commodore

COMPUTING

international

July 1982 £1.00

The independent magazine for Commodore computer users

SEC South East Computers

Service – at Supermarket Prices!

£500

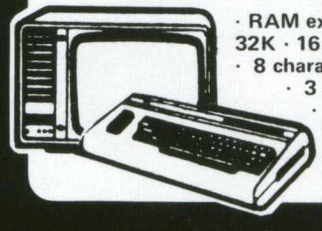
COMMODORE
PET 16K 4016

COMPLETE RANGE OF COMMODORE SOFTWARE – AND PET EQUIPMENT

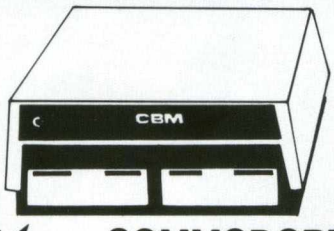
VIC 20

£185

COLOUR COMPUTER



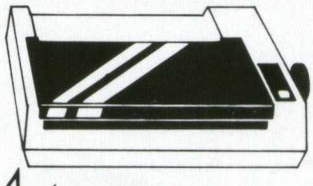
- RAM expandable to 32K
- 16 screen colours
- 8 character colours
- 3 tone generators
- High resolution capacity



£600

COMMODORE
DUAL DISK
DRIVE 4040

EVEN MORE SPECIAL OFFERS ON COMPUTERS, PERIPHERALS SOFTWARE & MEDIA
RING HASTINGS (0424) 426844 FOR DETAILS



£400

COMMODORE
PRINTER
4022P

Prices inclusive of VAT and FREE delivery

SEC *The Complete Computer Service!*

15 CASTLE STREET, HASTINGS, EAST SUSSEX TN34 3DY

Contents

Editorial

Editor
Pete Gerrard

Advertising Manager
David Lake: tel 01-839 2846

Advertising Executive
Peter Chandler: tel 01-839 1881

Editorial Assistant
Fiona McCormick

Production
Three's Company

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.

Typesetting by Centrepoint Typesetters Ltd, London
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.

Speculation has been abounding in the computer press over the latest wave of new hardware to appear, and how the traditional market leaders would react. In particular, the first euphoric greeting for Clive Sinclair's Spectrum revolved around 'well that's it, no-one can possibly match that?'.

There was a period of about two weeks when no-one quite knew what was going on. As far as Commodore were concerned, this reached particularly ludicrous limits as different statements were being issued almost daily from a large variety of sources, each of which were subsequently denied and quashed by the statement that came out the next day.

Well, it now seems that Commodore (and indeed the rest of the industry) has calmed down a little, and now that the dust has settled we can take a look at what is really going on.

The first point that needs to be made is that the Sinclair is not as good as everyone originally thought it was. There is no denying that it IS a good machine, but with a number of limitations. Consequently, the 'big boys' have all breathed collective sighs of relief, and gone back to (almost) original marketing ideas.

It would now appear that all the originally planned Commodore machines will be coming out, and possibly sooner than expected, although of course time alone will tell on that one. All the original specifications are being matched, and it is good to know that the company is not making any hurried, rash decisions. Delivery dates will have to be speeded up, and prices will have to be dropped on the bottom end of the range machines, that's for sure. I think Commodore can manage that.

At the top end of the market we've seen nothing around, either existing in fact or rumour, that can match what is coming out of C.B.M. this year. Let's just hope that nothing goes wrong.

PETs that you can program by talking to them? See you next month . . .

Letters

Dear Sirs,

With lots of mucho grovellings, is there information of ANY sort of information available for the CBM 64 machine? I am trying to assemble as much information as possible about this machine. A colleague is currently involved in a CAL project, the leader of which is interested in using BBC computer equipment (UGH!). We are therefore combining our efforts to redirect the interests of this poor demented fellow and the '64 could prove suitable.

Yours sincerely
R.P.E.H. Mitchell
Bath

Dear Mr. Mitchell,

We did give a mention to the Commodore 64 in last months issue, but in case you missed that, here's a brief rundown on what we know so far. It is a 64K machine, with 40K of that being directly available in Basic, the rest has to be accessed by machine code. Graphics resolution is 320 by 200, and the high resolution mode revolves around characters known as sprites, which are 24 by 21 pixels each. Up to 256 of these can be displayed simultaneously on the screen. 16 different colours can also be shown at any one time. With full synthesiser capabilities as well, this is an impressive machine. Of course, the crucial questions are price and delivery. Well, price is expected to be around 500 pounds, and delivery round about January of next year, but I would suggest to Commodore that they think again about both of those.

Dear Sirs,

I'm writing to enquire if anyone has come across a routine for (simply) changing the device number of a disk drive. The reason for asking is that a fellow Pet enthusiast has an 8050, and I have a 3040, so exchanging of programs is rather redious. Can you offer a solution?

Yours sincerely
S.M. Bennet
Bristol

Dear Mr. Bennet,

We certainly can! The following three lines of code, either typed in direct from the keyboard, or used from within a program, will do the trick:—

```
OPEN 15,OD,15
PRINT#15, "M-W"
CHR$(TY) CHR$(0)
CHR$(2) CHR$(32+ND)
CHR$(64+ND):CLOSE 15
```

Here, OD is the old device number (usually 8), and ND is the new device number (anything in the range 8 to 15). TY depends on the type of disk unit you're playing around with. For a 2040 or a 3040 TY = 50, for a 4040 or an 8050 TY = 12, and for a 2031 or a Vic disk drive, TY = 119.

Dear Sirs,

I write concerning the screen save program by Roger Davis, originally published in CPUCN Volume 2 issue 8. I have only just come across the program! As you are no doubt aware, this program does not work correctly, and so I come to you for a correct listing. I appreciate that this is an inherited problem as far as you are concerned, but would appreciate any assistance you can offer me.

Yours sincerely
Clive Truman
London NW8

Dear Mr. Truman,

Rather than go over old ground, we'll turn this one over to you, the readers, with the chance to win a 10 pound prize! What we're looking for is a machine code routine that can save a screen to disk, and one to pull it back afterwards, from within a program. It should also allow you to design screens. The best entry will be published in a future issue of Commodore Computing, and the winner duly sent the readies.

Dear Sirs,

I am a student at the University of New Brunswick, Canada, and currently tackling a thesis project with the bio-engineering institute. What I'm intending to do is working with the Pet computer, and changing its normal keyboard to an extended one, for use with physically disabled people. I also intend interfacing the Pet to printers other than those normally compatible, and interfacing this to a speech synthesizer for non-vocal disabled people.

The reason I write is that I would be grateful for any information you could send me that you think might be of help.

Thanking you in anticipation

Yours sincerely

Constantinos Pattichis
780 Montgomery Street
Apt 104
Fredericton
NB E3B2Y1
Canada

Dear Mr. Pattichis,

Thank you for an interesting and informative letter (we didn't have room to print it all). We've already sent some material off, but if any of you readers out there know of any information that you think might help Mr. Pattichis, I'm sure he would be extremely grateful to receive this. Thank you for your help.

Dear Sirs,

The enclosed program listings will append program files from a disk onto a program already in the computer. The code can be located anywhere were it's convenient e.g. in the second cassette buffer at $033A : at present it only works on Basic 2 roms. One thing: beware of line numbers, and make sure the first line number of the second program is higher than the last line number of the first program.

The syntax is as follows: SYS X "filename", 8 : where X is the starting address of the routine, and 8 is the device number of the disk unit.

