248,173,12,2
130 DATA 208,49,160,0,32,204,241,145,247,230
 ,247,208,2,230,248,173
140 DATA 12,2,240,240,201,64,208,27,165,247,
 133,229,165,248,133,230,172
150 DATA 159,3
155 DATA 240,14,133,125,133,127,133,129,165,
 247,133,124,133
160 DATA 126,133,128,32,125,242,169,5,32,205,242,172
165 DATA 159,3
170 DATA 240,1,96,76,154,197,234,1,162,5,142
173 DATA 159,3
175 DATA 32,220,247,165,122
180 DATA 133,247,32,48,242,165,123,133,248,32,
 48,242,160,0,177,247
190 DATA 32,48,242,230,247,208,2,230,248,165,
 248,197,125,208,239,165
200 DATA 247,197,124,208,233,240,188
```

ADDRESS DEC	HEX	MACHINE CODE	ASSEMBLER CODE	
826	33A	A9 01	LDA	#$01
828	33C	D0 02	BNE	$340
830	33E	A9 00	LDA	#$00
832	340	8D 9F 03	STA	$039F
835	343	A2 05	LDX	#$05
837	345	20 8B F7	JSR	$F78B
840	348	20 CC F1	JSR	$F1CC
843	34B	85 F7	STA	$F7
845	34D	AD 0C 02	LDA	$020C
848	350	D0 3B	BNE	$38D
850	352	20 CC F1	JSR	$F1CC
853	355	85 F8	STA	$F8
855	357	AD 0C 02	LDA	$020C
858	35A	D0 31	BNE	$38D
860	35C	A0 00	LDY	#$00
862	35E	20 CC F1	JSR	$F1CC
865	361	91 F7	STA	($F7),Y
867	363	E6 F7	INC	$F7
869	365	D0 02	BNE	$369
871	367	E6 F8	INC	$F8
873	369	AD 0C 02	LDA	$020C
876	36C	F0 F0	BEQ	$35E
878	36E	C9 40	CMP	#$40
880	370	D0 1B	BNE	$38D
882	372	A5 F7	LDA	$F7

```
884  374  85 E5     STA $E5
886  376  A5 F8     LDA $F8
888  378  85 E6     STA $E6
890  37A  AC 9F 03  LDY $039F
893  37D  F0 0E     BEQ $38D
895  37F  85 7D     STA $7D
897  381  85 7F     STA $7F
899  383  85 81     STA $81
901  385  A5 F7     LDA $F7
903  387  85 7C     STA $7C
905  389  85 7E     STA $7E
907  38B  85 80     STA $80
909  38D  20 7D F2  JSR $F27D
912  390  A9 05     LDA #$05
914  392  20 CD F2  JSR $F2CD
917  395  AC 9F 03  LDY $039F
920  398  F0 01     BEQ $39B
922  39A  60        RTS
923  39B  4C 9A C5  JMP $C59A
926  39E  EA        NOP
927  39F  EA        NOP
928  3A0  A2 05     LDX #$05
930  3A2  8E 9F 03  STX $039F
933  3A5  20 DC F7  JSR $F7DC
936  3A8  A5 7A     LDA $7A
938  3AA  85 F7     STA $F7
940  3AC  20 30 F2  JSR $F230
943  3AF  A5 7B     LDA $7B
945  3B1  85 F8     STA $F8
947  3B3  20 30 F2  JSR $F230
950  3B6  A0 00     LDY #$00
952  3B8  B1 F7     LDA ($F7),Y
954  3BA  20 30 F2  JSR $F230
957  3BD  E6 F7     INC $F7
959  3BF  D0 02     BNE $3C3
961  3C1  E6 F8     INC $F8
963  3C3  A5 F8     LDA $F8
965  3C5  C5 7D     CMP $7D
967  3C7  D0 EF     BNE $3B8
969  3C9  A5 F7     LDA $F7
971  3CB  C5 7C     CMP $7C
973  3CD  D0 E9     BNE $3B8
975  3CF  F0 BC     BEQ $38D
```

Using the Disk Load/Save Program

When the PET is first switched on the Basic Disk Load/Save Program is read in from cassette and run. From now on the program will remain in the appropriate buffer until the PET is switched off or the buffer is overwritten.

Load a Disk program from Basic command level

Type OPEN 5,8,0, ''program-name ,P,R''
SYS(826)

The machine should return with the flashing cursor prompt on completion of the load

Loading an over lay Disk program from within a program

Including the following code at the end of the calling program.

line-no OPEN 5,8,0,''overlay-name,P,R''
line-no REM INTERROGATE THE ERROR CHANNEL
line-no INPUT # 15,AA,BB$,CC,DD
line-no IF AA<>OTHEN PRINT AA;BB$;CC;DD: stop;end
line-no SYS(830):END

After loading the overlay program it will then be automatically entered.

Saving a program to Disk

Type OPEN 5,8,1,''program-name,P,W''
SYS(928)

the machine should return with the flashing cursor prompt on completion of the save

Note:— The secondary address is 1 for save and 0 for load

The file mode is W (write) for save and R (read) for load

The logical file number of 5 must only be used

The entry points shown in the load and save examples are for use when the Disk Load/Save program resides in Buffer 2. The following table gives the entry points when using Buffer 1.

Entry Points

Function	Buffer 2	Buffer 1
Normal load	826	634
Overlay load	830	638
Save	928	736

Possible causes of failure when using the Disk Load/Save Program

The Open command was not typed exactly as in the instructions.

The 'program-name' program does not exist when loading or already exists when saving, on the current disks mounted on the Disk drive. A file with the logical file number of 5 is already open. In this case close it with := CLOSE 5. Note: When using this system, logical file number 5 should be reserved for program loading and saving only.

If while loading a program the VERIFYING message appears on the screen, ignore it as it appears becuse the last cassette function was verify and in fact a load will be taking place.

Future Enhancements

Having produced a routine enabling a Disk unit to be used with the Old ROM PET, the next step is to be able to use the standard DOS Support program in conjunction with this routine. In a forthcoming article I shall explain how I produced an Old ROM version of the Commodore DOS Support Program.

Using the 8010 Modem

Since June last year, Mike Bolley and David Parkinson of Ariadne Software Ltd, have been working intermittently on communication software, principally using the 8010 modem. Their main project in this time has been PETNET, an error-protected message and program communication system for PETs using a mainframe host, which is currently being tested by Commodore. In this article, David Parkinson passes on various suggestions concerning the 8010, many of them quite polite.

The 8010 modem is a 300-baud, full duplex, somewhat-buffered acoustic coupler sitting on the IEEE bus, marketed by Commodore. Despite rumours of a 'modem bug' which were circulating some months ago, our experience is that usually these devices work very well (though they can be a bit picky about which other devices they choose to talk to). However, the modem is not nearly as easy to use as other IEEE devices like Commodore disk drives and printers, and unless the software is absolutely right, it is very easy to lose characters. In the first part of this article I will consider why modem software presents problems; in the second part I will suggest solutions to these problems.

What's the problem?

The PET version of the IEEE bus is designed for efficient communication of fairly intelligent peripheral units, with the PET as 'controller' taking overall responsibility for events on the bus. For example, if a disk drive has a file open to read, then it prepares to put the next character on the bus, and waits until the PET signals that it is ready to receive it. The disk drive will wait indefinitely for this - say until a long BASIC program reaches its next GET statement - so there are no timing problems to bother the programmer.

The problem with inputting characters from a modem is that in this case there is another source of timing involved - the computer at the other end. This computer (say a mainframe or another PET) is every bit as much in control as you are, and on a simple acoustic link it is not going to hang around between characters until you indicate that you are ready for the next one. Thus characters must be removed from the modem as they come in, or they will be lost. Timing therefore becomes very important. The problem becomes worse if you want to use other devices on the bus along with the 8010 - say input a file character by character from the modem and simultanously dump it to disc. Output of characters on the IEEE is designed to allow simultaneous output to devices of differing speed - say a disc drive and a printer. This is done by a clever handshake involving several lines, including NRFD (not ready for data). Devices on the bus indicate a readiness to receive data by letting this line go high; the Pet then puts a character on the bus, and the devices hold NRFD at earth potential while they digest it. Nothing else can happen on the bus until NRFD goes high again, and this cannot happen until the slowest device has finished and lets NRFD go. The bus is normally only disabled for a fraction of a second (even when a printer executes a carriage return), but this can be long enough to lose several characters coming in off the modem. Thus communication with other devices on the IEEE bus must be handled with extreme caution if there is an 8010 around.

These problems would be a lot less severe if the modem had intelligence comparable to the disc drive and printer, and a reasonably large amount of internal RAM for buffering purposes. The modem could then hang onto characters until the rest of the system had sorted itself out, and the PET was ready to receive them. Then, provided that the overall system speed was adequate, there would be very little problem. The 8010 however is a comparatively inexpensive device, and unfortunately it is also comparatively dumb. In particular it only has a two byte character buffer, which is better than nothing, but only two bytes better than nothing.

Some possible solutions

The above paragraphs may appear rather terrifying; however given an awareness of the nature of the problem, it is reasonably easy to find solutions. A number of approaches will be discussed, the primary object being not to present complete programs, but to provide pointers which may assist people in writing their own modem software.

The keep on looking approach

The simplest way to handle the modem is to get characters from it in such a tight loop that there is no chance of any characters being lost. When GET# is performed in BASIC or machine code, a byte will be returned if there is one available, with the status word ST set to zero.

Applications

Otherwise the system will time out after 65 milliseconds, and the status word set to 02. This can be used to write very simple terminal software, similar to the examples in the 8010 manual:

```
10 OPEN 5,5:PRINT"(clr)ON LINE"
20 PRINT "(LEFT ARROW,cl)";:GET A$:IF A$<>"" THEN PRINT 5,A$,
30 GET 5,A$:IF ST=0 THEN PRINT " (cl)";A$;
40 GOTO 20
```

This works in full or half-duplex operation. The tight loop is in lines 20 to 40: first we print a back arrow cursor; then we check for keyboard input and if found send it to the modem; then we check for modem input and if found print to the screen; then we go and do it again

The problem with this approach is that time considerations make it very difficult to do any other processing while this is going on. By a trick of Jim Butterfield's using two index arrays, it is possible to convert to and fro between PETSCII and ASCII for communication with other machines. This is about the best that can be done though; simultaneous output to a disc is not on, and any form of error checking by parity, check sums or (best) by cyclic redundancy checks (CRCs) is completely out of the question.

Quite a lot of the time problem is due to the speed of BASIC interpretation, so it is worth redoing the same routine in machine code. Probably the easiest and safest way of doing this is to use the jump table INCHR and OUTCH routines at $FFCF and $FFD2 (this isn't the fastest way, but the extra time taken by these routines is negligible in all normal circumstances). First one needs to open 5,5 to the modem; the main loop is then as follows:

LISTING 2

```
2000 MAIN   JSR CGETL           ;CHARACTER FROM KEYBOARD?
2010        BNE MAIN10
2020        PHA                 ;IF SO SAVE CHAR ON STACK
2030        LDX ##$05           ;    SET OUTPUT ON MODEM
2040        JSR COOUT
2050        PLA                 ;    PULL CHAR OFF STACK
2060        JSR OUTCH           ;    OUTPUT TO MODEM
2070        JSR CLSCHN          ;    RESTORE  NORMAL 1/0
2080 MAIN10 LDX  $05            ;NOW SET INPUT CHANNEL TO MODEM
2090        JSR COIN
2100        JSR INCHR           ;ATTEMPT TO HANDSHAKE CHAR IN
2110        LDX CSTAT           ;STATUS WORD ST=0 IF SUCCESSFUL
2120        BNE MAIN20
2130        JSR OUTCH           ;IF SO OUTPUT TO SCREEN
2140 MAIN20 JSR CLSCHN          ;NOW RESTORE NORMAL 1/0
2150        JMP MAIN            ;AND REPEAT FOR EVER.
```

This is a dismal program designed only to illustrate a possible technique, and it is not suggested that you try to run it. It has no cursor, no control key, and it's stuck in a closed loop with no escape! These problems could of course be overcome. The program suffers from a more fundamental fault however, in that it is still impossible to do very much processing (disk I/O, CRC error checking etc.) while still looking at the modem. The reason for this is the 65 millisecond time-out on handshaking characters in from the modem . This is not a long time by basic standards; in machine code however it is several eternities, and in this time the CPU is stuck in the ROM in a tight loop in the routine starting at $F1CO (BASIC 4), doing nothing of any use. A similar thing happens when outputting characters to a modem. At 300 baud, the interval between characters is about 30ms, so time-out should not occur; however the ROM routine starting at $F109 may have to wait doing nothing for up to this time before the modem lets NRFD go high. Thus although there is time to do a few other things (eg ASCII to PETSCII and vice versa), this approach is not extendable to very sophisticated modern processing.

We continue our explorations in next month's issue.

"Now let's see, for optimum results add 1.29987143 cups of flour."

Graphics

Graphics

As promised, this month we take a look at high resolution on the VIC. Some of the more common commercially available programs appear to be aids to designing your own high resolution characters, for use in games programs and the like. Here we present a listing for just such a program.