Yours sincerely
Mike McLean
London E10

Dear Mr. McLean,

Thanks for the routine, which we list below. If anyone comes up with a Basic 4 version, let us know, and we'll include it in a future edition of the magazine.

```
ADDR.   CODE        MNEMONICS          REMARKS

033A    A9 00       LDA #$00
033C    85 9D       STA $9D            SET 'LOAD' FLAG
033E    20 3E F4    JSR $F43E          GET PARAMETERS FOR A LOAD
0341    A9 60       LDA #$60
0343    85 D3       STA $D3            SEC. ADDR. WITH BITS 5,6 SET
0345    A4 D1       LDY $D1            LENGTH OF FILE NAME
0347    D0 03       BNE $034C
0349    4C 03 CE    JMP $CE03          'SYNTAX ERROR' IF NO FILENAME
034C    20 0A F4    JSR $F40A          PRINT 'SEARCHING FOR' FILENAME
034F    20 66 F4    JSR $F466          SEND FILENAME TO IEEE BUS AND
                                       UNLISTEN BUS
0352    20 B6 F0    JSR $F0B6          SET DISK UNIT AS TALKER
0355    A5 D3       LDA $D3            SEC. ADDR.
0357    20 28 F1    JSR $F128          SEND SEC. ADDR.
035A    20 8C F1    JSR $F18C          INPUT START ADDR. FROM DISK
035D    20 8C F1    JSR $F18C          AND DISCARD IT
0360    38          SEC
0361    A5 2A       LDA $2A            SET POINTERS FOR START ADDR.
0363    E9 02       SBC #$02           OF LOAD TO END OF PROGRAM
0365    85 FB       STA $FB            ALREADY PRESENT
0367    A5 2B       LDA $2B
0369    E9 00       SBC #$00
036B    85 FC       STA $FC
036D    20 52 F3    JSR $F352          LOAD PROGRAM FROM DISK INTO RAM
                                       UNTALK BUS
0370    4C DD F3    JMP $F3DD          PRINT 'READY': SET END OF BASIC
                                       POINTERS: STXT PTR: LINK PROGRAM
                                       RETURN TO BASIC
```

New Product News

Training Courses

A number of companies have set up training courses on Commodore equipment recently. One such is Comsoft, who write: "We are proposing to run a series of Assembler courses aimed at both the professional and the home computer student. The professional course will be restricted to five students per course, will run for ten evenings, and will cost 180 pounds including VAT. The introductory course is aimed more at the hobbyist, and will be less detailed. This will run for five evenings, the classes will be larger, but the cost will be 49.45 pounds including VAT."

If anyone is interested, they're welcome to contact T. Chisman of Comsoft, at 2c-2d Wake Green Road, Moseley, Birmingham B13 9EZ, or ring 021-449-9151 during office hours.

The second company, with the rather similar name of Compsoft, are opening a training centre for DMS, their own very successful data base program which (amongst other things) works on the Commodore series of machines. Apparently there are over 3000 installations for DMS, so it seems to be a reasonable idea to have training courses based around the program. Originally intended just for the Commodore and CP/M dealer network, the scheme has been extended to include end users as well. These courses cover not only existing users, but potential ones as well, so if you want to see what the product looks like before buying it, now's your chance. Course fees start at 75 pounds, depending on group size, which is normally restricted to twelve people, although they can accommodate up to 30 if necessary. They are held at Compsoft's headquarters in Surrey, and if you're interested your contact there is the lovely Heather Kearlsey, who can be reached at: Great Tangley Manor Farn, Wonersh, Nr. Guildford, Surrey. If you can't be bothered finding a stamp, their 'phone number is Guildford 898545.

Roms and Eproms

JCL (tel. 0892 27454) launched at the PET Show their MK III Eprom Programmer, which provides features to program the most often used Eproms and Eeroms. It can, in fact, program the following devices:— 2508,2708 1K by 8 Eproms (24 pin), 2532,2732 4K by 8 Eproms (24 pin), 2564,2764 by 8K Eproms (28 pin), and the Hitachi HN 48016-P (2716 compatible, so

New Product News

they tell me).

It's also possible to program 16K devices, and using the programmer in conjunction with either JCL's Assember, or the standard Commodore one, allows you to convert any 16K or 32K Pet into a complete development system. Inter-faced to the Pet via the user port, the device is a very useful one.

Other goodies from JCL include their Rom pager, which comes in two versions. One allows up to 8 roms for any one socket, and the other allows up to 4 roms for any two adjacent sockets: both sets of 4 roms can be passed separately. This just plugs in inside the Pet, and thus is transparent to the user of the machine. Selection of the required rom can be done either via the keyboard, or from within a program.

The 8 slot one can also take up to 28K of Basic or machine code which has been stored in Eprom. Price for doing this is available only on application, but the other two come at 45 pounds for the 8 slot, and 47.50 pounds for the 4 by 4 one.

Breaker Breaker

Well, not quite, but all amateur radio operators should be interested in this new product from T.A.L. Computer Division (Tel. 0525 372114). This is a plug in RTTY module, which will work with any of the Commodore range, including the Vic. Specifications given are output/input of 5V TTL to/from the demodulator, baud rates ranging from 45.45 to 150, and the whole thing is programmed in machine code sitting in an eprom.

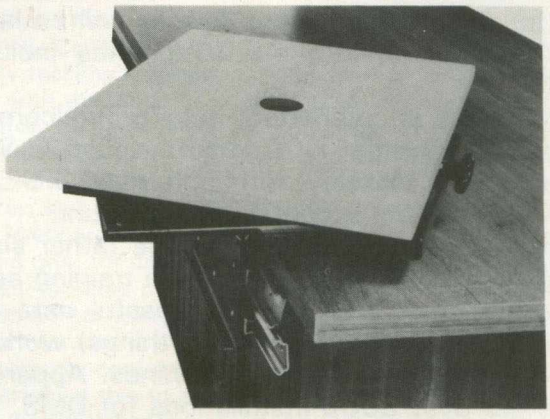
Facilities available on receive include baud rate change, auto format for 40 columns (semi format for the Vic because of its 22 character screen width) and 4/5 user messages. On transmit, we have a number of pre-programmed messages, auto insertion of carriage return, line feed etc., a relay for transmit/receive switching, and the whole device is completely mechanically compatible. Expect this to be available shortly, with a retail price of around 55 pounds.

Okay good buddy, what are you waiting for?

Computer Slide

An interesting little number from Chasmood (tel. 0843 77161), called the 'Computer Slide'. It is aimed at solving the problem of moving Pets (and indeed any sort of monitor, television etc.) around the desk, with the minimum amount of effort and fuss.

The Computer Slide consists of a turntable, approximately 16 inches square, which can be rotated on ball bearings through 360 degrees. Presumably people do want to look at the back of a Pet from time to time. The angle of inclination can also be adjusted, in case you're being dazzled by an infuriating beam of sunlight bouncing off the screen. It can be moved backwards and forwards by 10 centimetres on a separate set of steel ball slides, and finally the complete assembly can be moved sideways by approximately 2 metres when you want to zoom along your desk. Top weight load is approximately 45 kilograms.



The Chasmood Slide

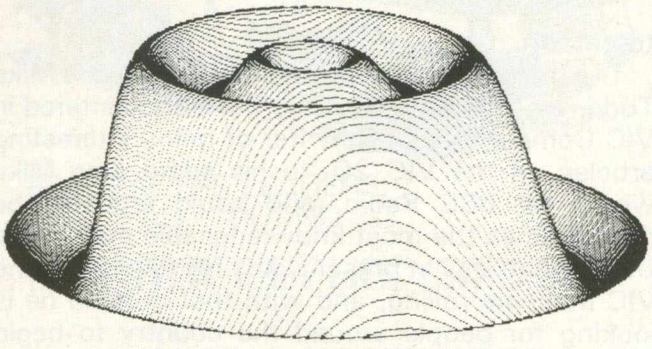
Keeping Your Pet Clean

I'll say one thing for Huberta Kingsbury of Automation Facilities (Tel. 073 522 3012), she is certainly persistent! The number of press releases, editorial copy and so on that I've had recently has been legion. Automation market something called the Petkit, a box full of all kinds of cleaning materials for a complete Pet system: disk head cleaning, cleaning fluid, anti-static and lint-free cloths, and a seemingly endless list of other items. Still, it's all in a good cause, and she does have a valid point: you're not going to get the best out of a system if it's not maintained properly.

More On COMAL

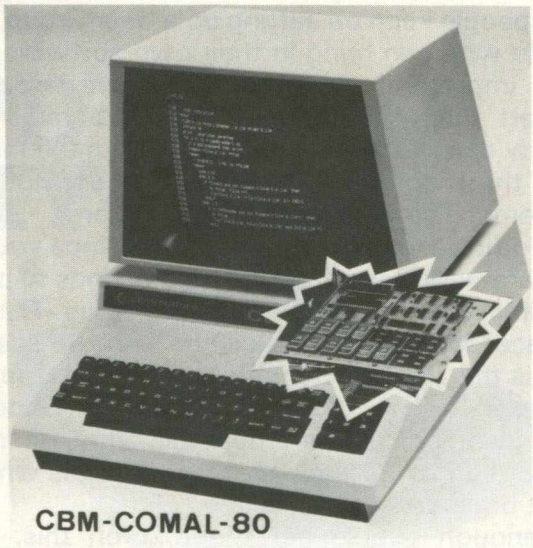
Thanks first of all to Len Lindsay, of the United States COMAL Users Groups, who popped up in town the other day. Even if it did mean going into work at some obscure hour on a Saturday morning, it was a most interesting chat, and we were kept up to date with the latest developments concerning this new language. Thanks also to Nick Green, for turning up and being his usual interesting self, John Collins for providing a 64K board (the old 8032 didn't know what had hit it), Roy Atherton, and someone else whose name I've forgotten (sorry!).

The main point that Len had to make, was that over in Denmark recently they'd just finished a



series of meetings, which finally (apart from one or two minor points) defined the complete and overall structure of COMAL. The language is now in its definitive state.

In this issue we'll be starting a regular spot on COMAL, now that FORTH has had it's run, and we'll be using that to keep you fully up to date with the language. Meanwhile, as a foretaste of what's to come, the diagram shown here shows the kind of high resolution that can be obtained using COMAL and the accompanying high resolution board.



CBM-COMAL-80

The Beebox

This neat, compact unit, which sits underneath your Vic, is designed to give you a 40 column by 25 line display, and increase the amount of available memory from 3K to 32K. It connects up to the expansion socket on the Vic, but has a further socket of its own, so nothing is lost and quite a bit gained.

All this sounds very impressive, but is it as good as the press release claims it to be? Like all press releases, this one is not telling lies, but is quite often leaving a bit of the truth out! One gets the impression that you would have a true 40 × 25 screen area on which you can merrily program impressive graphics for anything from arcade games to word processing packages. Not so.

On connecting up the unit and powering it all up, you have lost the traditional 'window' screen display of the Vic, and the whole screen is there for you to marvel at. Somewhat reminiscent of the old Commodore Pet 4032 display. The old Vic appearance is not all you've lost however. Also gone are the Vic graphics, colour controls, ability to use high resolution graphics, and indeed just about everything you've become familiar with. In their place is the Prestel character set, which in its defence is quite impressive, and a variety of other control characters: for some reason best known to Beelines, actioning these involves positioning them on the screen, thus losing some of the 40 × 25 area.

These control characters include the ability to produce double height characters, flashing characters, and so on. What you can't do is alter the background colouring: you start off with white on a black background, and black it will remain, whatever you try and do about it.

One further feature that I personally would like to see changed is that you cannot revert to the ordinary Vic screen, once the board is wired up. To get back into Vic mode you have to disconnect everything and start again. It would have been nice to be able to swap from one to the other at will. One the plus side, the colour quality looks distinctly better than on a normal Vic, with only a slight shimmer when scrolling through a listing, or simply scrolling off the screen.

So, it does give you an extra amount of memory, and all told is probably fairly cheap for an additional 29K of RAM and a 40 × 25 display area, even if it is only a display area. At 220 pounds it will probably be of most use to the businessman or technician who wants to use a larger screen area (for say stock control, or whatever), but for the average hobbyist I can't honestly see it being of much use.

The Beelines Beebox

Club News

Vic & The User Club

VIC & The User Clubs

It didn't take too long after the first microcomputers started appearing for the very first User Groups in this country to begin springing up. These were usually started up by enthusiastic amateurs, who wanted to pool their resources in order to further their own individual knowledge concerning whatever particular micro they owned.

Given time, these User Groups got onto a much firmer footing, and nowadays the most successful of these is probably the Independent PET Users Group, or as they recently renamed themselves, the Independent Commodore Products Users Group. They have their own chairman, secretary, treasurer and so on, and regularly hold committee meetings in order to determine the future path of ICPUG (as we shall call them from now on).

It soon became apparent that running a nationwide set of groups was more than just one body of people could handle, and so ICPUG 'split up' into a number of regional sub-groups: today there are 20 or so of these regionals, all coming under the ICPUG umbrella of organisation. As a point of information, the contact for any queries, offers of help etc. is Mrs. Eli Pamphlett, at 7 Lower Green, Tewin, Welwyn, Hertfordshire (Tel. Welwyn 7325).

Firstly, what is the purpose of joining, organising, starting, or indeed having anything to do with a VIC Users Group? User Groups exist for many purposes, not least of which is the dissemination of information pooled from many sources: something which an individual working alone cannot possibly hope to do. Many brains working together can more often that not accomplish far more than one lone sole slaving over memory maps, soldering irons and the like.

User Groups also have the ability to organise seminars, attend exhibitions etc., again something that one person alone cannot do.

Thirdly, if, as ICPUG have, you can establish a good working relationship with the manufacturer whose products you're using, you stand a very good chance of being kept abreast of all the latest information regarding that company.

Finally, it is a chance to meet other brethren enthusiasts, and find out for yourself what others have already discovered. The wheel is re-invented many times in the microcomputer industry: there's no reason why you should have to do the same!

So, what are ICPUG doing in terms of the VIC, and how do they hope to proceed in the months to come?

The main man behind the current set-up is Mike Todd, whose name you may have encountered in VIC Computing as the writer of many interesting articles on the VIC 20. In his spare time Mike works for BBC Radio, and could arguably be described as the most knowledgeable man in the U.K. on the VIC at present. It is his role to get the VIC machine rolling, and with that in mind he is looking for people around the country to begin setting up VIC User Groups. As stated earlier, there are twenty PET groups at the moment: if you contact Mrs. Pamphlett at the address given earlier, she'll be able to tell you where they all are.

Initually, ICPUG would like to see VIC people joining existing PET groups: if the demand is sufficient, these can later disassociate themselves from the PET side of the group, but of course keep in with ICPUG and retain all the benefits which that entails. The sort of person you are (technical, software, hardware) doesn't really matter: the important point is that you own, or have access to, a VIC, and are enthusiastic about the machine!

The kind of thing that ICPUG are looking for, from people keen on setting up a User Group, is a) people who can bring in their own software, and more importantly b) people who are willing to convert the vast library of PET programs that already exist to work on the VIC. With the majority of the programs this should present no great problems. The point is that it gets done!

If enthusiasm isn't enough to convince you, ICPUG are willing to offer up to 50 pounds to cover the initial expenses of setting up a group. This will be necessary to cover mailouts and the like. As well as this, ICPUG are in the position to receive a vast amount of information from Commodore, both technical and promotional, and are more than willing to disseminate that to people who go about organising a VIC group.

If enough people get together on this, it is possible that at some future date an IVUG might appear, with it's own committee and so on. However, time alone will tell on this one.