```
            HI-RES AID

1 REM********************
2 REM********************
3 REM****HI-RES AID****
5 REM**BY MARK BLOOM**
6 REM********************
7 REM********************
8 PRINT"⌐MOVING CHR GENERATOR"
9 GOSUB4010
10 PRINT:PRINT
15 REM**SET VARIABLES**
20 Q=160:S3=36876:POKES3+2,15
25 REM**PRINT OUT GRID**
30 PRINT"▚X":PRINT
40 PRINT"1 2 3 4 5 6 7 8"
50 FORI=1TO8
60 PRINT"┌┬┬┬┬┬┬┬┐"
70 PRINT"└┴┴┴┴┴┴┴┘"I
80 NEXTI
90 PRINT"⌐",TAB(19)"Y"
100 Z=7700
102 PRINT
105 REM**ENTER CO-ORDINATES**
110 INPUT"XY";A$
120 IFLEN(A$)<>2THEN230
125 REM**PICK OFF X,Y**
130 X=VAL(LEFT$(A$,1))
131 Y=VAL(RIGHT$(A$,1))
135 REM**CHECK TO SEE IF IN RANGE**
140 IFX>8ORY>8THEN230
145 REM**END OF DESIGN**
150 IFX=0ANDY=0THEN1000
155 REM**BLEEP ON**
160 POKE S3,200
165 REM**CALCULATE TOP LEFT HAND CORNER OF BLOCK**
170 P=Z+X*2+(Y*44)
175 REM**CHECK FOR DOUBLE ENTRY**
180 IFPEEK(P)=QTHEN230
185 REM**PRINT OUT THE BLOCKS**
190 POKEP,Q:POKEP+1,Q:POKEP+22,Q:POKEP+23,Q
195 REM**CALCULATE VALUE OF BLOCK**
200 U=ABS(X-8)
210 A(Y)=A(Y)+(2↑U)
215 REM**BLEEP OFF**
220 POKE S3,0
225 REM**MOVE BACK TO INPUT**
230 PRINT"⌐⌐":GOTO110
1000 PRINT"⌐"
2000 REM**STORE PATTERN'S VALUE IN NEW GENERATOR**
2010 POKES3,0
2100 FORI=1TO8
2300 POKE6143+I,A(I)
2400 NEXTI
2500 REM**CHANGE REGISTERS**
3000 POKE36869,253
3002 POKE36866,PEEK(36866)OR128
3005 REM**CHANGE SCREEN COLOUR TO BLACK**
3010 POKE36879,8
3045 PRINT"⌐"
3075 REM**PRINT HI-RES CHARACTER ON SCREEN**
3100 FORI=7702TO8185 STEP4
3110 POKEI,128
3120 NEXTI
3140 PRINT"▚▬     HIT ANY KEY     "
3200 GETA$:IFA$=""THEN3200
3250 REM**RESTORE REGISTERS**
3300 POKE36869,240
3350 POKE36866,150
3400 PRINT"⌐▬"
3500 POKE36879,27
3550 PRINT"THESE ARE THE VALUES"
3560 PRINT"▚TO ENTER AS DATA":PRINT:PRINT
3600 FORI=1TO8
3650 PRINTA(I)
3660 NEXTI
3700 END
4000 REM**MOVE CHARACTER GENERATOR**
4010 FORI=0TO1024
4020 POKE5120+I,PEEK(32768+I)
4030 NEXTI
4040 PRINT"⌐"
4050 RETURN
READY.
```

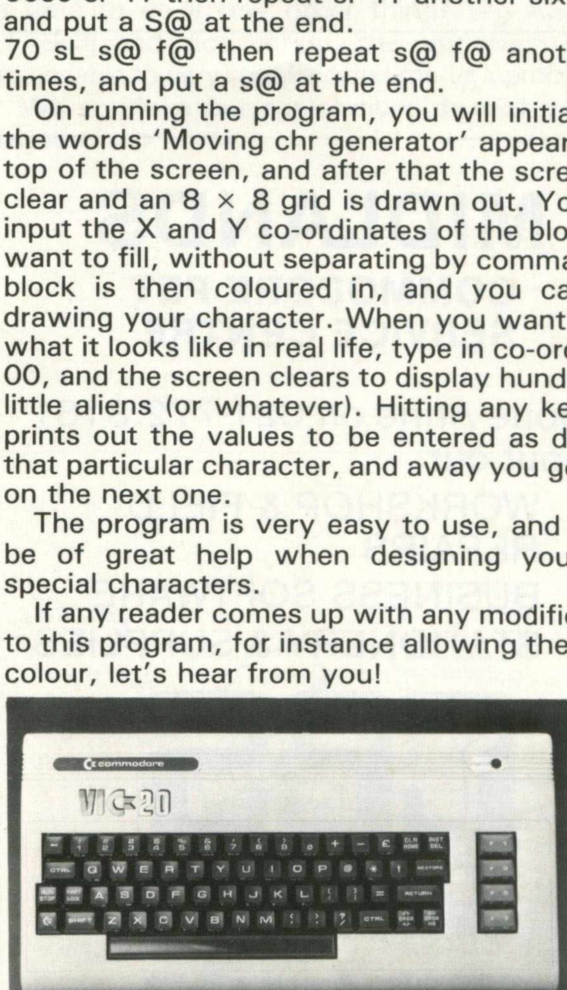
The program will fit into the standard VIC, provided you remove all the REM statements. It should be easy enough to follow without them, although of course you can always refer to this listing if necessary. Otherwise, you'll require a RAM expansion pack.

A few notes about the program itself. The graphics in line 60 are as follows (s indicates shift, and f the Commodore flag):—
60sO sP fT then repeat sP fT another six times, and put a S@ at the end.
70 sL s@ f@ then repeat s@ f@ another six times, and put a s@ at the end.

On running the program, you will initially see the words 'Moving chr generator' appear at the top of the screen, and after that the screen will clear and an 8 × 8 grid is drawn out. You then input the X and Y co-ordinates of the block you want to fill, without separating by commas. The block is then coloured in, and you carry on drawing your character. When you want to see what it looks like in real life, type in co-ordinates 00, and the screen clears to display hundreds of little aliens (or whatever). Hitting any key then prints out the values to be entered as data for that particular character, and away you go again on the next one.

The program is very easy to use, and should be of great help when designing your own special characters.

If any reader comes up with any modifications to this program, for instance allowing the use of colour, let's hear from you!



Our old friend the VIC

A high speed stepper motor for the PET

The combination of a computer and a stepper motor gives us one of the most precise methods of producing and controlling rotary and linear motion. Stepper motors come in all sizes, from the small motors used in calculator printers to multi-horsepower motors for use in computer-numerically-controlled(CNC) machine tools.

The ease with which a stepper motor can be controlled by a microprocessor based computer like the PET makes it an ideal subject for experimentation, especially for those interested in robotics.

A stepper motor is totally different in design from an ordinary electric motor, which is characterised by its continuous motion when energised. The shaft of a stepper motor, on the other hand, moves in small incremental angular steps on command and will maintain this position between commands.

To achieve this, the motor consists of a permanent magnet with alternate poles magnetised around its periphery. The stator consists of a set of coils which can be magnetised under control of external electronics. Motion is produced by the alternate attraction and repulsion of the permanent magnet and the coils of the rotor moving to the position of greatest magnetic attraction - a position which it will hold until the coil magnetisation is changed.

In a previous article we looked at some methods of controlling stepper motors. However, the results were severely limited in both power and speed, being confined to just a few tens of steps per second.

The reason for this is that the designs were based on simple drive electronics which utilised only a small part of the start/stop operating curves for a given motor. These curves give the motor torque for a given speed at which the motor may be stopped or started instanteously without loss of command position.

Cash Limits

Beyond this curve is another, termed the slew curve, which, depending on motor and drive, may extend to thousands of steps per second. This is possible by accelerating the motor field up to and down from the high speed. The limit on these step rates is eventually the motor, but the drive cost also becomes an important consideration for the home user. So here we will consider high speeds as being in the motor's slew region, up to a maximum of 5000 step/sec, and a drive price of approximately £25.

There are many different types of motor, with varying numbers of steps/revolution. The motors driven and described here are four-phase hybrid stepping motors with 200 step/rev. The motor phases (coils) may also be sequenced differently to produce the normal 200 steps or half-stepped to give 400 steps. The drive card used does this and all references to the step pulses assume 400 steps/rev.

Torque Versus Speed

The torque available will, as already stated, depend on the motor and drive combination. In figure 1 we see speed torque curves for a 22 frame size motor drive from a APKS 1186 drive card. The lowest curve is the start/stop curve, the upper curves represent the torque available when the field is accelerated.

If the motor is driven from its nominal voltage with no external resistors and a coil energised, the current will rise exponentially. To reach 95% of full current will take 3LR.

If, however, we double the voltage and fit a series resistor equal to R, the time to reach the same current will only be 1.5LR, due to the time constant for the exponential being halved.

As the total resistance is doubled, the motor current remains the same but switching times are improved, allowing current to be maintained in the coils at higher step frequencies. With the PKS 1186 drive card at nominally 24 volts, and a 6 volt 22 frame size motor, we have the L/2R torque curve. This may be extended to the L/4R curve by increasing the drive voltage to 48 volts if required.

Higher voltages and larger resistors are possible but losses are also high. More sophisticated drives, such as the PKS 1054, overcome this by current regulation. With this technique step frequencies of 20,000 steps/sec. (3000 RPM) may be achieved.

Interfacing

System Torque Calculations

Often the rotary motion of the motor shaft has to be translated into a linear movement. The most common forms are a wheel moving along a surface , or a screw thread driving a nut. The torque required to drive such systems may be calculated and a motor chosen from its torque curve.

1. Wheel Drive
   B = weight of system kp.
   R = radius of wheel cm.
   M = mass of wheel kg.

Inertia of wheel
JW = M*R*R/2kp/cm2.
Effective inertia of weight JE = B*R*Rkp/cm2.
If velocity = V cm/sec,
then angular VEL W = V/2*PI*R rev/sec.
If the motor is driven at 400 steps/rev, then clock frequency = (400*V)/(2*PI*R)Hz
If JM = motor inertia
TS = time to accelerate the system, then total inertia JT = JM + JW + JE
Torque T = JT*400V/2R*TS*0.9/180*1.021 /1000 kp/cm
Here friction has been assumed to be negligible. If this is not so, add the frictional load to T for the total value.

B = weight kp
P = pitch of screw mm/rev.
E = efficiency of screw in %

Note:— The efficiency of a normal acme screw is only 35-45%. For best results a ball screw or acme screw with low friction nut (PTFE) should be used. The efficiency will then be greater than 80%.

Linear speed mm/sec = (steps/sec)/(steps/rev) *60*P*1E-3
Axial force kp = 62.8*T/P*E
I = torque kp/cm
Rotational equivalent of weight

JE = B*P*P*2.53*1E-4kp/cm2

The inertia of the screw
JS = D4*L*7.721E-4kp/cm2
D = diameter of screw cm
L = length of screw cm

Note:— The factor 7.72 assumes a steel screw,.
As before, the total inertia JT = JM+JE+JSc
Where JM = motor rotor inertia
therefore if V = velocity of mm/min step frequency SF = (V*400)/(60*P*.P1E-3) at 400 steps/rev torque T = JT*SF/TS*9.9PI/180*1.027/1000 kp/cm2
where TS = time to accelerate the system torque to overcome friction F

TF = F*P/62.8*Ekp/cm2

The total torque is then the sum of these two torque values.

Gearing

Gearing of the motor to the load can be advantageous. Any gearing multiplies the motor torque by the gear ratio (G), but divides the effective inertia by the gear ratio squared (G^2). This, therefore, improves the loading of the motor, but requires G times the motor speed to keep the system speed the same. Gearing will always incur some losses and more motor torque is required to make this up. Despite this, advantages are to gained from gearing, and where possible the effective inertia of the system should be equal to the motor's rotor inertia for maximum power transfer.

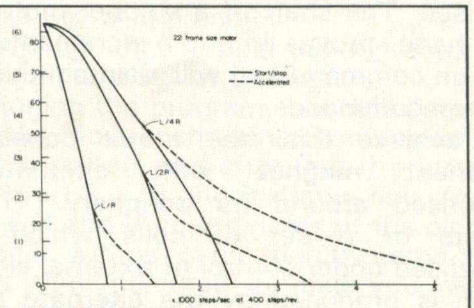


Figure 1: 22 frame size motor

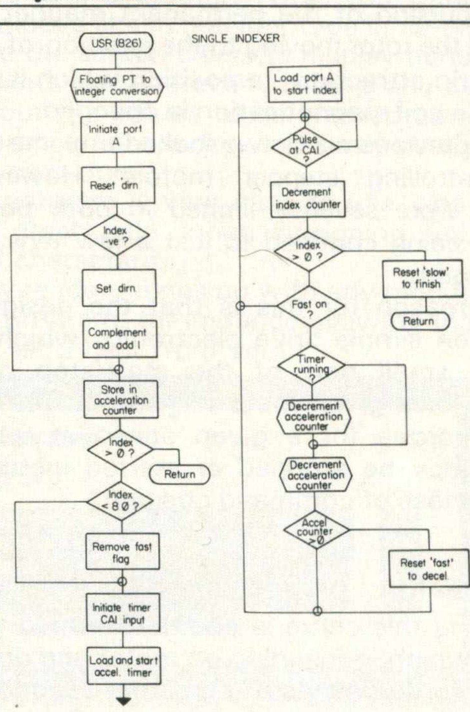


Figure 2: Single indexer

Drive Card

The drive used here is a PKS 1186 bipolar L/R drive. The coils of the motor are driven from two bridges of four transistors each, the four coils being wired as two pairs. The use of a bridge enables the current to be driven through the coils in either direction (figure 6).

The oscillator to step the phases is incorporated on the card. Two inputs control this oscillator and are termed 'fast' and 'slow', variable resistors setting the frequency. The 'S' input gives frequencies in the range of the Start/stop curve and thus will instantaneously produce and terminate motion. The 'F' input runs the oscillator into the motor's slew region and here the clock is accelerated, or ramped. This ramping is fixed by a capacitor on the card, but the acceleration time may be increased by addition of further capacitance. The sequence of the phase switching is controlled by the card with the 'dirn' input governing the stepping direction of the motor.

The card operates, and derives its cmos 12 volts, from one supply which is nominally 24 volts. No other supplies are necessary for operation.

Indexing

The algorithm used to control the position of the motor uses two counters and a timer. The timer is set to time the period in which the motor is accelerated, and this combined with one of the counters gives the deceleration point. This point is required to allow enough time, or steps, in which to decelerate the motor to a frequency at which it may be stopped without loss of rotor synchronisation.

During the timer period, the acceleration counter (AC) will be decremented from the total index value twice for each motor step pulse. The other counter, index counter (IC), is decremented only once for each pulse.

We start the index by loading AC and IC with the number of steps to run. The timer is loaded with the time taken to reach full speed. 'Fast' and 'Slow' are taken low to start the clock and the timer initiated. For each step pulse AC = AC-2 and IC = IC-1, in basic syntax. This will continue until the timer times out or if AC=0 we must have been constantly accelerating and moved half the distance.

If this is so, we will arrive at IC=0 if we now decelerate at the same rate. If the timer times out we start to decrement AC only once for each step pulse as IC. When the AC now reaches zero, and if we had reached full velocity before the timer finished, then we have a difference between IC and AC that is equal to the acceleration distance. Thus if we decelerate at this point, we will again reach zero speed as IC=0.

In practice we know that the motor in its start/stop region I.E. 'slow' may be stopped instantaneously without error. This being so, we decelerate to 'slow' and run to the end, IC=0, at slow speed. Zero on AC controls the 'fast' input to the drive, removing 'fast' when AC=0. Two profiles for a short index, AC=0 before timeout, and a long index are shown in figure 4.

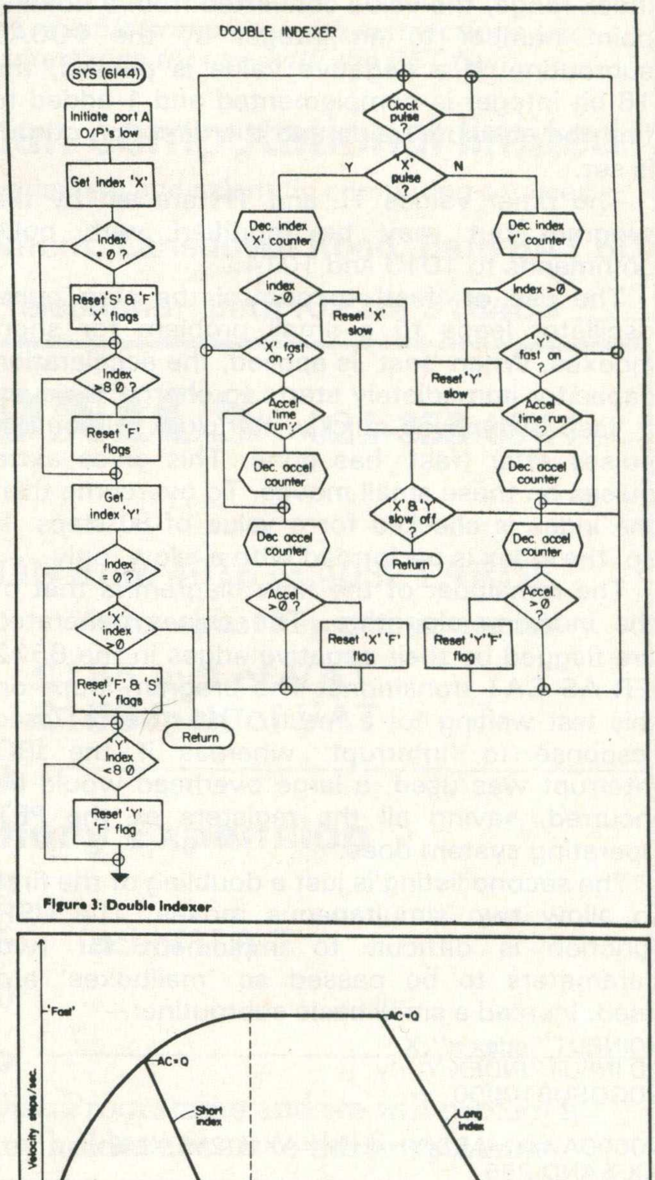


Figure 3: Double Indexer

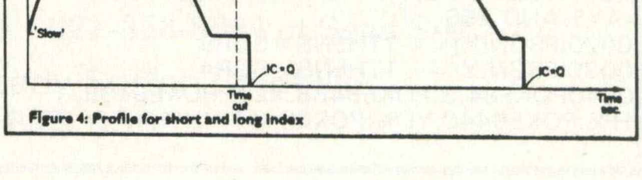


Figure 4: Profile for short and long index

Interfacing

Computer Implementation

To implement the indexing algorithm on a computer, the counters become consecutive memory locations and the timer may be an interrupt-driven counter or a hardware counter controlled by software.

The listings shown here are for a PET computer using the 6522 via to give the timer function and port I/O's for drive card control.

The first listing and flow-chart shows a single-axis control direction from a basic program with the index value passed by the USR function. This allows +/— 32767 steps as the index range, the value converted from a floating point number to an integer by the $D0A7 subroutine. If a negative value is passed, the 16-bit integer is complemented and 1 added to find the absolute value and the direction output is set.

The timer values TL and TH are set by the program but may be modified with poke commands to 1013 and 1014.

The use of 'fast' to control the step pulse oscillator leads to a small problem for short indexes. When 'fast' is applied, the acceleration capacitor immediately starts to charge, so even if 'fast' is removed quickly, the clock will output pulses after 'fast' has gone. This gives extra pulses on these small moves. To overcome this, the index is checked for a value of 80 steps. If so, the index is performed with a 'slow' only.

The remainder of the flow diagram is that of the indexing algorithm. The pulses generated are flagged by their negative edges in the 6522 IFR AS CA1 transitions. The program loops on this test waiting for a result. This gives a 7µsec response to 'interrupt' whereas if the IRQ interrupt was used, a large overhead would be incurred, saving all the registers as the PET operating system does.

The second listing is just a doubling of the first to allow two simultaneous moves. The USR function is difficult to implement for two parameters to be passed so 'mailboxes' are used. Instead a small basic subroutine:—

```
10INPUT"index x";X
20 INPUT"INDEX Y";'y
30GOSUB10000

10000AX%=ABS(X):XH%=AX%/256:XL%=
AX%AND 255
10010AY%=ABS(Y):YH%=AY%256:YL%
=AY%AND 255
10020IFSGN(X)=—1THENS=SOR8
10030IFSGN(Y)=—1THENS=SOR4
10040POKE84,S:POKE6438.XL%:POKE6439,
XH%:POKE6440,YL%:POKE6441,YH%
```

loads locations 6438 — 6441 ($1926 — $1929) with the two indexes. SYS(6144) transfers operation to the indexing program. This program transfers these values to zero page and performs the indexing as before. Two flags CA1 and CB1 are now used to detect the pulses generated by the two aces. Sign is provided by 'S' in the basic program and no negative conversion is necessary.

Due to the length of the double index program, the second cassette buffer cannot be used. Instead the top of memory is lowered to 6144 ($1800),POKE134,0:POKE135,24. and the program assembled above here.

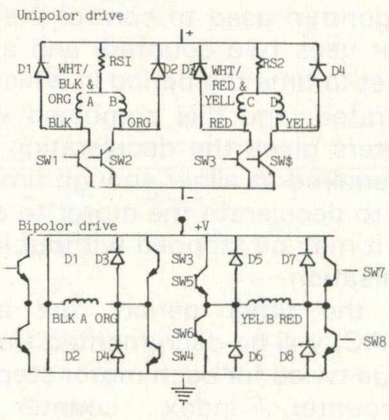
Interfacing

The PKS 1186 drive-card logic is all 12 volt cmos. To make the 5 volt port I/O's compatible, simple open collector transistors were used. A 5 volt supply for the CA1, CB1 input pullups is required and derived from the drive card voltage by a 400 MX zener diode.

The complete connection diagram for two drives is shown in figure 5.

All program listings will appear in next months issue.

CONNECTION DIAGRAM



FULL-STEP

	COIL			
STEP	A	B	C	D
1	1	0	0	1
2	1	0	1	1
3	0	1	1	0
4	0	1	0	1
5	1	0	0	1

HALF-STEP

	COIL			
STEP	A	B	C	D
1	1	0	0	1
2	1	0	0	1
3	1	0	1	0
4	0	0	1	0
5	0	1	1	0
6	0	1	0	0
7	0	1	0	1
8	0	0	0	1
1	1	0	0	1

FULL-STEP

	COIL	
STEP	A	B
1	1	1
2	+1	-1
3	-1	-1
4	-1	+1
1	+1	+1

HALF-STEP

	COIL	
STEP	A	B
1	1	1
2	+1	0
3	+1	-1
4	0	-1
5	-1	-1
6	-1	0
7	-1	+1
8	0	+1
1	+1	+1

Programming Tips

LOAD and SAVE revealed

This article describes the SAVE and LOAD commands, and shows how they can be used to provide a program overlay facility, to pre-load BASIC variables with constant values and to provide a local variable facility for subroutines.

The layout of the BASIC random-access memory is given in the CBM User Manual, pages A-8 and A-9. The important features are, first, that each BASIC statement is stored with a pointer to the next statement, and an end-of-statement marker (a zero byte). GOTO and GOSUB statements use the pointers to find their way about, but all other statements and Edit rely on the fact that each statement starts immediately after the preceding one. All statements, and Edit, recognise a pointer with a higher order byte of zero as the end of the last statement.

The second feature is that the start-of-variables (SOV) pointer in 42/43 usually points to a store location immediately after the zero pointer at the end of the program, but need not - a gap can be left without any ill effects. Whenever a program is edited, all the store contents from the point of edit up to SOV are moved to take account of any changes in length, so that the contents of any space between end of program and SOV are preserved.

The SAVE command

This can be obeyed either as a program statement or as an immediate mode command; the only difference is that in immediate mode, messages relating to the progress of writing to cassette tape are displayed on the screen. SAVE transfers a "store image", i.e. the actual store contents, from the address specified in 40/41 to the address specified in 42/43 inclusive, together with the start and end addresses. This will normally be the program statements, but 42/43 can be adjusted to make SAVE store, for example, the values of BASIC variables also.

The LOAD command

This operates in one of two different ways, depending on whether it is being used as a program statement, or as an immediate mode command.

In immediate mode, the store image is transferred to the store at the addresses specified with the SAVEd program. The value of the pointer to start-of-program remains unchanged; the value of SOV (42/43) is set to the address following the address of the last byte loaded into the store, start-of-arrays and end-of-arrays is set to the same value (i.e. the CLR operation is performed), and all strings deleted. The current program and data pointers 119/120 and 62/63 are set to start-of-program. This means that all previous information program and data, becomes inaccessible.

The LOAD statement

If LOAD is obeyed as a statement, the action is significantly different. The stored image is transferred to the PET store, at the addresses specified with the SAVEd program. All the pointers remain unchanged except for the current program and data pointers, which are set to start-of-program. Thus if a program is LOADed which is no bigger than the one which contained the LOAD statement, all the variables, arrays and strings are still accessible; the new program can still use the old values. The new program will be started at its first statement. If the new program is smaller than the old one, there will be an unused gap between the end of the new program and the start of variables. Programs can continue to LOAD each other in a chain of any length, provided that no program is bigger than the first one. No check is made on this; if you load a program which is bigger than the first one, the new program will treat part of itself as the variable store, probably with catastrophic results.

Possible problems

Apart from the size problem, there are two other pitfalls to avoid. Strings are normally stored at the top of memory, and can be left for succeeding chained programs. However, if a reference is made to a simple string stored in a program, e.g. by an assignment statement such as

```
A$ = "ABCDE"
```

or a READ statement, e.g.

```
READ B$
=======
DATA "FGHIJ"
```

then the string is left in place in the program. If the program is replaced by its successor, the variables A$ and B$ may point to bits of the new program, and their values will be lost. This can be avoided by forcing the string into the top of memory, by making an expression out of it, e.g. A$ = "ABCDE"+""

```
READ B$: B$ = B$+""
```

Program Overlays

Overlaying is a technique by which a program can LOAD its successor, but leaves parts of itself accessible to be used by its successor. Parts which remain are known as resident routines: parts which are overwritten are known as overlays. The resident program could be a "menu" program, which calls different overlays under the direction of the user, each of which returns to the resident program on completion: or it could be a package of subroutines left to be used by successive chained programs. On the PET, when a short program is LOADed, the end of the preceding (longer) program is left undisturbed in the store: the only problem is to link it to the end of the new one.

To do this, we can take advantage of the fact that PAGE 0 locations 58/59 contain the address of the terminator of the preceding statement: thus during the first statement of a line, 58/59 points to the zero byte at the end of the previous line. If the section of the current program to be preserved starts at line 10000, then its address can be preserved in a variable I by starting line 10000 as follows:

```
10000 I = PEEK(58)+256*PEEK(59)+-1:RETURN
```

When the successor program is LOADed, I is still available. The last line of the new program can also find out where it is, and replace its forward pointer (to a zero end-of-program pointer) by the address of line 10000 from I: thus the last line of the new program is line 9999 and is called as a subroutine when the new program is started e.g.

```
9999 J=PEEK(58)+256*PEEK(59)+1:K-
=INT(I/256):
POKE J, I-256*K:POKEJ+1,K:RETURN
```

Note that GOTO, GOSUB and RETURN can bridge the gap, but DATA statements cannot. If the new program uses READ and DATA statements, the user must be careful not to read beyond the end of the new program. The new overlay and the old resident program are now linked together. Line 10000 can be called again prior to loading the next overlay and so on.

If string constants in the resident program have been assigned to string variables or READ into them, their values will still be accessible to an overlay. Similarly, a function DEFined in the resident program will still be available.

Pre-loading BASIC variables

For some applications, such as assemblers and disassemblers and menu-driven programs, there is a need to build up arrays of constant data when the program is started. This is usually done by an initial sequence of READ statements loading up the variables with their initial values, and can result in quite a long delay before the program gets going. There is also quite a waste of storage space, since the loading sequence of program, and DATA statements for numeric data take up space which is unused once the arrays have been loaded. DATA statements containing strings must be retained since the corresponding string variable will refer to them. The problem then has three parts: how to load up arrays in the first place and delete the loading sequence; how to save the program complete with arrays; and how to make the arrays accessible again when the saved program is loaded. The listing of a sample program is given in figure 2.

Setting up the pre-loaded values

The setting up part of the program and all DATA statements for numeric data are placed at the end of the program, in lines 61000 onwards in the example. Line 61000 records the address of what will eventually be the end of the program. Between lines 61000 and 62000 are inserted the statements for reading string values: the corresponding DATA statements must be before line 61000. Strings must not be produced by using string addition or string functions, as this would move them into the top of memory.

After all strings have been read, line 62000 obtains the current DATA address, which is checked to be less than the address of line 61000.

Statements to load up numeric values, and DATA statements containing the values, are inserted between lines 62000 and 63000. Numeric values can be READ, INPUT or calculated.

To remove all of the program from 61000 to the end, and any simple variables used during the pre-loading operation, we turn everything from the start of line 61000 to the start of arrays into one "line" of BASIC, then DELETE it. The two bytes immediately before start of arrays are set-up to look like end-of-program (line 63030) and their address (K) planted in the head of line 61000 (I + 1, I + 2). In lines 63000 and 63030, the program "types in" the line number 61000 (which deletes it), followed by CLR to tidy up the various pointers. The end-of-program pointer 42/43 is set to point to the end of arrays (46/47), so that the arrays will be included in a SAVE. At END, the BASIC interpreter will delete the "line" 61000, moving all the arrays down to the new end of program and adjusting the end-of-program pointer. A SAVE command will now save the program, less the setting up statements, but including the arrays.

Gaining access to the pre-load arrays

When the program is LOADed and RUN, the first thing it has to do is to set-up the start-of-variables and start-of-arrays pointers (42/43 and 44/45) to point to the true end of the program, and the end-of-arrays pointer (46/47) to point to the apparent end of program. This is done by the routine at lines 60000 and 60040. This routine obviously must not be called twice, so line 60000 converts itself into a STOP statement. Line 60010 finds the address of the end of the last statement (60040). Line 60010 resets end-of-arrays, and lines 60030 and 60040 reset start-of-variables and start-of-arrays.

Variations on this technique can be used to carry machine-code subroutines around with a BASIC program.

Local variables in subroutines

It is generally regarded as good programming practice to write programs in the form of a hierarchy of subroutines, and in time a programmer or team of programmers will build up a library of subroutines which are frequently used. Several techniques have been published for merging BASIC programs stored on tape or disc into a program entered from the keyboard, but two problems soon become apparent: line number clashes and identifier clashes. Line number clashes can usually be resolved by a RENUMBER facility, but identifiers are less systematically organised, and cannot be dealt with automatically. The only effective solution is to provide a facility to the user writing the subroutine to enable him to avoid identifier clashes.

Identifiers used in subroutines fall into one of two main classes: global or local. A global identifier is one which is intentionally the same as one written in another part of the program, because it is required to represent the same value. A local identifier is one which is used only within the subroutine: the problem in BASIC is that if it happens to be the same as one in another part of the program it will affect the same value. A local variable is one which is created on entry to a subroutine, is used instead of any pre-existing variable of the same indentifier (which becomes temporarily inaccessible), and is destroyed on exit from the subroutine, restoring access to the unaltered value of any pre-existing variable. If variables can be specified as local the user can give them any indentifiers he chooses without risk of a clash. Since the BASIC interpreter finds identifiers by searching the identifier table from the beginning, the desired effect can be achieved by moving the start-of identifiers down and inserting local variables at the beginning, then moving the start-of-identifiers back again at the end of the subroutine. This can be done entirely in BASIC, but for efficiency the corresponding machine-code routine given in figure 3 should be used. It is written in position independent code, and may be placed anywhere in the store. The example shows it in use in the second cassette buffer.

Method of use

When the routines which make up the program have been written, it is necessary to calculate how much space to allow for local variables. For each subroutine and the main program, the number of bytes needed is 7* (1 + number of local variables) + the maximum number of bytes needed by any subroutine called: for example

Routine	Number of locals	Subroutines called	Space needed
10	0	100, 200	MAX(63,70) = 70
100	3	300, 400	28+MAX(35,42) = 70
200	4	500, 600	35+MAX(28,21) = 63
300	4	—	35
400	5	—	42
500	3	—	28
600	2	—	21

Any subroutine not using locals takes up no extra space. When the program is first RUN, start-of variables (42/43) must be increased by this amount to make room for the local variables e.g.