User Groups are important. They fulfil a vital role for both the newcomer and the experienced computer user. As a base for gathering and giving out facts on a particular machine they probably have no equal, as they have the ability to call on a large number of knowledgeable people in a specific area. Joining one, or starting one up, can only be of ultimate benefit to yourself as a computer user. It will take up some of your time, there is no doubt about that, but that time will be dramatically rewarded.

Communications

A significant step forward in the twin worlds of communications and education has been made by the recent announcement of a product called the VicSwitch, only just released by Datalect Limited, of Croydon (tel. 04862 63901/25995). This is a networking system for (surprisingly) the Vic.

The idea of networking is not a new one, and has been exploited by many 'add-on' manufacturers before now. However, no-one has yet produced any such system for the Vic, and Datalect look fairly certain to make a killing with this. So why should networking be of importance to educational users?

Educational Importance

Any educationalist worth his financial salt will always consider the cost of any project before embarking on that project. Schools are not alone in feeling the pinch at present, but they are possibly suffering more than most, and so any monetary commitment must be very carefully examined. This is why I believe that the VicSwitch will be a major success.

Despite the low cost of the Vic itself, the cost of the various peripherals is rather high: a disk drive at 396 pounds and a printer at 230 pounds become expensive items when more than one is required. What is more to the point, the Vic is making considerable inroads into the educational arena, but people may well have been put off by the prohibitive cost of buying complete systems every time.

This is unfortunate because complete systems are more often not only an advantage, but a necessity. Both the disk drive and the printer are extremely useful devices to have, and because of the nature of the majority of educational uses that a set-up is likely to be put to, a disk drive in particular becomes distinctly desirable. There can be nothing worse than losing a child's attention because he has to wait 5 minutes for a program to load from tape. It is also preferable to have hard copy at the end of the day as some measure of achievement. It would be a shame if Commodore were to lose out because people were put off by price.

Cheaper Access

No more! The VicSwitch provides an inexpensive way of allowing many users access to a single disk drive and printer. Cost of the mother unit is 50 pounds, and the requisite number of daugher units at 30 pounds each: definitely cheaper than 626 pounds for each combined set of disk drive and printer.

At the centre of the network is a unit which connects up to the back of the Vic, and from there on all the other Vics are linked up in a loop-type system. Once everything is connected up, each Vic can quite happily carry on processing information, or displaying some educational program on the screen, and get hold of the one disk drive or printer as required. If one user is already loading or saving a program (or file for that matter), the next user simply waits his or her turn until the unit is free, and then gains access himself.

Further advantages can be gained from the fact that each separate Vic can be up to 50 metres from its nearest neighbouring Vic, which allows for a lot of flexibility within fairly large schools.

Other Uses

Of course, it is not just schools that are likely to benefit from this system. Factories for instance, could have a number of Vics monitoring different processes going on, and then all of them could have access to a common main program on disk for continuous updating of information. Printout from each separate Vic could also be easily, and cheaply achieved.

I'm sure there are many other uses to which this could be put. The reason for including it in this section of the magazine should be fairly obvious: over the last two months we've concentrated on Petnet and Telecomputing, both forms of networking using Pets. It's nice to see the smaller end of the market getting a share of the action, and thus opening up this most useful field for people with more limited resources.

'A Bit Excessive, This One, Eh, Jenkins?'

Future Feedback

If you belong in an establishment that is using VicSwitch, please write and let us know, and in what sort of environment you're using the product. Both ourselves and the suppliers of the package would be extremely interested in any feedback.

FREE PET/CBM COMAL

"The excitement in Europe (over COMAL) seems to be growing by the hour and we look forward to America being able to share in the good fortune of having an easy-to-use, structured, planning language at last."

The power of PASCAL and the ease of BASIC can now be yours with Commodore COMAL, a new programming language from DENMARK. It is being distributed in the USA by the COMAL USERS GROUP. To find out more about COMAL and how you can get a free disk copy of Commodore COMAL, send a large self-addressed stamped (35 cents) envelope to:

COMAL USERS GROUP
5501 GROVELAND TER., MADISON, WI 53716.
Outside USA please add $2.00 for airmal and handling.
*PET & CBM are trademarks of Commodore Business Machines.

Your move

Commodore CBM systems are dependable, efficient, versatile, easy to maintain. That's why so many companies, big and small make their move through

ALPHA Business Systems

Electron House
Industrial Estate
Church Street
Ware SG12 9ES

Telephone Ware (0920) 68926/7

SERVICE MS TO INDUSTRY

COMMODORE SYSTEMS DISTRIBUTORS

- ALL BUSINESS SYSTEMS INCLUDE 12 MONTHS WARRANTY
- COMPLETE RANGE OF CBM EQUIPMENT IN STOCK, INCLUDING 8096, VIC 8023 PRINTERS, ETC.
- LEASING AVAILABLE
- NATIONAL MAINTENANCE CONTRACTS AVAILABLE
- SPECIALIST SOFTWARE INCLUDES
  - WORK STUDY
  - PLANNED MAINTENANCE
  - BONUS CALCULATION
  - PRODUCTION CONTROL
- STANDARD SOFTWARE IN STOCK INCLUDING:
  - STOCK CONTROL
  - WORD PROCESSING
  - D.M.S.
  - INTEGRATED AND INDIVIDUAL LEDGERS
  - SILICON OFFICE
- INSTITUTIONAL AND GOVERNMENT DEPT. PURCHASERS: SEND FOR OUR PRICE LIST
- CASH PURCHASERS PRODUCE THIS ADVERT. AND WE WILL PAY YOUR V.A.T.