```
10  I=PEEK(42)+25*PEEK(43)+70:K=INT(I/256):
    POKE 254, I-256*K
15 POKE 42, K:POKE 43, PEEK(254)
```

If the facility to pre-load arrays is used, this space must be created during setting-up of the pre-load values. If the program is to be re-started once it has been run, this sequence should not be obeyed again, otherwise another 70 bytes will be reserved.

When a subroutine with locals is called by GOSUB, the sequence POKE 254, n:SYS 826 is obeyed, where n is the number of local variables to the subroutine. A SYNTAX ERROR will be given if there is insufficient room between start-of-variables and end-of-program. Next, the locals are "declared", e.g. by the use of DIM and DEF statements. All identifiers appearing, except in the expression defining a user-defined function, will be treated as locals, e.g.

200 POKE 254,4:SYS 826:DIM I,A$
210 DEFFN A(X) = P*X+Q

The variables I, A$,A and X will be treated as locals. P and Q will be globals.

After this, do SYS 924. This checks that the specified number of identifiers has been declared, and that no arrays have been DIMensioned, and moves start-of-variables to include the new locals. Before leaving the subroutine by RETURN, do SYS 963 to MOVESTART-OF-VARIABLES back to eliminate the locals.

Limitations

In BASIC 4.0, a string stored at the top of memory contains a back-reference to the variable which refers to it. If a string variable is destroyed, e,g, because it was a local variable, the back-reference must be deleted also. The easiest way to do this is to assign a null string to the variable before destroying it, e.g.

299 A$ ="": SYS 963:RETURN

```
   1 REM "0:OLAY1"
  10 REM TEST FOR RESIDENT ROUTINES IN OVERLAYS.
  20 A$="HERE WE RE AGAIN "+"":X=111
  30 B$="HAPPY AS CAN BE"+"":Y=222
  40 REM JUST TO FILL IN A FEW VARIABLES.
  50 REM DON'T FORGET TO MOVE STRING CONSTANTS!
 100 GOSUB 1000
 110 REM LOAD ADDRESS OF RESIDENT ROUTINES INTO 'A'
 120 LOAD "0:OLAY2",8
1000 LA=PEEK(58)+256*PEEK(59)+1:RETURN
1010 REM A=ADDRESS OF LINE 1000.
2000 PRINTX,A$:RETURN
2010 PRINTY,B$:RETURN
```

```
  1 REM"0:OLAY2"
 10 GOSUB999:REM JOIN RESIDENT ROUTINES TO THIS OVERLAY.
 20 GOSUB2000:REM SEE IF THEY ARE ACCESSIBLE.
 30 GOSUB2010
 40 A$="ALL GOOD FRIENDS..."+"":X=333
 50 B$="..AND JOLLY GOOD COMPANY!"+"":Y=444
 60 GOSUB1000:LOAD"0:OLAY3",8
999 B=PEEK(58)+256*PEEK(59)+1:POKEB,A-256*INT(A/256):
    POKEB+1,INT(A/256):RETURN
```

```
  1 REM "0:OLAY3"
 10 GOSUB999:REM JOIN RESIDENT ROUTINES TO THIS OVERLAY.
 20 GOSUB2000:REM SEE IF THEY ARE ACCESSIBLE.
 30 GOSUB2010
 40 STOP
999 B=PEEK(58)+256*PEEK(59)+1:POKEB,A-256*INT(A/256):
    POKEB+1,INT(A/256):RETURN
```

```
   10 GOSUB60000
   20 FORJ=0TO3:PRINTA(J);A$(J):NEXT
   30 STOP
   40 DATA"HERE WE ARE AGAIN,
   50 DATA"HAPPY AS CAN BE.
   60 DATA"ALL GOOD FRIENDS AND...
   70 DATA"..JOLLY GOOD COMPANY!
60000 I=PEEK(58)+256*PEEK(59):POKEI+5,144:
      POKEI+6,58:POKEI+7,143:REM STOP RE-CALL
60010 FORI=ITO256STEP-1:J=I+1:I=PEEK(J)+256*PEEK(J+1)
      :NEXT:REM FIND END OF PROG.
60020 POKE 46,PEEK(42):POKE47,PEEK(43):REM
      RESET END OF ARRAYS.
60030 J=J+2:I=INT(J/256):POKE256,J-256*I:POKE45,I:
      POKE43,I

60040 POKE44,PEEK(254):POKE42,PEEK(254):RETURN
61000 I=PEEK(58)+256*PEEK(59):REM ADDRESS OF END
      OF PREVIOUS LINE.
61001 REM********************************
61002 REM PRELOADING SEQUENCE, ENTERED
61003 REM INITIALLY TO SET UP ARRAYS.
61004 REM SET UP STRING FIRST.
61005 REM********************************
61010 DIMA(3),A$(3):FORJ=0TO3:READA$(J):NEXT
62000 REM********************************
62001 REM CHECK STRINGS TO BE IN PERMAN-
62002 REM ENT PROGRAM, THEN LOAD NUMERIC
62003 REM********************************
62010 J=PEEK(62)+256*PEEK(63):IFJ>ITHEN PRINT
      "STRINGS OVERLAP":STOP
62020 FORJ=0TO3:READA(J):NEXT
62030 DATA111,222,333,444
63000 REM********************************
63001 REM MOVE END-OF-PROGRAM TO INCLUDE
63002 REM ARRAYS, DELETE FROM 61000 TO
63003 REM END OF SIMPLE VARIABLES.
63004 REM********************************
63010 K=PEEK(44)+256*PEEK(45)-2:REM (ADDRESS OF
      START OF ARRAYS)-2
63020 J=INT(K/256):POKEI+1,K-256*J:POKEI+2,J:
      REM MAKE PRELOAD PROGRAM ONE LINE.
63030 PRINT"(clr,3cd)";PEEK(I+3)+256*PEEK(I+4):
      POKE623,13:POKE158,2:REM DELETE.
63040 PRINT"(clr)":POKE624,13
63050 POKEK+1,0:POKE42,PEEK(46):POKE43,PEEK(47):
      END:REM SET E.O.P.=E.O.A.
```