DEMONSTRATIONS BY ARRANGEMENT

HEALEY MANAGEMENT

Head Office:
442/6 London Fruit Exchange
Spitalfields
London E1
(01) 247 2858

Also At:
Phoenix House
1 New Street
Worcester
(0905) 611545

COMAL

COMAL — Commands which have BASIC equivalents

by Brian Grainger

In the following article I will show on the left side of the page some COMAL statements. On the right side of the page I will show how the same result is obtained in BASIC. In this way you should be able to see how to convert BASIC programs into COMAL and also get some idea of the use of some of the COMAL commands. I shall show the COMAL statements as they would appear on the screen should they be listed. As you will see in the article on features unique to COMAL, it is not necessary to type all you see. It still works if you do though.

```
COMAL                          BASIC
0010 CASE A OF                 10 ON A GOTO 30,40
0020 WHEN 1                    20 PRINT"A IS OUT OF RANGE":
0030  PRINT"A IS ODD"               GOTO 50
0040 WHEN 2                    30 PRINT"A IS ODD":GOTO 50
0050  PRINT"A IS EVEN"         40 PRINT"A IS EVEN"
0060 OTHERWISE                 50 ,,,,,,
0070  PRINT"A IS OUT OF RANGE"
0080 ENDCASE
-------------------------------------------------------------
A:=1 ; B:=2                    A=1:B=2
CHAIN "FILENAME"               LOAD"0:FILENAME",8:RUN
CLOSE                          DCLOSE
CON                            CONT
DEL 100                        100 <RETURN>
DIM MATRIX(0:100,0:10)         DIM MATRIX(100,10)
A DIV B                        INT(A/B)
-------------------------------------------------------------
0010 EXEC SUBPROGRAM           10 GOSUB 100
;                              ;
0100 PROC SUBPROGRAM           100 ,,,,
;                              ;
0200 ENDPROC SUBPROGRAM        200 RETURN
-------------------------------------------------------------
0100 PROC FNR(X)               100 DEF FNR(X)=INT(X*100+.5)
0110 FNR:=INT(X*100+.5)
0120 ENDPROC FNR
NO:=FALSE                      NO=0
FOR I=1 TO 10 DO PRINT I       FOR I=1 TO 10:PRINT I:NEXT
-------------------------------------------------------------


0010 FOR I=1 TO 10 DO          10 FOR I=1 TO 10
0020  PRINT I                  20 PRINT I
0030  PRINT I*2                30 PRINT I*2
0040 NEXT I                    40 NEXT
-------------------------------------------------------------
0010 GOTO FINISH               10 GOTO 110
;                              ;
0100 FINISH:                   ;
0110 PRINT"END OF PROGRAM"     110 PRINT"END OF PROGRAM"
-------------------------------------------------------------
0010 IF A=B THEN               10 IFA=BTHENPRINT"A EQUALS B"
0020 PRINT"A EQUALS B"               :GOTO 40
0030 ELIF A>B THEN             20 IFA>BTHENPRINT"A GREATER
0040  PRINT"A GREATER THAN B"       THAN B":GOTO 40
0050 ELSE                      30 PRINT"A LESS THAN B"
0060  PRINT"A LESS THAN B"
0070 ENDIF                     40 ;;;;;
-------------------------------------------------------------
INPUT"WHAT IS NUMBER? ":NO     INPUT"WHAT IS NUMBER";NO
LOAD"FILENAME"                 LOAD"0:FILENAME",8
A MOD B                        A-INT(A/B)*B
OPEN 2,"FILENAME",READ         OPEN2,8,8,"0:FILENAME.SEQ,R"
OPEN 2,"FILENAME",WRITE        OPEN2,8,8,"0:FILENAME.SEQ,W"
OPEN 2,"FILENAME",APPEND       APPEND#2,"FILENAME"
OPEN 2,"FILENAME",RANDOM 100   DOPEN#2,"FILENAME",L100
ORD("A")                       ASC("A")
SELECT OUTPUT "LP"             OPEN1,4:CMD1
followed by                    followed by
SELECT OUTPUT "DS"             PRINT#1:CLOSE1
PRINT A$;B$;C$                 PRINT A$;" ";B$;" ";C$
PRINT A$,B$                    PRINT A$;B$
-------------------------------------------------------------
0010 ZONE:=10                  PRINT A,B
0020 PRINT A,B
-------------------------------------------------------------
0010 REPEAT                    10 I=I+1
0020  I:=I+1                   20 PRINT I
0030  PRINT I                  30 IF I<>10 GOTO 10
0040 UNTIL I=10
A=RND(X,Y)                     A=X+INT((Y-X+1)*RND(0))
A=RND(0)                       A=RND(0)
SAVE"FILENAME"                 SAVE"0:FILENAME",8
SIZE                           PRINT FRE(0)
STATUS                         OPEN15,8,15:INPUT#15,A$,B$,
                               C$,D$:PRINTA$;B$;C$;D$
                               :CLOSE15
PRINT STATUS(2)                (e.g.)INPUT#2,A$:INPUT#15,ER
                               :PRINT ER
NO:=TRUE                       NO=1
-------------------------------------------------------------
0010 WHILE I>0 DO              10 IF I<=0 GOTO 50
0020  NO:=NO+I                 20 NO=NO+I
0030  I:=I-1                   30 I=I-1
0040 ENDWHILE                  40 GOTO 10
0050 PRINT NO                  50 PRINT NO
-------------------------------------------------------------
READ FILE 2:A$                 INPUT#2,A$
WRITE FILE 2:A$                PRINT#2,A$
READ FILE 4,12,3:A$            RECORD#4,12,3:INPUT#4,A$
WRITE FILE 4,12,3:A$           RECORD#4,12,3:PRINT#4,A$
EDIT                           LIST
CAT                            CATALOG
CAT 0                          CATALOG D0
CAT 1                          CATALOG D1
-------------------------------------------------------------
PRINT NAME$(1:N)               PRINT LEFT$(NAME$,N)
PRINT NAME$(M:M+N-1)           PRINT MID$(NAME$,M,N)
PRINT NAME$(END-N+1:END)       PRINT RIGHT$(NAME$,N)
-------------------------------------------------------------
A# (integer variable)          A%
N.B. This uses less storage    N.B. This uses same storage
     than A                         as A
```

Some of the COMAL examples above may look clumsy compared with the BASIC equivalents. This is because I have contrived to present exact equivalents. In practise the COMAL program would be written bearing in mind the use of the COMAL commands, not the use of BASIC equivalents. Some of the COMAL commands shown above have extra uses. See the article on features unique to COMAL.

COMAL — Commands without BASIC equivalents

In this article I shall be looking at the features of COMAL which are not available in BASIC. I will introduce some new commands as well as different forms of commands already discussed. Finally I want to discuss the relationship between what needs to be typed and what COMAL will automatically fill in for itself. ALL COMAL examples will be shown as if they were listed on the screen. Having said that let us get underway with the features unique to COMAL:

1) AUTO is a command which will generate line numbers automatically.

e.g. AUTO generates 10,20,30
AUTO 100 generates 100,110,120
AUTO 100,5 generates 100,105,110

Automatic line numbering is turned off by pressing return to a line number.

2) The CASE command is far more versatile than a replacement for ON . . .GOSUB. The expression to be tested can be a string expression or logical expression as well as being numerical as in BASIC. In addition where BASIC expects the values taken by the expression to be sequential the CASE command will allow ANY values:

```
0010 CASE COLOURS OF
0020 WHEN "RED", "YELLOW", "BLUE"
0030 PRINT COLOURS: "IS A PRIMARY COLOUR"
0040 OTHERWISE
0050 PRINT COLOURS: "IS A MIXED COLOUR"
0060 ENDCASE
```

3) The DEL command can also be used for block deletions. Syntax is identical to the BASIC LIST command:

e.g. DEL 100-250 will delete all lines from 100 to 250 inclusive.

4) Array handling in COMAL is extremely powerful. The DIM statement of COMAL equivalent to that of BASIC looks clumsy but consider that in COMAL the index of an array can range from ANY integer to ANY larger integer:

e.g. DIM FIELD (−3:7) declares a one dimension array with indices from −3 to 7 inclusive.

e.g. DIM NUMBER(6) declares an array with indices 1 to 6 inclusive (there is NOT a 0 element as in BASIC unless declared specifically).

In COMAL the maximum length of a string MUST be defined:

e.g. DIM NAMES of 10 declares a string of maximum length 10 characters.

e.g. DIM NAMES(2:5.3:7) of 10 declares a string two dimensional array of 20 elements, each with maximum length 10 characters.

N.B. ALL array variables MUST be dimensioned. There are no defaults as in BASIC.

5) The ENTER command can be used to merge previously LISTed files into the current program:

e.g. ENTER"FILENAME" will merge FILENAME from disk into the current program. Lines will be overwritten and reordered as necessary:

e.g. ENTER"FILENAME",1 will take the file named FILENAME from cassette and merge it with the current program.

6) EOD — This is a logical variable which is TRUE when the last item in DATA statements is READ.

7) EOF(X) — This is a logical variable which is TRUE when the last item from channel X is taken.

8) EXEC is more than a replacement for GOSUB. Together with the PROC command we have a very powerful feature. Parameters can be passed to a subroutine. Variables used in a subroutine can be global to the whole program (in which case changes made to the variables in the subroutine will be recognised in the main program), or local to the subroutine (in which case the main program will not recognise any parameters used in the subprogram). All parameters to a subprogram are local unless identified as REF parameters. As output parameters must be recognised by the main program they must be defined as REF parameters. For space limitations all arrays must be defined as REF parameters:

```
0010 I:=2; J:=3; D:=5
0020 EXEC MULT(I,J,K)
0030 PRINT K,D
0040 END
0050 PROC MULT(A,B,REF C) CLOSED
0060 D:=A*B
0070 C:=A*A+B*B+2*D
0080 ENDPROC MULT
```

The EXEC command will cause MULT to be executed with A replaced by IB replaced by J and C replaced by K. Because C is a REF parameter the value will be kept by K when the procedure is complete. As the procedure heading has the word CLOSED appended all variables used by the procedure are local and can take the same identity as those elsewhere in the program with no confusion. If the program above is RUN you will find D is printed as 5 not 6 which would be the value if the D in the procedure had any effect. If the word CLOSED had been omitted then D would have been global and D would be printed as 6. The storage used for local variables is dynamic so that on exit from the subroutine the storage is released for other use.

The PROC used as a function is also more powerful than the BASIC DEF FN. In BASIC one can only use 1 line to define the function. In COMAL one can use as many lines as necessary.

9) The IN function, used with strings determines whether one string is contained within another and returns the position of the first character if so. If the string is not found a value of 0 is returned:

```
0005 DIM NAMES OF 9
0010 NAMES:="FREDERICK"
0020 A:=RACK IN NAMES
0030 B:="DAVE" IN NAMES
0040 PRINT A,B
```

The above example will result in A being 6 and B being 0.

10) The INPUT command is bombproof. It does not exit if a return is pressed. It will also accept commas and colons etc. in string input. If a listing of strings are being input, it does mean that each variable must be separated by return .

More next month

Software Review

PetSpeed

There are a number of Basic compilers available for the Commodore range of computers at present. One such is called PetSpeed, from Oxford Computer Systems, distributed at present by Commodore themselves, and presently available for 4000 and 8000 series machines. Before examining the package in some detail, let us first of all answer the question: ''What is a Compiler?''

When you're writing your program in Basic, you are using the built-in Basic interpreter of the Pet to do so. Consequently, when you are satisfied that your program is correct and you RUN it, it is the interpreter that takes care of what is actually happening. It recognizes the type of statement, checks the syntax, does many other things and finally obeys the statemement.

Slow Programs

This as you can readily appreciate, takes a considerable length of time to do, it all has to be repeated each time the statement is met, and so your program will not be running particularly quickly. This is not usually any great hindrance but there are occasions when speed is vital, and you wished you knew a little bit more about machine code than 'it looks like a lot of meaningless figures and numbers'.

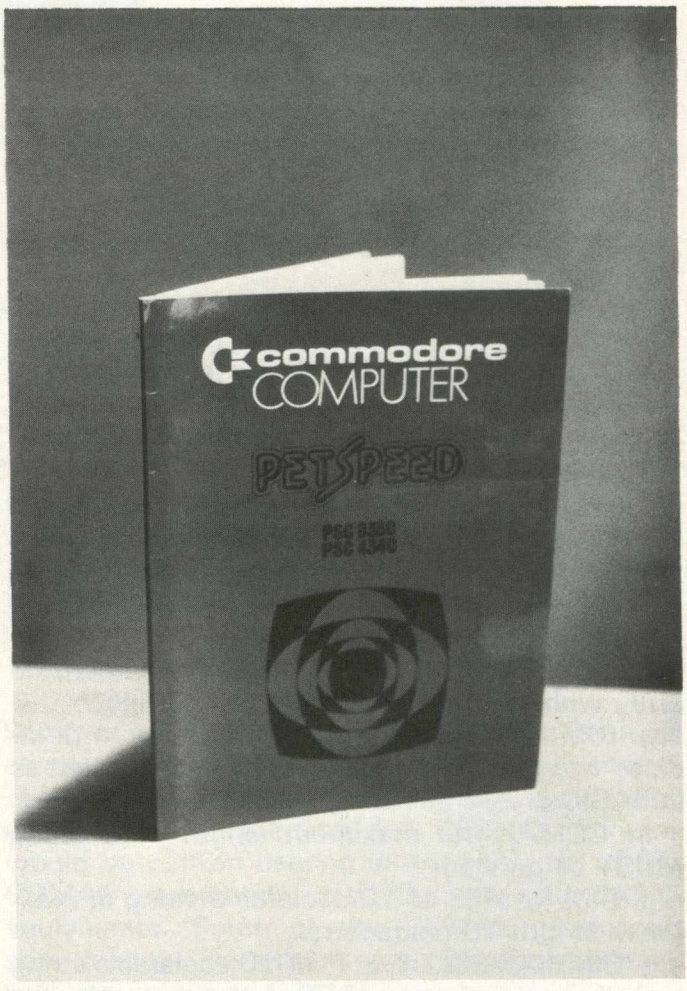
This is where the compiler comes in. When a program is compiled, all it does in executing the program is recognise the statement, check syntax etc. once only, and it is then just the obeying of the statement that has to be done each time. Thus, there is a considerable saving in the amount of time required to execute a program.

In essence then, a compiler takes your Basic program and converts it into machine. code. Purists will howl at that definition, but it will do as a starting block on which to build future arguments. What was once 700 lines of Basic code is reduced to a single line (in the case of PetSpeed this becomes SYS(1061)). Thus the program becomes impossible to edit, and hence you must ensure that the program is fully de-bugged before compilation.

So, if it doesn't convert it into machine code, what does it do?

Compiler inaction

PetSpeed converts your Basic program, over a four pass phase, into what they term Speedcode



The Speedy Pet

(pseudo-code), which is then interpreted at run time. The whole aim of PetSpeed is to make your compiled program run as fast as possible, so let's take a look at it.

PetSpeed Manual

It is sadly inevitable when reviewing something from Commodore themselves, that the accompanying manual must come under fairly close scrutiny. Well, Gail Wellington, have no fear! This manual is good, well organised, well printed, and well presented. It is also short, succint,and to the point. In other words, a lesson for other manual writers.

PetSpeed Program

On acquiring your copy of PetSpeed, the first action to be taken is to insert the now familiar protection device, the dongle, into the cassette

port at the rear of the machine. Secondly, a disk with the program to be compiled must be placed in drive one, and you must ensure that the diskette has at least 500 blocks free, as PetSpeed produces a number of different files in the process of compilation, one of which may be several hundred blocks long.

Placing the PetSpeed disk in drive zero, and pressing shift and run/stop gets the show on the road. All you have then to do, having (as we stressed earlier) checked that your source program is snytactically correct, is type in the name of that source program when requested, and then sit back with a cup of coffee whilst PetSpeed goes about its business.

There are four stages to this process. Pass One is concerned principally with symbol table building, which it sorts in order to give priority to those parts of the code which will be accessed most frequently at run time. This pass also performs a complete analysis of any data statements contained in the source program, converting them in some circumstances to both CBM Ascii (PETSCII!) and binary entries in its own file.

Pass Two performs the main syntax analysis, and it is here that a very detailed version of the original program is built up and stored on drive one. This is the large program mentioned earlier.

Pass Three takes this detailed version, and breaks it down into a more simplified structure. From this structure, the final Speedcode is generated.

Finally, Pass Four arranges all the program segments in memory, and links everything up correctly. When this is completed, the program is saved to drive 1 (with the suffix .gt) and all the other files that have been created (with one exception, for use in error checking) are then scratched.

Compatibility with Pet Basic

There are few limitations which you have to overcome before compiling your program. The three to watch out for are (1) you cannot have RUN in your program e.g. RUN 1000 is not allowed. (2) the same goes for LIST as well. (3) PetSpeed cannot handle dynamic arrays (e.g. those that are defined as DIM A$(N)), so you have to find the highest value of N that is likely to occur, and dimension A$ that way. Thus, if N is never going to exceed 100, your original program must contain the statement DIM A$(100).

PetSpeed does not expect to come across calls to machine code subroutines in a program. These should be removed prior to compilation, but there is a reasonably detailed section of the manual devoted to telling you how to get those machine code parts back in again, and how to pass variables from the Speedcode generated by PetSpeed to your routines and back again.

As if to make up for these limitations, there are a number of enhancements that have been added. User-defined strings, integer FOR loops, options to have long variable names that are actually recognised, and true integer arithmetic (which considerably speeds up program execution).