```
  1 REM "LOCALBAS" 25/05/82
 10 FORI=826TO975:READA:POKEI,A:NEXT:STOP
826 DATA166,40,165,41,134,31,133,32,160,0,
    177,31,170,200,177,31,208,242,230,31
846 DATA208,2,230,32,165,42,133,92,165,43,
    133,93,166,254,232,136,56,165,92,233
866 DATA7,133,92,176,2,198,93,202,208,242,
    165,32,197,93,208,4,165,31,197,92
886 DATA176,50,169,128,145,92,160,6,185,41
    ,0,145,92,136,208,248,165,92,164,93
906 DATA133,42,132,43,105,7,144,1,200,133,
    44,132,45,133,46,132,47,96,160,2
926 DATA177,42,217,43,0,208,5,217,45,0,240
    ,9,166,119,208,2,198,120,198,119
946 DATA96,136,208,232,160,6,177,42,153,41
    ,0,136,192,2,208,246,96,160,2,177
966 DATA42,170,136,177,42,133,42,134,43,96
```

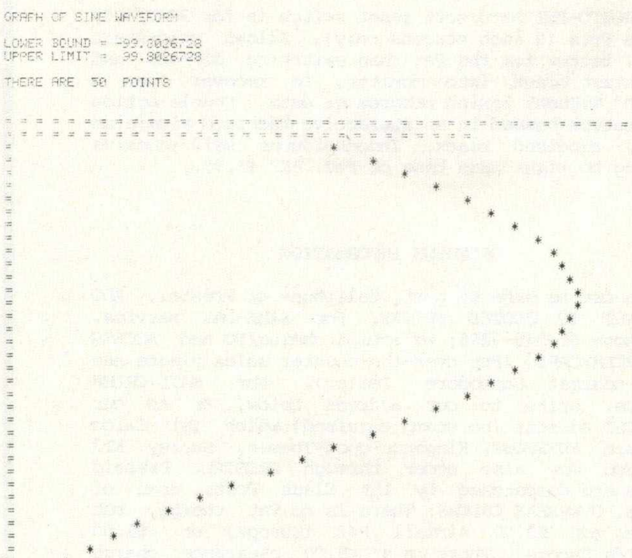
```
  1 REM "LOCALSTEST" 25/05/82
  5 POKE254,PEEK(42)+21+256*(PEEK(42)>=235):POKE43,
    PEEK(43)-(PEEK(42)>=235):POKE42,PEEK(254):CLR
  7 REM (A METHOD OF MOVING S.O.V. BY 21 WITHOUT USING
    ANY VARIABLES)
 10 A=1:B=2:C=3:D=4
 20 DIMX(3):FORI=0TO3:X(I)=I+10:NEXT
 30 GOSUB100:REM DISPLAY POINTERS.
 40 GOSUB200
 50 GOSUB100:REM POINTERS RESTORED.
 60 END
100 FORI=40TO52STEP2:PRINTPEEK(I)+256*PEEK(I+1);:
    NEXT:PRINT
110 PRINTA;B;C;D;:FORI=0TO3:PRINTX(I);:NEXT:PRINT
    :PRINT:RETURN
200 POKE254,2:SYS826
210 DIMB,C
220 SYS924
230 B=22:C=33:GOSUB100:REM NOTE S.O.V. CHANGE.
240 SYS963:RETURN
```

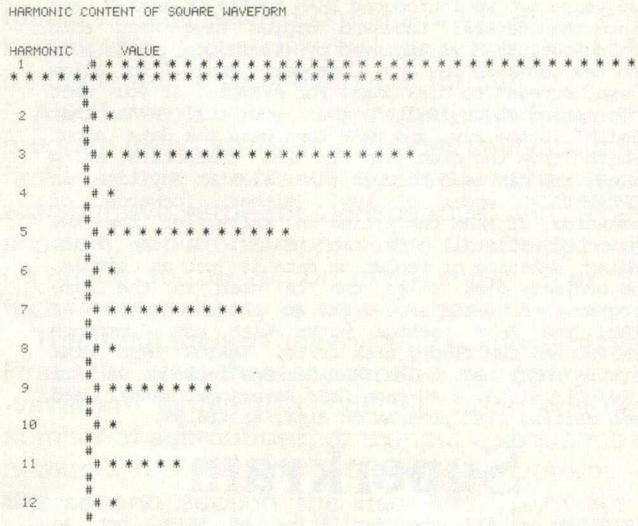
Fourier analysis on the PET

WE PRESENT an entirely practical, non-mathematical and basic - and Basic description of how to take a single cycle of any waveform and reduce it to its component parts, revealing in some detail the nuts and bolts' which go to make up all kinds of sounds and vibrations. This should prove of considerable interest to Pet owners who also happen to be musicians, engineers, mechanics, teachers and students in various branches and stages of physics and elementary mathematics.

It is possible, and sometimes useful, to describe any periodic waveform in terms of a number of pure sinewaves. Fourier analysis is a technique which can be used to extract each of the possibly numerous harmonics that constitute a complex waveform. Harmonic content distinguishes the tone of one musical instrument from another, even though they are playing the same note.

Spoken vowel sounds are different because the throat, nose and mouth form filters with different resonant frequencies which emphasise certain of the harmonics because it has an upper frequency limit of 20K Hz. That will lead inevitably to a distorted squarewave being fed





into the loudspeakers, since the theoretical harmonic series of a squarewave continues to infinity in the sound spectrum.

Figure 2 shows the relative content of the first 15 harmonics showing that the higher harmonics are less and less significant and contribute less to the final sound. Such distortion may not be particularly important, as the human ear cannot hear these higher harmonics. This transcription error, however, would be detected by audio test equipment.

It may not be immediately obvious that a periodic wave form need consist of only its fundamental — the period with which it repeats — and harmonics of that fundamental. Note, however, that fundamental frequency means just that. If the waveform contains any component waveform that is not an integer multiple of the fundamental, the fundamental was chosen incorrectly.

Harmonics analysis of the tides might show that the fundamental frequency of oscillation was the lunar month. After all, it is the attraction of the moon which causes the tides. Should the sun have any effect on the tides, the periodic fundamental would have to be the lowest common denominator between the effect of the moon and the sun.

A practical tide predictor, which is one application for such analysis, may, for the sake of simplicity, ignore the minor effect of the sun if the prediction is accurate enough.

Figures 2 and 4 show the harmonic content of a square wave and a triangle wave. Figure 2 shows that the square wave contains a greater

proportion of harmonics than the triangle. Both contain only odd harmonics — fundamental, third, fifth, seventh and so on — yet in one case they add up to a square wave and in the other a triangle. This is due to phase differences between the harmonics. If one considers two sine waves of the same freqency (as figure 1) and superimposes them, when the peaks occur in the same place both the peaks and the troughs are magnified. The new waveform is the sum of the two.

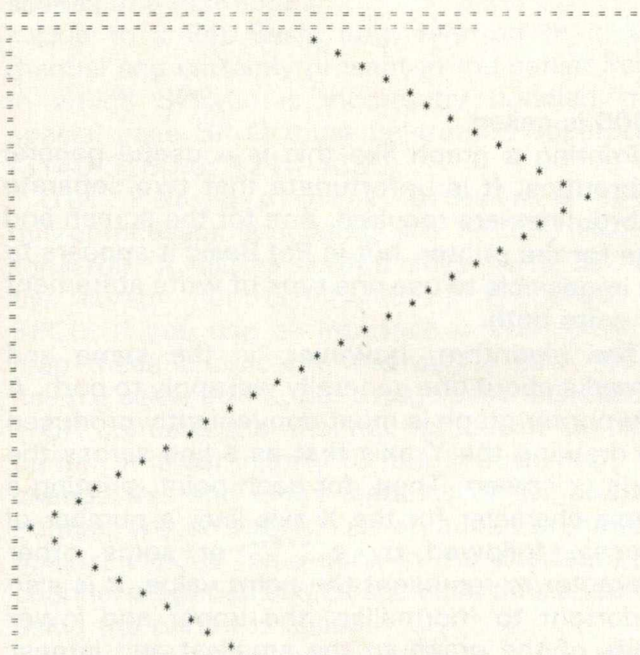
If the second waveform is shifted along half of the total wavelength (180 degrees) the peak of the first would be summed with the trough of the second.

The net result would be of the two waveforms cancelling-out to leave nothing at all. When two stones are dropped in a pond and the ripples meet, the wavelets are amplified in some places, reduced to still water in others.

GRAPH OF TRIANGLE WAVEFORM

_OWER BOUND = -130
UPPER LIMIT = 120

THERE ARE 50 POINTS



Three peaks

The third harmonic has three peaks in the same length, space or time as the one of the fundamental. When the third harmonic is added to the fundamental in such a way that the peak and trough of the fundamental are accentuated, then a triangle wave is produced. When the third harmonic reduces the peak and trough of the fundamental, a squarewave is formed

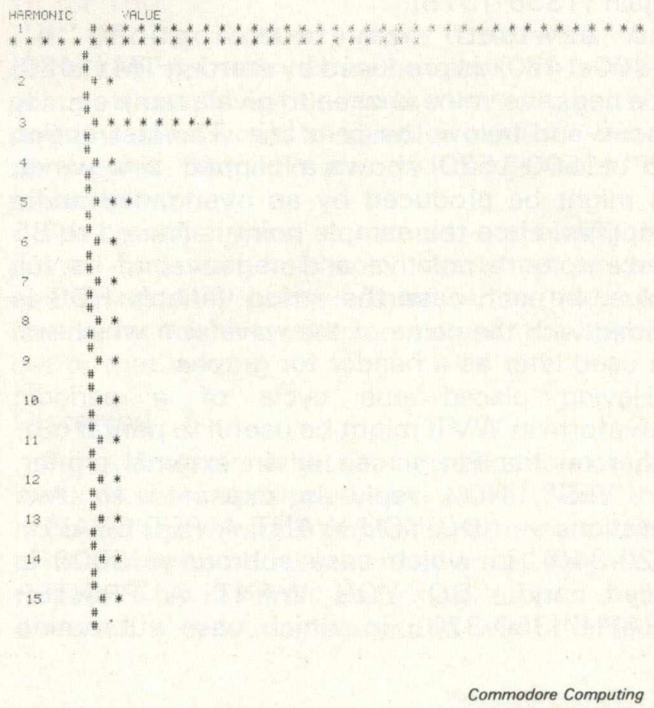
All the examples were generated by the program given at the end.Fourier analysis is set firmly in the realms of applied mathematics, although we settle for a descriptive, and, where possible, a pictorial approach. No attempt will be made to prove, or even show, that the technique or theory is soundly based.

The program is in several logical sub-sections. First, the user has the option to generate a waveform within the code by calling one of a number of subroutines, or to input a sequence of numbers representing a digitised waveform. In the second section the user may print-out a graph of the waveform so produced, either on the Pet screen, or to an external printer through an IEEE bus-to-RS232 converter.

The third stage is to analyse the waveform once for each harmonic, printing the amplitude of that harmonic and its phase angle. They are the two items of information required to say how much of information required to say how much of each harmonic must be added to the fundamental, and in what phase relationship.

In the last section the user may print-out a pictorial representation of the harmonic content, either on the screen or the external printer. As a further option the bar chart of the harmonics may be displayed as a logarithmic value; this has the effect of compressing widely-diverging values, so making the display more usable.

HARMONIC CONTENT OF TRIANGLE WAVEFORM



Basic Programs

When the program is started it asks the user for the number of sample points the waveform is to contain (120). Ten or fewer are too few for a meaningful analysis, more than 255 would not fit in a Pet Basic array (130-150).

Next, the user must select one of six options to set up the waveform in the array VW (160-250). If a zero is entered the program executes subroutine 1000 (260-280). This subroutine then asks the user to input NO — the number of sample points — digitised waveform points.

If the user typed "1" when selecting the waveform option, WV is set equal to a sine wave (1100-1150). The inherent SIN() function is called. It takes a value in radians — there are X^2* PI radians in a full circle, equivalent to 360 degrees; this accounts for the 6.283 ... constant in the calculation (1130).

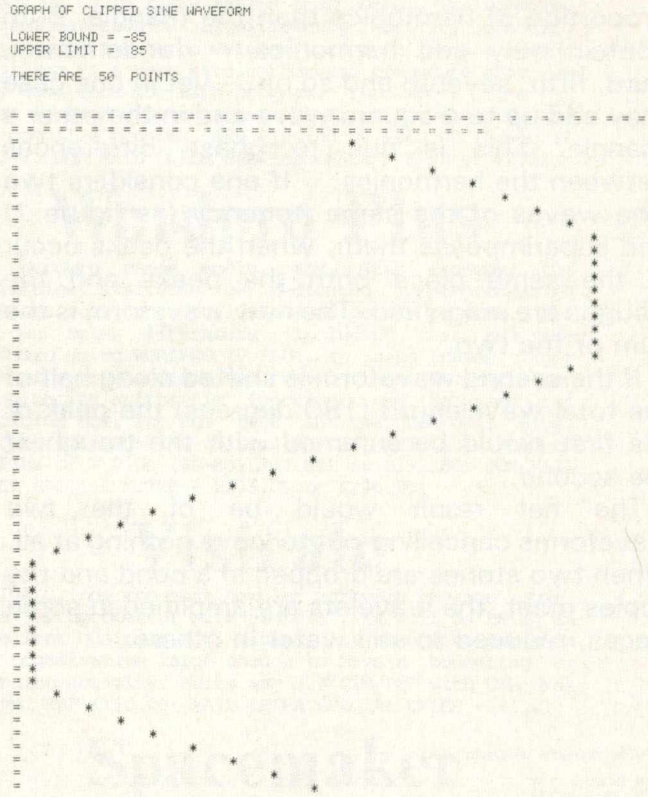
Normalising

SIN() returns a value in the range — 1 to + 1; this is multiplied by 100 to normalise it partially with the other waveforms and also to provide reasonably large numerical values from the calculations.

By selecting option "2" a square wave is placed in WV (1200-1290). The first half of the array is set to — + 100. A triangle wave, option "3" (1300-1375), is constructed by starting a counter at zero (TM, statement 1310) adding + 10 for the first quarter of the cycle (1315-1330), then subtracting 10 for the next half of the cycle (1335-1350).
The last quarter of the cycle is constructed by adding + 10 to the counter until it reaches zero again (1335-1375).

A sawtooth waveform, option "4" (1400-1470), is produced by starting TM (1420) at a negative value chosen to give a ramp equally above and below the zero line. The last option "5" (1500-1520) shows a 'clipped' sine-wave, as might be produced by an overloaded audio amplifier. Here the sample point is limited to 85 percent, both positive and negative, of its full value. In each case the string variable HDS is loaded with the name of the waveform which will be used later as a header for graphs.

Having placed one cycle of a periodic waveform in WV it might be useful to print it out, either on the Pet screen or an external printer. A "YES"/"NO" reply is expected to two questions — "DO YOU WANT A PET GRAPH" (320-340), in which case subroutine 3000 is called; and "DO YOU WANT A PRINTER GRAPH"(350-370), in which case subroutine



4000 is called.

Printing a graph like this is a useful general subroutine. It is unfortunate that two separate subroutines are required, one for the screen and one for the printer, but in Pet Basic it appears to be impossible to use one type of write statement to write both.

The algorithm, however, is the same and remarks about one generally will apply to both. A line-printer graph is most conveniently produced by drawing the Y axis first as a line across the page or screen. Then, for each point, printing a single character for the X-axis line, a number of spaces, followed by a "*" or some other character to represent the point value. It is also important to 'normalise' the upper and lower limits of the graph to the smallest and largest value in the array of points to be displayed.

Subroutine 4500 places the smallest and largest values to be found in WV into MN and MX. Again the algorithm is a simple and effective one. MN and MX are loaded with the first value in WV — WV(1) (4510-4520). Then the remaining values are checked; if any value is greater than MX it is placed in MX (4540) and if any value is smaller than MN, it is placed in MN (4550).

Both the graph plotting routines first print-out a header consisting of the name of the waveform — from HDS, the upper and lower graph plot limits and the number of points which make up the plot (3030-3050) or (4030-4120). Next a line of 'hash' characters is printed to form the Y-axis line (3060-3090 or 4130-4160).

The line length is fixed at 39 for the Pet as its screen is 40 columns wide, but it will be varied for the printer according to the paper width in use. The column width is stored in the variable PW, which is set initially to 70 in statement 54. One line of output is generated for each waveform point.

TW contains the total width of the plot (3120-4190) and each point will be some proportion of this (3130- or 4200), before the function INT() is applied, which has the effect of rounding the value in SP to the nearest integer. This is a useful trick — well worth remembering.

SP contains the number of spaces which must be printed between the X-axis line and the graph point. The PRINT function SPC() can be used on the Pet screen. It moves the cursor X places to the right, and is much faster than printing-out spaces in a FOR-loop.

Due to a Pet Basic bug, referred to in the manual and certainly present in the earlier Pets, in which SPC(O) is incorrectly handled, the special case SP O must be treated separately (3140-3160 or 4210-4230).

Our Pet-to-printer interface is a microprocessor simulating the IEEE to RS232 converter. It has its eccentricities but it handles the cursor control characters produced by SPC(). If you use an interface which does not map these characters, subroutine 8000-8040 has the same effect, but using space characters.

CH contains the channel number of the IEEE printer. It is set initially to four in statement 16; change this and every instance of the channel number will be altered. As an added precaution against various 'time-outs' in the interface, the channel is opened and closed each time a routine using the printer is called.

It is the statements 400 to 600 which calculate the harmonic content of the waveform. The process is very simple. For each harmonic every point in the waveform is multiplied by a point on a sinewave in the corresponding place in the wavelength, and then by a cosine point.

A cosine wave is always 90 degrees out of phase with a sinewave of the same wavelength — one-half PI radians, a quarter wavelength. The inner loop (480-520) multiplies each point in the waveform by the value a sinewave and a cosine wave would have at that point.

For the first harmonic — the fundamental — there is one complete sine and cosine wave. The products are summed into CS and SS. BN is an indication of how much cosine component there is in the waveform at the fundamental frequency, and AN how much sine-wave component. Inherent in this pair of numbers is the phase angle, and the harmonic is often interpreted as having so much sine component and so much cosine component. This is sufficient to describe the harmonic completely.

Always positive

Alternatively, we can use the harmonic amplitude — the square root of the sum of these two components squared (HA = SQR (AN+BN*BN)) and the phase angle — the arctangent of AM/BN. The harmonic amplitude will always be positive, as there cannot be less than zero harmonic amplitude. The phase angle will always be in the range of — PI radians to +PI radians.

This process is repeated, so that each sample point is multiplied by points generated as though there are two complete sine and cosine cycles in the wavelength.

This gives the sine and cosine components, the harmonic amplitude and phase angle for the second harmonic. Then with three cycles and four, and so on.

The variable DG is used to determine how many harmonics will be tackled. With DG set to 10 (statement 20) a harmonic series up to one-tenth of the number of sample points will be produced — i.e., five harmonics for 50 points, 15 for 150.

If DG is reduced more harmonics are computed for the same number of sample points. It should not be reduced to one, as then only one point would be sampled in each cycle of the highest harmonic and the result would be meaningless.

The graphs show a sample containing 50 points and the harmonic charts show one containing 150 points. There is no particular reason for this discrepancy, except that it looks better that way.

Distorted

Figure 6 shows what the numeric table output looks like for a clipped sine-wave. Note that although this example contains harmonics, there is no cosine component — they are all very close to zero. This is hardly surprising, as it is only a distorted sine-wave, computed with the Basic

Basic Programs

```
HARMONIC NUMBER 1
COS COMP=  2.56050262E-07  SINE COMP=  93.2256294
HARMONIC AMPLITUDE=  93.2256294
PHASE ANGLE=  2.74656512E-09

HARMONIC NUMBER 2
COS COMP= -9.71951522E-09  SINE COMP= -1.2501856E-07
HARMONIC AMPLITUDE=  1.2539581E-07
PHASE ANGLE=  .077588509

HARMONIC NUMBER 3
COS COMP=  1.18365278E-07  SINE COMP=  5.28212875
HARMONIC AMPLITUDE=  5.28212876
PHASE ANGLE=  2.24086318E-08

HARMONIC NUMBER 4
COS COMP=  4.56365524E-08  SINE COMP= -1.35236187E-07
HARMONIC AMPLITUDE=  1.42728838E-07
PHASE ANGLE= -.325458277

HARMONIC NUMBER 5
COS COMP=  2.72912439E-08  SINE COMP= -2.98098795
HARMONIC AMPLITUDE=  2.98098795
PHASE ANGLE= -9.15510039E-09


HARMONIC CONTENT OF CLIPPED SINE WAVEFORM

HARMONIC      VALUE
 1          # * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * *
         #
         #
 2        # *
         #
         #
 3        # * * *
         #
         #
 4        # *
         #
         #
 5        # * *
         #
         #
```

SIN() function.

The last stage in the Fourier analysis of the chosen waveform is to print-out the bar chart of the harmonics. Subroutine 6000 displays a series of lines of stars on the screen, one for each harmonic produced. The bars are normalised so that the harmonic with the most power (greatest amplitude) stretches right across the screen and all the others are some percentage of this.

Subroutine 7500 finds the largest value in HB and puts it into MX. HB contains the number of stars which will represent the harmonic (6030 or 7070). A FOR-loop is set to print-out the stars (6040-6060 or 7080-7100).

The printer routine has a few extra aids to improve the layout. For instance, there are two blank lines between each of the bars, thus increasing legibility many times (7024-7040). The harmonic number is printed. It is unfortunate that the printer interface did not like the use of a comma as a tabulate function in the PRINT statement. Because of that, subroutine 8100 was written to print a number padded -out to 10 columns with spaces.

This is also something of a cunning trick; convert the number to a string with STRS(), concatenate 10 spaces, and then print only the first 10 characters of the resulting string using LEFTS().

Fourier analysis is used widely throughout the engineering sciences to examine sounds and vibrations. Every building or bridge has its own natural resonant frequencies, since it is impossible and undesirable to make them totally rigid. Cracks and faults can be detected in such things as pipes and locomotive wheels by the way they ring when tapped.

The defects will cause different harmonics to appear; this is not to say that there are not more direct methods of testing.

High accuracy

An analogue – to – digital converter could be added to this program to enable the user to analyse many waveforms. For instance, the clipped sinewave could have been produced by an audio amplifier under test. It shows that the amplifier has saturated, and has produced harmonic distortion, primarily the odd harmonics. It must be realised, however, that results obtained with these waveforms are an ideal, produced to high accuracy by the internal SIN() function. Real digitisation will upset the results by introducing spurious harmonics.

For those who wish to experiment with the program, try the ramp waveform, since it should contain all harmonics,both odd and even. A pure sinewave will produce only one harmonic bar, at the fundamental frequency.

Try also placing a sheet of transparent graph paper over some oscillograms and inputting some real waveforms using the "O" option.

In a later part of this article some further waveforms will be analysed, including a pulse and pseudo – random noise. Further, having dissected a waveform in this way, it is possible to reconstitute it.

In practice this is like dismantling a complex mechanism – there is always something left over which should have been included. By using only some of the constituent harmonics, audio filters will notice the difference. In any case, we will continue to ignore such wonder as the fast – Fourier transform and vast amounts of mathematical theory.