Programming

Use of PetSpeed allows you to produce more legible Basic programs, as you no longer have to worry about avoiding spaces and REMs, packing a number of statements onto the same line, declare frequently occuring variables at the start of a program, or any of the things normally done to produce swift code. You can now write programs that are readable by people other than yourself (and yourself six months later, come to that). PetSpeed goes all out to make your program run as fast as possible: speed gains of up to 40 times have been recorded.

Summary

PetSpeed is a most useful addition to anyone's software library, and at 240 pounds it should be available from any of the Commodore dealers.

You don't have to learn anything (other than how to type) to be able to use the package, which is a great advantage. It allows you to improve your own Basic programming, and then when you're happy that everything is error free and running correctly, produce a version of your program that is significantly faster in execution. For instance, we compiled the well known game Adventure: as a lengthy Basic program, involving a lot of string handling, we were interested to see how the uncompiled and compiled versions measured up to each other. The difference was, literally at first, staggering. I won't tell you what one person said when he saw the compiled version running, but I gathered that he was impressed.

And so were we all. Commodore taking a program on board is usually the kis of death, but PetSpeed has survived intact. Highly recommended.

The PET Single Disk Drive

The 2031 disk unit is supplied with a mains lead (complete with 13A plug!) but the appropriate signal lead necessary to connect the unit to the PET has to be purchased separately. This should be a PET to IEEE lead if no other devices are connected to the IEEE port or an IEEE lead if a printer, for example, is already connected to the PET. A demonstration disk with a performance test and a few utilities and example programs is supplied. In order to run the performance test a blank disk must also be used so this is something which should not be forgotten when buying your disk unit (Why not persuade the dealer to give you one?).

The documentation supplied included the 2031 User Manual (plus errata sheets), a note on how to change the device number, an alternative (presumably working) listing of the example program in the manual, an errata sheet for the PET/CBM Personal Computer Guide and a test certificate (which described the unit as a 4040!).

The manual covers connection to the PET and power up in some detail with cautionary notes on when power should be switched on. It omits to mention that the unit was probably left switched on at the factory and will power up as soon as you plug it in to the mains. The top of the switch on the rear panel has to be pushed in to switch off. I have since noticed that there is an OFF label but it is on the wrong side of the switch! One feature which I think would improve the unit is an indication that it is switched on. The light on the front is only on when the disk is being accessed or when an error has been detected (in which case it flashes). I am sure a suitably placed resistor could be used to make the light glow dimly while power is applied.

There are step by step instructions for running the performance test and this presentd no problem. The time quoted to run this test (7 minutes) is one of the clues that the manual is a hastily edited version of a 4040 manual. The test actually takes only two and half minutes.

Once the unit has passed the performance test it is ready for normal use. The next set of commands in the manual, described as disk maintenance commands, permit the user to format a blank disk, read the disk directory and copy, rename and scratch (delete) files.

The NEW command (not be confused with NEW in BASIC) must be used to format a blank disk before that disk can be used. The space on the disk is organised into a number of tracks and sectors by recording track and sector numbers together with synchronising information on to the disk. Each sector can hold 256 bytes of data and the number of sectors per track is as shown in Table 1. By reducing the number of sectors on tracks nearer the centre of the disk, excessive recording density is avoided.

Table 1

Track	Sector range
1 to 17	0 to 20
18 to 24	0 to 18
25 to 30	0 to 17
31 to 35	0 to 16

Track 35 is nearest centre of disk

The total number of sectors or blocks is 683. A map of these, called the Block Availability Map (BAM), is maintained on the disk to show which sectors are free and which have data stored in them. This map is used by the disk operating system (the program which controls the disk unit) to rapidly locate enough space to store new information. A disk name and two character identifier specified by the user are also written on to the disk by the NEW command.

The disk directory which lists all the files on the disk with their sizes and types can be read into the PET and displayed on the screen. The disk name and identifier, the disk operating system (DOS) version and the amount of free space available to the user are also indicated.

After successfully formatting a new disk, I skipped on a few pages in the manual to the save and load commands. These are the familiar commands used with the tape drive but do not benefit from the defaults; the unit number (normally 8 for disk) must be specified after the file name. It is not necessary to include the drive number because the 2031 has only one drive. Using these commands I transferred a number of my more popular programs from tape to disk. The load from disk was not as fast as I had expected, but 10 seconds to load 8 k is much better than

waiting 2 minutes for a tape to load. My daughter was most impressed at the speed with which she could load her favourite programs and soon learned the new command formats; her little brother just tried to stuff his sock into the disk unit's 'mouth'.

Having learned the commands necessary to use the disk unit, I turned my attention to the programs on the demonstration disk. The following programs are included:

DOS SUPPORT	A program which can be loaded in to the top of PET memory to provide 'single key' equivalents of disk commands.
PERFORMANCE TEST	To check that the disk unit is functioning correctly.
GRAPHIC BAM VIEW	To plot on the PET screen the disk Block Availability Map.
DISPLAY T & S	To display the contents of a user specified track and sector.
SEQ EXMPL-UNIV	To demonstrate sequential files.
SEQ EXMPL-BAS4	As above but with the more compact BASIC 4 commands.
PRINTER DEMO	To exercise the Commodore printer.
RANDOMEXAMPLE	To demonstrate relative files.

The program to display the BAM provided some interesting information about the disk unit. The BAM on a newly formatted disk shows that only sectors 0 and 1 of track 18 are occupied. The manual indicates the format of these sectors which hold the BAM and the first block of the directory. Note that the number of blocks free indicated by this program includes the track 18 blocks which are reserved for further directory entries. As files are written on to the disk, tracks nearest to track 18 are used first. By storing the directory on the centre track and storing files as near as possible to this track, the time to access a file after finding its location in the directory is minimised.

Another sign of intelligence in the unit's control program is revealed when the 9th file has been written to the disk and a second directory block generated. The new block is not, as may have been expected, stored in track 18 sector 2 but in 18,4. This is the sector which will be approaching the read/write head when the first directory block has been read and searched and the required entry not found. This technique of spacing the blocks allows a full directory to be searched in only 3 revolutions of the disk rather than the 20 revolutions which would be required if consecutive blocks were used.

The DISPLAY T & S program required some corrections before it would work properly. It must be an edited 4040 program which was not adequately tested on a 2031.

The program will display or print the contents of a single sector in hexadecimal and character forms. It then directs the user to the next sector in the file. If this is not required, new track and sector numbers can be entered. End of file is not recognised and it is necessary to specify an invalid track number to exit from the program and terminate disk activity correctly. In spite of these shortcomings, it is a useful program which shows clearly how data is structured on the disk.

The sequential file example programs show how data can be transferred to and from a disk file. They move data from DATA statements in the program to a sequential file on disk then read it back and display it. The programs are generously commented and illustrate the recommended practice of checking for errors after each disk operation. Because the BASIC 4 disk commands are not recognised by BASIC 3 it LISTs them as some other command which is confusing when you look at a BASIC 4 program using BASIC 3. DOPEN and DCLOSE appear as FOR and NEXT in the example program.

I did not try the PRINTER DEMO program because I do not have a Commodore printer. My colleague and fellow 2031 owner Ron Cason ran it and produced four or five pages of assorted examples of the capability of the printer. The RANDOMEXAMPLE is too big for an 8k PET so I did not try that either.

This concludes the account of my initial work with the 2031 disk unit. There are several other commands to study which I hope to report on in a future article. I think the 2031 will be welcomed by many home computer enthusiasts and I congratulate Commodore for an excellent product but they must try to pay more attention to detail and get the documentation right.

Book Review

The PET Index

It must say something for the popularity of the Pet that someone could compile an index of "every" article that has ever appeared about the machine, and that a publishing company are willing to take the risk of getting the book into print and distribute it.