```
10 REM DICTIONARY
12 REM AN - COSINE COMPONENT
14 REM BN - SINE COMPONENT
16 CH=4:REM PRINTER CHANNEL
18 REM CS - SUM OF COSINE
20 DG=10:REM HARMONIC INTERVAL
22 REM DX - SINE INTERVAL
24 REM FG - A FLAG
26 REM HA - CURRENT HARMONIC AMPLITUDE
28 REM HB(NO) - HARMONIC AMPLITUDES
30 REM HD$ - WAVEFORM NAME
32 REM I - LOOP COUNTER
34 REM J - LOOP COUNTER
36 REM K - LOOP COUNTER
38 REM L - LOOP COUNTER
40 REM MN - MINIMUM VALUE IN ARRAY
42 REM MX - MAXIMUM VALUE IN ARRAY
44 REM NO - NUMBER OF POINTS IN SAMPLE
46 REM OP - OPTION SELECT
48 REM P - NUMBER OF HARMONICS
50 REM PA - PHASE ANGLE
52 REM PR - 0 IF NOPRINT, 1 OTHERWISE
54 PW=70:REM PRINTER WIDTH
56 REM SP - SPACE/BAR LENGTH
58 REM SS - SUM OF SINE LOCATIONS
60 REM TM - TEMPORARY LOCATION
62 REM WV(NO) - POINTS IN SAMPLE
64 REM Y$ - YES/NO OPTION
100 PRINT"FOURIER ANALYSIS PROGRAM"
110 PRINT"======= ======== ======="
120 INPUT"HOW MANY SAMPLE POINTS";NO
130 IFNO>10ANDNO<255THEN155
140 PRINTNO;" OUT OF RANGE - TRY AGAIN"
150 GOTO120
155 DIM WV(NO),HB(NO)
160 PRINT"PLEASE SET OPTION"
170 PRINT"0 - INPUT WAVEFORM FOR ANALYSIS"
180 PRINT"1 - FOR SINE WAVE"
190 PRINT"2 - FOR SQUARE WAVE"
200 PRINT"3 - FOR TRIANGLE WAVE"
210 PRINT"4 - FOR SAWTOOTH WAVE"
220 PRINT"5 - FOR CLIPPED SINE"
250 INPUT"OPTION";OP
260 IF OP<>0 THEN290
270 GOSUB 1000
280 GOTO330
290 IF OP>0AND OP<6THEN320
300 PRINT"NO SUCH OPTION - TRY AGAIN"
310 GOTO160
320 ON OP GOSUB1100,1200,1300,1400,1500
330 INPUT"DO YOU WANT A PET GRAPH";Y$
340 IF LEFT$(Y$,1)="Y"THEN GOSUB3000
350 INPUT"DO YOU WANT A PRINTER GRAPH";Y$
360 IF LEFT$(Y$,1)="Y" THEN GOSUB4000
370 PR=0
380 INPUT"DO YOU WANT RESULTS TO PRINTER";Y$
390 IFLEFT$(Y$,1)="Y" THEN PR=1
400 REM DO THE HARMONIC ANALYSIS
410 REM INTERVAL DEGREE
420 P=INT(NO/DG)
430 FOR I=1TO P
440 DX=2*3.14159265/NO
450 CS=0
460 SS=0
470 REM THE SUMMATION
480 FOR J=1 TO NO
490 TM=I*J*DX
500 CS=CS+WV(J)*COS(TM)
510 SS=SS+WV(J)*SIN(TM)
520 NEXT J
530 REM COSINE COMPONENT
540 AN=2*CS/NO
550 REM THE SINE COMPONENT
560 BN=2*SS/NO
570 REM HARMONIC AMPLITUDE
580 HA=SQR((AN*AN)+(BN*BN))
585 HB(I)=HA
590 REM PHASE ANGLE BETWEEN AN AND BN
600 PA=ATN(AN/BN)
610 PRINT"HARMONIC NUMBER ";I
620 PRINT"COS COMP= ";AN;" SINE COMP= ";BN
630 PRINT"HARMONIC AMPLITUDE= ";HA
640 PRINT"PHASE ANGLE= ";PA
650 PRINT
660 IF PR=1 THEN GOSUB5000
665 NEXT I
670 FG=1
680 INPUT"DO YOU WANT HARMONIC CHART ON PET";Y$
690 IF LERT$(Y$,1)="Y" THEN GOSUB6000
700 INPUT"DO YOU WANT CHART ON PRINTER";Y$
710 IFLEFT$(Y$,1)="Y" THEN GOSUB7000
720 IF FG=0 THEN STOP
730 FG=0
740 INPUT"DO YOU WANT LOG(AMPLITUDES)";Y$
750 IF LEFT$(Y$,1)<>"Y" THEN STOP
760 FOR I=1TOP
770 HB(I)=LOG(HB(I))
780 NEXT I
790 GOTO680
1000 REM INPUT WAVEFORM POINTS
1010 INPUT"NAME WAVEFORM";HD$
1020 PRINT"THERE ARE ";NO;" POINTS TO BE ENTERED"
1030 FOR I=1 TO NO
1040 PRINT"INPUT POINT ";I;
1050 INPUT WV(I)
1060 NEXT I
1070 PRINT"O.K."
1080 RETURN
1100 REM GENERATE SINE WAVE
1110 HD$="SINE WAVEFORM"
1120 FORI=1TO NO
1130 WV(I)=SIN(I/NO*6.28318531)*100
1140 NEXT I
1150 RETURN
1200 REM GENERATE SQUARE WAVEFORM
1210 HD$="SQUARE WAVEFORM"
1220 TM=NO/2
1230 FOR I=1 TO TM
1240 WV(I)=-100
1250 NEXT I
1260 FOR I=TM TO NO
1270 WV(I)=+100
1280 NEXT I
1290 RETURN
1300 REM GENERATE TRIANGLE WAVE
1305 HD$="TRIANGLE WAVEFORM"
1310 TM=0
1315 FOR I=1 TO INT(NO*.25)
1320 WV(I)=TM
1325 TM=TM+10
1330 NEXT I
1335 FOR J=I TO INT(NO*.75)
1340 WV(J)=TM
1345 TM=TM-10
1350 NEXT J
1355 FOR I= J TO NO
1360 WV(I)=TM
1365 TM=TM+10
1370 NEXT I
1375 RETURN
1400 REM GENERATE SAWTOOTH WAVE
1410 HD$="SAWTOOTH WAVEFORM"
1420 TM=-NO*5
1430 FOR I=1 TO NO
1440 WV(I)=TM
1450 TM=TM+10
1460 NEXT I
1470 RETURN
1500 REM GENERATE CLIPPED SINEWAVE
1520 GOSUB 1100
1530 FOR I=1 TO NO
1540 IFWV(I)>85THEN WV(I)=85
1550 IFWV(I)<-85THEN WV(I)=-85
1560 NEXT I
1565 HD$="CLIPPED SINE WAVEFORM"
1570 RETURN
3000 REM PRINT GRAPH OF WAVEFORM
3001 REM IN WV ON PET SCREEN
3010 PRINT "GRAPH OF ";HD$
3020 GOSUB 4500
3030 PRINT "LOWER BOUND = ";MN
3040 PRINT "UPPER LIMIT = ";MX
3050 PRINT "THERE ARE ";NO;"POINTS"
3060 FORI=1TO39
3070 PRINT "#";
3080 NEXT I
3090 PRINT
3100 FOR I=1 TO NO
3110 PRINT"#";
3120 TW=MX-MN
3130 SP=INT(((WV(I)-MN)/TW*36)+0.5)
3140 IFSP>0THEN3170
3150 PRINT"*"
3160 GOTO3180
3170 PRINT SPC(SP);"*"
3180 NEXT I
3190 RETURN
4000 REM PRINT GRAPH OF WAVEFORM
4001 REM IN WV ON EXTERNAL PRINTER
4010 OPEN CH,CH
4020 GOSUB 4500
4030 PRINT#CH
4040 PRINT#CH
4050 PRINT#CH,"GRAPH OF ";HD$
4060 PRINT#CH
4070 PRINT#CH,"LOWER BOUND = ";MN
4080 PRINT#CH,"UPPER LIMIT = ";MX
4090 PRINT#CH
4100 PRINT#CH,"THERE ARE ";NO;" POINTS"
4110 PRINT#CH
4120 PRINT#CH
4130 FORI=1TOPW
4140 PRINT#CH,"= ";
4150 NEXT I
4160 PRINT#CH
4170 FOR I=1TONO
4180 PRINT#CH,"= ";
4190 TW=MX-MN
4200 SP=INT(((WV(I)-MN)/TW*PW)+0.5)
4210 IF SP>0 THEN4240
4220 PRINT#CH,"*"
4230 GOTO4250
4240 PRINT#CH,SPC(SP);"*"
4250 NEXT I
4260 CLOSE CH
4270 RETURN
4500 REM FIND LARGEST (MX) AND SMALLEST
4501 REM (MN) VALUES IN WV
4510 MX=WV(1)
4520 MN=WV(1)
4530 FOR I=2TONO
4540 IFWV(I)>MXTHEN MX=WV(I)
4550 IFWV(I)<MNTHEN MN=WV(I)
4560 NEXT I
4570 RETURN
5000 REM PRINT RESULTS ON PRINTER
5010 OPEN CH,CH
5020 PRINT#CH
5030 PRINT#CH,"HARMONIC NUMBER";I
5040 PRINT#CH,"COS COMP= ";AN;" SINE COMP= ";BN
5050 PRINT#CH,"HARMONIC AMPLITUDE= ";HA
5060 PRINT#CH,"PHASE ANGLE= ";PA
5070 CLOSE CH
5080 RETURN
6000 REM PRINT HARMONIC BAR CHART ON PET
6010 GOSUB7500
6020 FOR K=1TO P
6030 SP=INT(((HB(K)/MX)*38)+0.5)
6040 FOR L=1TO SP
6050 PRINT"*";
6060 NEXT L
6070 PRINT
6080 NEXT K
6090 RETURN
7000 REM PRINT HARMONIC BAR CHART ON PRINTER
7010 GOSUB7500
7020 OPEN CH,CH
7022 PRINT#CH,"HARMONIC CONTENT OF ";HD$
7024 PRINT#CH
7026 PRINT#CH
7030 PRINT#CH,"HARMONIC       VALUE ";
7035 IF FG=0 THEN PRINT#CH," (LOG)"
7040 PRINT#CH
7050 FOR I=1 TO P
7055 GOSUB8100
7060 PRINT#CH,"# ";
7070 SP=INT(((HB(I)/MX)*(PW-10))+0.5)
7080 FOR J=1 TO SP
7090 PRINT#CH,"* ";
7100 NEXT J
7105 PRINT#CH
7110 PRINT#CH,"           #"
7120 PRINT#CH,"           #"
7140 NEXT I
7150 CLOSE CH
7160 RETURN
7500 REM LARGEST VALUE IN HB TO MX
7510 MX=HB(1)
7520 FOR K=2 TO P
7530 IF HB(K)>MX THEN MX=HB(K)
7540 NEXT K
7550 RETURN
8000 REM PRINT SP SPACES TO THE PRINTER
8010 FOR K=1 TO SP
8020 PRINT#CH,"  ";
8030 NEXT K
8040 RETURN
8100 REM PRINT NUMBER IN I ON PRINTER
8101 REM IN 10 COLUMNS
8110 PRINT#CH,LEFT$(STR$(I)+"          ",10);
8120 RETURN
READY.
```

Machine Code Programming

This month we bring you Tinymon, a machine code monitor for the VIC. It occupies less than 800 bytes, so still leaves you plenty of room to explore, even on the standard VIC.

Having typed in Tinymon (and more of that later), and RUN it, the monitor can be accessed at any time by typing SYS 13 (followed by RETURN). The Monitor gives you 6 extra commands.

1) All monitor instructions start with the prompt '.' To display memory, enter:-

. M 123E 12A7 (for example)

And the VIC will respond with something like:-

. 123E AA 39 17 FF 04 etc.

To change memory, display memory first, and then just type over the locations to be changed.

2) To display registers, enter:-

R.

and the VIC will display (for example)

```
   PC      SR      AC      XR      YR      SP
.;1234   30      00      22      33      F8
```

These are the internal registers of the 6502 processor. To change them, just type over the ones you wish to alter.

3) To return to Basic, enter:-

.X

4) To run a machine language program, enter:-

.G 1400 (for example)

5) To save specific memory locations, enter:-

.S ''NAME'', 01,1250,13A0

Where 01 is the device number (cassette deck in this instance), 1250 is the start address and 13A0 is the end address plus one.

6) To lead non-Basic programs, enter:-

.L ''NAME'', 01

Programs always load to where they were saved from.

The next program (listing 2) is the hex dump. If you're entering this originally on a PET, it's probably easiest to use this hex dump, and just type it in 'as is'. Type in the Basic program first (listing 1) enter the monitor with a SYS4 call, and away you go. When you've finished, the whole program can be saved to tape with the following syntax:-

S''TINYMON1'',01,0400,0815

```
TINYMON - THE PROGRAM


 100 PRINT"▒▒▒▒▒▒ TINYMON
 110 PRINT"▒ JIM BUTTERFIELD"
 120 SYS(PEEK(43)+256*PEEK(44)+078)
READY.
```

```
C*
    PC   IRQ  SR AC XR YR SP
.;  B780 E455 2C 34 3A 9D F8
.
.:  0400 00 18 04 64 00 99 22 93
.:  0408 11 11 12 1D 1D 1D 20 54
.:  0410 49 4E 59 4D 4F 4E 20 00
.:  0418 31 04 6E 00 99 22 11 20
.:  0420 4A 49 4D 20 42 55 54 54
.:  0428 45 52 46 49 45 4C 44 22
.:  0430 00 4C 04 78 00 9E 28 C2
.:  0438 28 34 33 29 AA 32 35 36
.:  0440 AC C2 28 34 34 29 AA 30
.:  0448 37 38 29 00 00 00 EA EA
.:  0450 A5 2D 85 22 A5 2E 85 23
.:  0458 A5 37 85 24 A5 38 85 25
.:  0460 A0 00 A5 22 D0 02 C6 23
.:  0468 C6 22 B1 22 D0 3C A5 22
.:  0470 D0 02 C6 23 C6 22 B1 22
.:  0478 F0 21 85 26 A5 22 D0 02
.:  0480 C6 23 C6 22 B1 22 18 65
.:  0488 24 AA A5 26 65 25 48 A5
.:  0490 37 D0 02 C6 38 C6 37 68
.:  0498 91 37 8A 48 A5 37 D0 02
.:  04A0 C6 38 C6 37 68 91 37 18
.:  04A8 90 B6 C9 BF D0 ED A5 37
.:  04B0 85 33 A5 38 85 34 6C 37
.:  04B8 00 00 00 00 00 00 00 00
.:  04C0 BF 78 AD FE FF 00 AE FF
.:  04C8 FF 00 8D 16 03 8E 17 03
.:  04D0 A9 80 20 90 FF 58 00 00
.:  04D8 68 85 05 68 85 04 68 85
.:  04E0 03 68 85 02 68 85 01 68
.:  04E8 85 00 00 BA 86 06 38 A5
.:  04F0 01 E9 02 85 01 A5 00 00
.:  04F8 E9 00 00 85 00 00 20 B2
.:  0500 FE 00 A2 42 A9 2A 20 DB
.:  0508 FD 00 A9 52 D0 1C A9 3F
.:  0510 20 D2 FF 20 B2 FE 00 A9
.:  0518 2E 20 D2 FF A9 00 00 85
.:  0520 27 20 40 FE 00 C9 2E F0
.:  0528 F9 C9 20 F0 F5 A2 07 DD
.:  0530 E6 FF 00 D0 12 85 1C 8A
.:  0538 0A AA BD EE FF 00 85 C1
.:  0540 BD EF FF 00 85 C2 6C C1
.:  0548 00 00 CA 10 E6 4C 4B FD
.:  0550 00 20 BD FD 00 90 F8 20
.:  0558 EE FD 00 20 BD FD 00 90
.:  0560 F0 20 EE FD 00 20 4C FE
.:  0568 00 F0 1F 20 B2 FE 00 A2
.:  0570 2E A9 3A 20 DB FD 00 20
.:  0578 C5 FD 00 A9 05 20 6F FE
.:  0580 00 A5 C3 C5 C1 A5 C4 E5
.:  0588 C2 B0 DF 4C 50 FD 00 4C
.:  0590 50 FD 00 20 FE FD 00 85
.:  0598 C1 86 C2 60 A5 C2 20 CC
.:  05A0 FD 00 A5 C1 48 4A 4A 4A
.:  05A8 4A 20 E4 FD 00 AA 68 29
.:  05B0 0F 20 E4 FD 00 48 8A 20
.:  05B8 D2 FF 68 4C D2 FF 18 69
.:  05C0 F6 90 02 69 06 69 3A 60
.:  05C8 A2 02 B5 C0 48 B5 C2 95
.:  05D0 C0 68 95 C2 CA D0 F3 60
.:  05D8 20 0D FE 00 90 07 AA 20
.:  05E0 0D FE 00 90 01 60 4C 4B
.:  05E8 FD 00 A9 00 00 85 2A 20
.:  05F0 40 FE 00 C9 20 F0 F9 20
.:  05F8 20 FE 00 90 17 20 40 FE
.:  0600 00 C9 30 90 10 20 35 FE
.:  0608 00 06 2A 06 2A 06 2A 06
.:  0610 2A 05 2A 85 2A 38 60 C9
```

```
.:  0618 3A 08 29 0F 28 90 02 69
.:  0620 08 60 20 CF FF C9 0D D0
.:  0628 F8 68 68 4C 50 FD 00 A5
.:  0630 91 C9 FE D0 05 08 20 CC
.:  0638 FF 28 60 20 61 FE 00 2C
.:  0640 2D 91 30 F8 60 20 4C FE
.:  0648 00 D0 08 A9 03 85 9A A9
.:  0650 00 00 85 99 60 85 1E A0
.:  0658 00 00 20 AF FE 00 B1 C1
.:  0660 20 CC FD 00 20 A4 FE 00
.:  0668 C6 1E D0 F1 60 20 0D FE
.:  0670 00 90 0B A2 00 00 81 C1
.:  0678 C1 C1 F0 03 4C 4B FD 00
.:  0680 20 A4 FE 00 C6 1E 60 A9
.:  0688 02 85 C1 A9 00 00 85 C2
.:  0690 A9 05 60 E6 C1 D0 06 E6
.:  0698 C2 D0 02 E6 27 60 A9 20
.:  06A0 2C A9 0D 4C D2 FF A2 00
.:  06A8 00 BD D0 FF 00 20 D2 FF
.:  06B0 E8 E0 16 D0 F5 20 B2 FE
.:  06B8 00 A2 2E A9 3B 20 DB FD
.:  06C0 00 A5 00 00 20 CC FD 00
.:  06C8 A5 01 20 CC FD 00 20 99
.:  06D0 FE 00 20 6F FE 00 4C 50
.:  06D8 FD 00 20 FE FD 00 85 01
.:  06E0 86 00 00 20 99 FE 00 85
.:  06E8 1E 20 83 FE 00 D0 FB F0
.:  06F0 EA 20 BD FD 00 A9 05 85
.:  06F8 1E 20 83 FE 00 D0 FB F0
.:  0700 DC 20 CF FF C9 0D F0 07
.:  0708 20 BD FD 00 85 01 86 00
.:  0710 00 A6 06 9A A5 00 00 48
.:  0718 A5 01 48 A5 02 48 A5 03
.:  0720 A6 04 A4 05 40 78 A6 06
.:  0728 9A 6C 02 C0 4C 4B FD 00
.:  0730 A0 01 84 BA 84 B9 88 84
.:  0738 B7 84 90 84 93 A9 02 85
.:  0740 BC A9 40 85 BB 20 CF FF
.:  0748 C9 20 F0 F9 C9 0D F0 1A
.:  0750 C9 22 D0 D9 20 CF FF C9
.:  0758 22 F0 26 C9 0D F0 0B 91
.:  0760 BB E6 B7 C8 C0 10 F0 C5
.:  0768 D0 EA A5 1C C9 4C D0 E2
.:  0770 A9 00 00 20 D5 FF 20 58
.:  0778 FE 00 A5 90 29 10 D0 F0
.:  0780 4C 50 FD 00 20 CF FF C9
.:  0788 0D F0 E2 C9 2C D0 F0 20
.:  0790 0D FE 00 29 0F F0 D3 C9
.:  0798 03 F0 FA 85 BA 20 CF FF
.:  07A0 C9 0D F0 CA C9 2C D0 E6
.:  07A8 20 BD FD 00 20 CF FF C9
.:  07B0 2C D0 F4 20 FE FD 00 85
.:  07B8 AE 86 AF 20 CF FF C9 20
.:  07C0 F0 F9 C9 0D D0 EC A5 1C
.:  07C8 C9 53 D0 F8 20 B2 FE 00
.:  07D0 A9 01 85 B9 20 82 F6 4C
.:  07D8 50 FD 00 0D 20 20 20 50
.:  07E0 43 20 20 53 52 20 41 43
.:  07E8 20 58 52 20 59 52 20 53
.:  07F0 50 4D 52 58 47 3A 3B 4C
.:  07F8 53 86 FD 00 B7 FE 00 23
.:  0800 FF 00 02 FF 00 F4 FE 00
.:  0808 E1 FE 00 2D FF 00 2D FF
.:  0810 00 1B FD 00 AA AA AA AA
.
.
READY.
```

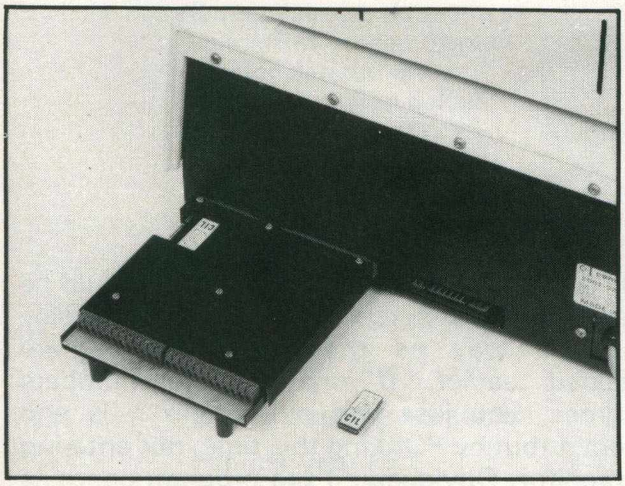
```
165  45   133  34   165  46   133  35   165  55
133  36   165  56   133  37   160  0    165  34
208  2    198  35   198  34   177  34   208  60
165  34   208  2    198  35   198  34   177  34
240  33   133  38   165  34   208  2    198  35
198  34   177  34   24   101  36   170  165  38
101  37   72   165  55   208  2    198  56   198
55   104  145  55   138  72   165  55   208  2
198  56   198  55   104  145  55   24   144  182
201  191  208  237  165  55   133  51   165  56
133  52   108  55   0    0    0    0    0    0
0    0    191  120  173  254  255  0    174  255
255  0    141  22   3    142  23   3    169  128
32   144  255  88   0    0    104  133  5    104
133  4    104  133  3    104  133  2    104  133
1    104  133  0    0    186  134  6    56   165
1    233  2    133  1    165  0    0    233  0
0    133  0    0    32   178  254  0    162  66
169  42   32   219  253  0    169  82   208  28
169  63   32   210  255  32   178  254  0    169
46   32   210  255  169  0    0    133  39   32
64   254  0    201  46   240  249  201  32   240
245  162  7    221  230  255  0    208  18   133
28   138  10   170  189  238  255  0    133  193
189  239  255  0    133  194  108  193  0    0
202  16   230  76   75   253  0    32   189  253
0    144  248  32   238  253  0    32   189  253
0    144  240  32   238  253  0    32   76   254
0    240  31   32   178  254  0    162  46   169
58   32   219  253  0    32   197  253  0    169
5    32   111  254  0    165  195  197  193  165
196  229  194  176  223  76   80   253  0    76
80   253  0    32   254  253  0    133  193  134
194  96   165  194  32   204  253  0    165  193
72   74   74   74   74   32   228  253  0    170
104  41   15   32   228  253  0    72   138  32
210  255  104  76   210  255  24   105  246  144
2    105  6    105  58   96   162  2    181  192
72   181  194  149  192  104  149  194  202  208
243  96   32   13   254  0    144  7    170  32
13   254  0    144  1    96   76   75   253  0
169  0    0    133  42   32   64   254  0    201
32   240  249  32   32   254  0    144  23   32
64   254  0    201  48   144  16   32   53   254
0    6    42   6    42   6    42   6    42   5
42   133  42   56   96   201  58   8    41   15
40   144  2    105  8    96   32   207  255  201
13   208  248  104  104  76   80   253  0    165
145  201  254  208  5    8    32   204  255  40
96   32   97   254  0    44   45   145  48   248
96   32   76   254  0    208  8    169  3    133
154  169  0    0    133  153  96   133  30   160
0    0    32   175  254  0    177  193  32   204
253  0    32   164  254  0    198  30   208  241
96   32   13   254  0    144  11   162  0    0
129  193  193  193  240  3    76   75   253  0
32   164  254  0    198  30   96   169  2    133
193  169  0    0    133  194  169  5    96   230
193  208  6    230  194  208  2    230  39   96
169  32   44   169  13   76   210  255  162  0
0    189  208  255  0    32   210  255  232  224
22   208  245  32   178  254  0    162  46   169
59   32   219  253  0    165  0    0    32   204
253  0    165  1    32   204  253  0    32   153
254  0    32   111  254  0    76   80   253  0
32   254  253  0    133  1    134  0    0    32
153  254  0    133  30   32   131  254  0    208
251  240  234  32   189  253  0    169  5    133
30   32   131  254  0    208  251  240  220  32
207  255  201  13   240  7    32   189  253  0
133  1    134  0    0    166  6    154  165  0
0    72   165  1    72   165  2    72   165  3
166  4    164  5    64   120  166  6    154  108
2    192  76   75   253  0    160  1    132  186
132  185  136  132  183  132  144  132  147  169
2    133  188  169  64   133  187  32   207  255
201  32   240  249  201  13   240  26   201  34
208  217  32   207  255  201  34   240  38   201
13   240  11   145  187  230  183  200  192  16
240  197  208  234  165  28   201  76   208  226
169  0    0    32   213  255  32   88   254  0
165  144  41   16   208  240  76   80   253  0
32   207  255  201  13   240  226  201  44   208
240  32   13   254  0    41   15   240  211  201
3    240  250  133  186  32   207  255  201  13
240  202  201  44   208  230  32   189  253  0
32   207  255  201  44   208  244  32   254  253
0    133  174  134  175  32   207  255  201  32
240  249  201  13   208  236  165  28   201  83
208  248  32   178  254  0    169  1    133  185
32   130  246  76   80   253  0    13   32   32
32   80   67   32   32   83   82   32   65   67
32   88   82   32   89   82   32   83   80   77
82   88   71   58   59   76   83   134  253  0
183  254  0    35   255  0    2    255  0    244
254  0    225  254  0    45   255  0    45   255
0    27   253  0
READY.
```

A NOTICE TO ALL

PET LOVERS

INTRODUCE YOUR PETS TO OURS

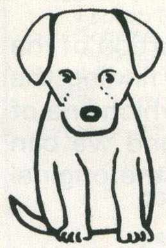
AND SOLVE YOUR INTERFACE PROBLEMS



Only £195.00

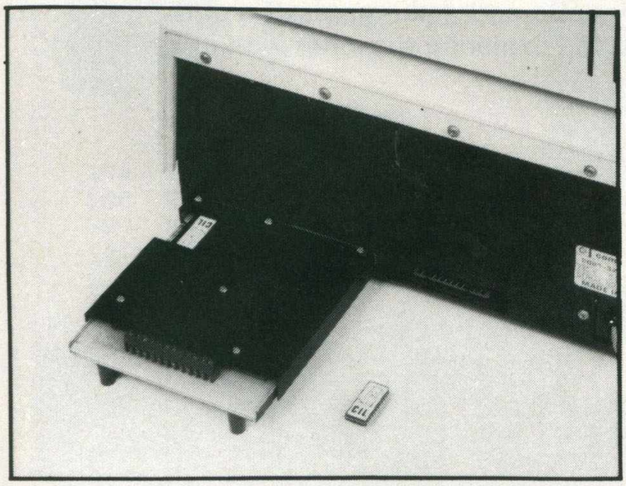
ANALOGUE/DIGITAL I/O

THE PUPI



* 4 ANALOGUE INPUTS (12BIT)
* 2 ANALOGUE OUTPUTS (12 BIT)
* 4 RELAY OUTPUTS
* 4 LOGIC INPUTS

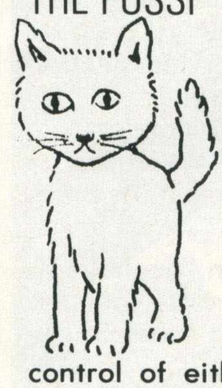
When connected to the "PET" User Port the PUPI gives you all the above features together with an operating system in EPROM, which interacts with Basic's variables, giving extremely simple operation. Inputs and outputs are ±10V and relays are rated at 10VA. Logic inputs can be used for microswitch sensing etc.



Only £195.00

HIGH SPEED A/D CONVERTER

THE PUSSI



* 4 ANALOGUE INPUTS (12 BIT)
* 50 MICROSECOND CONVERSION
* STOP AND START TRIGGERS
* DATA ACQUISITION SOFTWARE

Using an operating system in EPROM, the PUSSI provides a high speed A-D Converter with 4 multiplexed inputs, which is under control of either software, or remote start/stop triggers. A-D Conversion can be carried out from Basic, or Machine Code, with up to 1500 readings entered directly into memory at a software determined rate.

CIL

MICROSYSTEMS LTD

CIL MICROSYSTEMS LTD.
DECOY RD.,
WORTHING,
SUSSEX BN14 8ND.
TELEX: 87515 WISCO G ATTMIC
TEL: (0903) 210474

Write, phone or to obtain further information circle number

Machine Code

When loaded into a VIC, the program will relocate itself automatically, Almost inevitably, there will be mistakes in typing in a program of this size. To assist in checking the program, once you've finished typing it all in, type NEW, and then enter the following short program:-

```
1 FORI=1104TO2072:S=S+PEEK(I):A=A+1:I-
FA=8THENPRINTS;:A=0:S=0
2 NEXTI
```

The following table shows what should be displayed on your screen. If any of the totals disagree, you've made a mistake. Each number represents the addition of 8 bytes of code: thus, the first figure of 756 belongs to the first line of machine code:-

450 A5 2D 85 A5 2E 85 23

(as this is where the machine code portion of the program begins), and so on. So, by finding the figure which is incorrect, we know which line of machine code contained the error, and we can then correct this by comparing it to the original listing.

To enter the program on a VIC is a slightly less straightforward matter. Since the program gives us a monitor on the VIC, the VIC doesn't have its own built in monitor, and the method we've described requires you to enter the monitor, you can see that we have a problem. However, there is a way around this, which involves hand POKEing the program in. This is done as follows:

First of all, remove any extraneous cartridges, memory expansion etc. from your VIC, as we need to type this in on an unexpanded VIC. As Tinymon relocates itself to wherever it finds available RAM, this presents no problems in terms of future use of the program when you've replaced RAM packs etc. Then, type in the Basic programe shown in listing 1.

To get the machine code part in, we have to hand POKE it in. Starting at memory location 4176, POKE the values shown in table 2 into memory. As you can see, the first value is 165. So, we POKE 4176, 165. The next value is 45, so again POKE 4177, 45. Continue this until the very final value (0) which, if you've counted correctly, belongs in memory location 5139: in other words POKE 5139, 0..

Before doing anything else, save this program onto tape, with the normal SAVE command. Then enter and run the following short checksum program:-

```
1 FORI=4176TO5140:S=S+PEEK(I):A=A+1;I-
FA=5THENPRINTS;:S=0:A=0
2 NEXT
```

756	780	802	910	886	853	801	784
876	840	835	1383	753	0	1422	589
816	720	584	680	535	576	944	972
1130	845	876	1357	1010	1188	1311	852
898	1109	1125	897	809	1021	1340	1078
1005	1212	905	902	770	1239	762	1133
1388	652	659	629	1072	803	748	150
617	413	1020	1030	1057	818	944	844
705	831	939	1072	639	1033	943	824
1137	970	929	1149	1395	940	654	840
807	926	706	1146	1015	1146	1175	742
563	645	695	860	1064	1042	1235	1202
1355	922	1445	1346	789	1068	1104	1204
975	1306	1339	1169	1168	1210	1340	1204
972	522	460	520	591	942	1010	1079
960							

The list of values shown in table 3 should be displayed on your screen: this works in precisely the same way as the checksum program explained earlier. If one of the numbers disagrees, you just find out where it is and correct it (but by POKEing this time, not entering the monitor. Obviously, if the program contains a mistake, you're not going to be able to enter the monitor!)

After doing all that you deserve a cup of coffee. Go and make yourself a drink, and then have fun exploring the inner working of the VIC!

83	0	390	396	641	513	607	478
609	477	467	510	430	662	497	502
611	550	1002	460	348	0	484	938
421	465	519	346	377	476	238	547
370	398	165	660	715	487	729	633
712	204	565	962	865	614	535	819
816	495	599	727	662	727	654	615
481	631	562	651	402	915	1018	485
619	713	681	815	368	683	420	495
855	774	276	633	788	857	638	353
412	500	344	551	585	453	567	499
96	101	369	323	299	791	677	574
813	539	479	530	458	521	456	572
461	606	703	677	395	317	948	407
648	430	495	694	831	575	427	703
652	953	685	631	563	401	451	642
651	409	672	167	570	625	946	560
447	951	916	481	268	491	310	407
403	554	598	479	768	724	556	814
923	514	919	714	596	821	1044	739
414	629	574	649	708	919	539	708
812	708	895	704	739	991	616	727
911	713	920	488	564	330	243	329
323	354	358	546	727	501	733	600
450							

PAMPER YOUR PET WITH Codewriter©

Write your data base applications programs in a fraction of the time usually required to do so.

Microsystems introduce to PET owners the CODEWRITER, a superb program generator for the 8000 series PET with dual disc drive unit.

Screen layout, data entry validation, screen display of user-defined error messages, screen calculations, searching by any field - all are child's play to CODEWRITER 1. CODEWRITER DISC 2 provides printed reports and menu generators.

Codewriter 1 - £125

Codewriter Disc 2 - £65

Dealer enquiries welcome.

Pamper your PET promptly, write or'phone
Microsystems Ltd.,
Summerfield House,
Vale, Guernsey, C.I.
Tel. (0481) 47377.

"INSTANT ROM"

'Instant Rom' ROM/EPROM EMULATORS contain CMOS RAM with internal battery backup. When the power is switched off, data is retained for up to 10 years.

In the PET, a 4K INSTANT ROM can be fitted in the $9000 or $A000 socket. Machine-code (and Basic) programs can be stored here, and are available at switch-on.

INSTANT ROM saves time. It can be used for long periods; when the program is finally 'bug-free', an EPROM can be programmed.

4K INSTANT ROM
(ROM socket replacement) £56.00
2K INSTANT ROM
(character generator replacement) £39.00
Adaptor GA1 (essential for PET users) £6.00

"G-ROM E"

G-ROM E is a 4K EPROM which will Auto-run, at switch-on, any Basic or Machine-Code program stored in INSTANT ROM. Basic programs can be stored with a few quick key-strokes. No skill is needed. Programs can now be run without a tape or disk unit, and can be changed without cost to the user.

Diagnostic aids and PETCLOCK programs are included. For example, a 'cold' reset no longer destroys the contents of RAM.

G-ROM E (specify type of PET) £25.00

"PETCLOCK"

Clock-Calendar Type GCC1 plugs into the User Port of any PET, and gives date and time using the program provided. No wiring or external power supply is required.

Accuracy is maintained when the PET is switched off. A lithium battery is used; it needs no recharging, and has a typical life of 10 years.

Format: Time 23:59:59; Date to 31:12:99 + 0-6 (day of week).
Software (tape or disc): UK and US format Basic programs.
Relocatable Machine-code programs. G-ROM E returns time and date in TX$, DX$, DY$ and DZ$.

GCC 1 .... £62.00

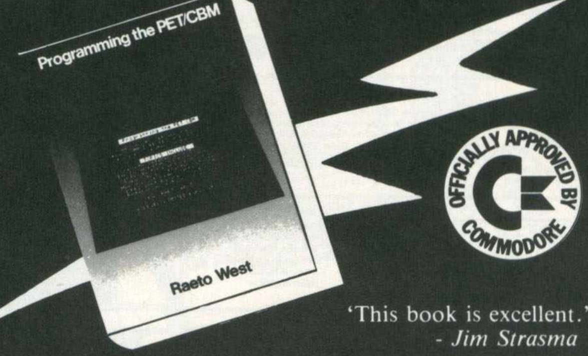
Postage (£1.00) and VAT are extra. Leaflets are available.

"INSTANT ROM" and "PETCLOCK" are COMMODORE APPROVED PRODUCTS.

GREENWICH INSTRUMENTS LIMITED,
22 Bardsley Lane, Greenwich, London SE10 9RF, UK.
Telephone: 01-853 0868. Telex: 896691 Attn. GIL.

NEW

Programming the PET/CBM



'This book is excellent.'
- Jim Strasma

'Unquestionably the most accurate and comprehensive reference I have seen to date.'
- Jim Butterfield

Bestseller — comprehensive teaching and reference book on all software aspects of Commodores 2000, 3000, 4000 and 8000 microcomputers and peripherals.

Many programs, charts and diagrams. 17 chapters, appendices, and index. iv + 504 pages. 19 x 26 x 2½cm. Paperback. ISBN 0 9507650 0 7. Price in UK and Europe £14.90 each (incl. post and heavy-duty packing). LEVEL LTD., PO Box 438, Hampstead, London NW3 1BH. Tel: 01-794 9848.

Cut out or copy coupon, or write to:
LEVEL LTD., PO Box 438, Hampstead, London NW3 1BH.

Send ...... copy/ies of *Programming the PET/CBM* at £14.90 (post free)

I enclose cheque/P.O. for £...........or official order.
NAME ....
ADDRESS ....
....

Fast Service — same day despatch

CC582

INDEX TO ADVERTISERS

Computer Supermarket

Big name hardware at cash-and-carry prices - and with service you'll find hard to match

SHARP, COMMODORE, TEXAS, RICOH, ATARI and TANDATA EQUIPMENT
Fully tested before despatch, or collection complete with instruction manuals, tapes, and fitted with 13 amp plugs.

COMMODORE EQUIPMENT

Model	User Ram	exc VAT	inc VAT
4016	12" 40 Col. 16K Mem	**445.00**	511.75
4032	12" 40 Col. 32K Mem	**560.00**	644.00
8032	12" 80 Col. 32K Mem	**699.00**	803.85
8096	12" 80 Col. 96K Mem	**1040.00**	1196.00
SUPERPET	Micromainframe	**1300.00**	1495.00
2031	121K Disk	**350.00**	402.50
4040	347K Disk	**560.00**	644.00
8050	1M Byte Disk	**755.00**	868.25
4022	Printer	**350.00**	402.50
8023	High Speed Printer	**785.00**	902.75
PET/IEEE	Cable	**18.00**	20.70
IEEE/IEEE	Cable	**20.00**	23.00
VIC 20	Personal Computer	**173.90**	199.99
VIC/C2N	Cassette	**36.00**	41.40
VIC 1011A	RS232 Int	**28.50**	32.78
VIC 1110	8K RAM Cartridge	**36.00**	41.40
VIC 1111	16K RAM Cartridge	**60.00**	69.00
VIC 1112	IEEE Int	**44.00**	50.60
VIC 1210	3K RAM Cartridge	**25.00**	28.75
VIC 1211M	3K RAM (Hi-Res) Cart	**28.00**	32.20
VIC 1212	Programmers Aid	**28.00**	32.20
VIC 1213	Machine Code Mon	**28.00**	32.20
VIC 1515	Matrix Printer	**186.96**	215.00
VIC 1540	Single Disk Drive	**344.35**	396.00
VIC Joystick		**6.52**	7.50
VIC Paddle (Pair)		**11.00**	12.65
VIC Expansion Unit (Arfon)		**78.00**	89.70
Lid for above expansion unit (Arfon)		**6.95**	7.99

SHARP EQUIPMENT

MZ80A	48K Ram	**434.78**	499.99
MZ80B	64K Ram	**950.00**	1092.50
MZ80FI	Floppy Disc I/O Card	**100.00**	115.00
MZ80MDB	Master Diskette & Manual	**31.00**	35.65
MZ80F15	Cable	**9.00**	10.35
MZ80FD	Disc Drive	**589.00**	677.35
MZ80EU	Expansion Unit	**50.00**	57.50
MZ80P5	Matrix Printer	**415.00**	477.25

RICOH

RP1600	Letter Qual. Printer IEEE	**1200.00**	1380.00
RP1600S	Letter Qual. Printer Cent.	**1300.00**	1495.00

TEXAS EQUIPMENT

TI–99/4A	16K RAM	**260.00**	299.00

Full range of peripherals available

ATARI EQUIPMENT

Atari 400 16K RAM Computer	**260.00**	299.00
Atari 800 16K RAM Computer	**456.52**	525.00

Full range of peripherals available

TANDATA EQUIPMENT

Micro Tantel	**152.17**	175.00
Alpha Tantel	**182.61**	210.00

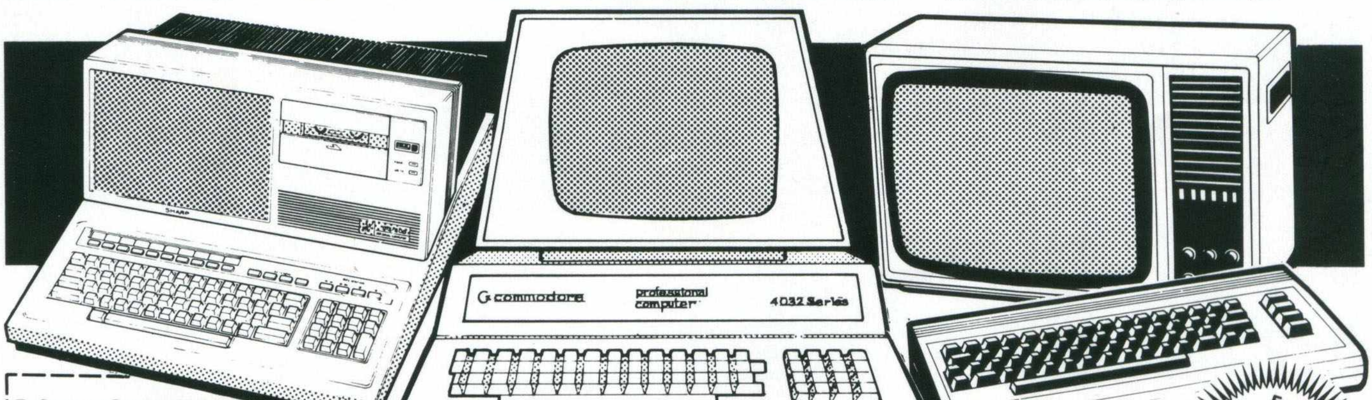
Full colour output. Connects to any TV. Full British Telecom approval. Requires British Telecom 96A jack-plug. Gives access to massive home computer base information from Mortgages to Theatres, Stocks to Holidays.
Telephone us for further information on ease of installation.

Prices are valid only for the cover date month of this magazine

Credit Facilities Available. Ring or write for full details.

Special Price List Available for bonafide Government and Educational establishments.

All orders will be acknowledged by return of post.



Free intro to BASIC part I with every VIC purchased

To Computer Supermarket Ltd.,
3rd Floor, Douglas House,
Queens Square, Corby, Northamptonshire.

Please send me

Model No.	Item	Price	Shipment	Total	Info only

I enclose my cheque for £ ______
Or debit my Access/Barclaycard/
Diners Card/American Express No. ______

(Cardholders may telephone orders to 05363 61587/8)
Signature ______
Name ______
Address ______

(BLOCK CAPITALS PLEASE)

Your remittance should be made payable to Computer Supermarket Reader's Account, and shall remain your money until the goods have been despatched to you at the address specified.
All goods offered are subject to Computer Supermarket conditions of sale, copies available on request. Reg. in England No. 2646589.
Prestel subscribers may order through the Prestel service, Directory No. 400400.

CC

Insured shipment arranged anywhere in UK for an additional £14.37 (inc. VAT). VIC, Atari and Texas shipped by insured post F.O.C.

Approved Distributor for Commodore, Sharp, Atari and Texas.
All goods sold with full manufacturer's warranty and subject to conditions of sale (available on request).
ALL MACHINES ARE FULL UK STANDARD.

COMPUTER SUPERMARKET

COMPUTER SUPERMARKET LTD
3rd Floor, Douglas House, Queens Square, Corby, Northamptonshire.
Telephone 05363 61587/8 and 62571 Telex COMPSU 341543/4 Prestel No. 400400