The PET Index, by Mick Ryan (published by Gower Press) is just such a book. Covering some 17 different publications, and extending over 290 issues of those publications, the author is to be congratulated on putting together this book: it must have been an awesome administrative task.

There are approximately 2100 references to Pet-related articles in the PET Index. These cover everything from Abacus counting to word processing, and stopping off to pay tribute to many other subjects on route. As a reference guide, it is invaluable for people such as the reviewer, who, through the nature of his job, has access to the various magazines mentioned in the index.

Just a quick skim through the book is enough to make you realise its value. If you want to find out about the 'bug' of garbage collection, turning to that reference will point you to six articles that have been printed about the subject. If you decide you're going to tackle the problem of interfacing a joystick to the Pet, you'll find that at least 3 circuit diagrams have already been published. And so it goes on: nine times out of ten you'll find that something has already been done, and this book tells you where.

Summary
It is an extremely comprehensive guide to just about everything that has been committed to print concerning the Pet. However, it's usefulness must be limited, as not everyone will have the 300 or so pounds necessary to subscribe to every publication covered. Still, as a reference source for even the two or three that you may get, it is to be recommended.

Getting Acquainted with your VIC 20
Oh dear, just about sums up my reaction to this book. Written by Tim Hartnell, and published by Interface at a cost of 5.95 pounds: what can I say?

Is there a value in publishing a damning indictement of a book? I think there is, because it at least has the virtue of making people try harder to produce worthwhile material, and not just leap into some subject for a fast buck (and I am NOT suggesting that Tim Hartnells doing this: rather, it is a general comment aimed at no-one in particular).

And so onto the book. A quick look at some of the other works that Tim Hartnell has produced gives the reason why this one doesn't quite work. Titles like Getting Acquainted with your Acorn Atom and Getting Acquainted with your ZX81 give the game away a little. For instance, the Vic DOES allow line numbers greater than 9999, it doesn't have commands like CLS and ABS (although all three of these can be applied to a certain other microcomputer). Basically this is just a re-hash of earlier works, and not a very good one at that. Still, once you've worked out where the errors lie, and if you're interested in a book that gives you over 60 progam listings, it is actually of use. Humourously written and presented, it gives you an introduction to programming in Basic generally, rather than specifically on the Vic, although the appendices go some way to amending this.

Summary
A specific book aimed at the Vic would have been much better than a re-working of earlier publications, however good that earlier work may have been. It contains numerous errors, and not enough attention has been paid to the Vic itself. You could probably count the number of times Vic is mentioned on the fingers of both hands. But if its listings that you're after, then trot along to the bookshop and have a look. Correcting the errors will probably teach you more about Basic programming than anything else in the book.

VIC 20 Programmers Reference Guide
One from Commodore themselves this time, retailing at a cost of 9.95 pounds, and should be available from any Commodore Vic dealer.

The aim of the book is to take you on further than the original manual supplied with the Vic, which shouldn't be too difficult. Being written by some respected names (Paul Higginbottom and Mike Tomczyk amongst others), you begin to feel fairly confident that it can live up to the introductory promise 'gives you more information about the Vic than any other source'.

The book is split up into four chapters, followed by a whole host of appendices, covering tables of musical notes, screen display codes and memory

maps, error messages, pinouts for input/output, peripherals and accessories, and a number of other diverse topics. They've even provided a circuit diagram of the beast itself, which is remarkable (for Commodore) in that it is very clearly drawn and laid out.

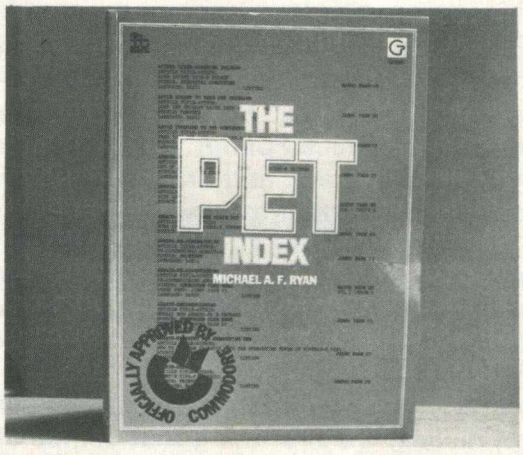
The Programmers Reference Guide starts off with a fairly gentle introduction to Basic programming on the Vic, followed by a chapter on programming tips, including quite good sections on the use of graphics, colour and sound.

The heart of the matter is reached by the time we get to chapter three, which is a guide to programming the Vic in machine language, always assuming you've got the Machine Code Monitor cartridge, or you typed in Tinyman from last month's issue. This is very nicely presented, as it starts off by assuming you know nothing, ends up talking about the use of the Kernal (a standardized jump table covering input/output and memory management routines), and covering the entire 6501-6505 microprocessor instruction set in some detail.

The final chapter devotes itself to input/output operations, including the user port, the serial bus, a little bit about the printer (why not the disk drive as well?), and some information on using joysticks, paddles and light pens.

Summary

A very good book, and well worth purchasing if you're thinking of using your Vic at all seriously. Well written, well presented, it occasionally suffers from being produced in the States (what IS a 'computerist'?!), but better grammatical errors than factual ones, which it seems remarkably free of. A good buy.



The Pet Index

Butterfield

Scanning the Stack

The stack is a group of locations from hexadecimal $0100 to $01FF that can be quickly and conveniently used by the 6502. In the PET, the stack range is limited to the area from $0140 to $01FA; but most of the time you don't need to know where the stack is working; you may just use it.

When you have something that needs keeping for a few moments, you can put it on the stack and call it back later. So long as you're neat, you don't even need to know where it will go — the processor keeps track of that with a special register called a Stack Pointer. It will put information away to the stack and bring it back without any special information from you.

But you must be neat. The slogan "Leave these premises as clean as you found them" applies critically to the way you use the stack. If you put something in there, you must be sure to call it back or you'll be in trouble.

The stack is appropriately named. It's like a stack of dishes: the first thing that comes off will be the last thing that was put on. It's called LIFO (Last-in-first-out) storage.

Standard usage

If you have a value in the A register that you want to put aside for a moment or so, you can push it to the stack using the PHA (48) instruction. Now you can use the A register for something else, and when you're finished you can call back the original value by pulling it from the stack with PLA (68).

Sometimes you might want to defer a decision. You've just done a comparison or some other activity, and the results are important — but you don't want to act on those results yet. You can push the status word — all various flags, such as Carry, Overflow, etc. — to stack with PHP (08). Now you can tidy up your registers without worrying about those losing flags. They will come back as soon as you give PLP (28) and you can then proceed with the Branch commands that will test the condition you previously set up.

When you call a subroutine with a JSR command, the stack is called into play automatically. The return address, minus one, is placed on the stack. Later, when the RTS is given, that address is called back from the stack and program execution resumes at the instruction following the JSR.

An example here might be worth while. If you are at location hex $1234, and give the instruction JSR $4455, the address $1236 will be placed on the stack. That's not your return point — you'll return to $1237 since the JSR command is 3 bytes long — but the RTS instruction will sort everything out correctly. Here's a little more detail: when the address $1236 is placed on the stack it will use two locations. The high-order part (12) goes onto the stack first, followed by the low-order portion (36).

When the 6502 receives an interrupt — and on this PET this happens 60 times a second — the current machine language instruction completes; the address of the next instruction is pushed to the stack; and finally, the processor Status Word is pushed to the stack. Then the processor starts to handle the interrupt by going to a new location and executing instructions there. When it's finished, it gives a Return from Interrupt instruction (RT) which restores the original Status Word and instruction address. The original program picks up exactly where it left off when it was interrupted.

You can see that three locations are used in the stack this time: two for the return address and one for the status word. They go onto the stack in that order: address-high, address-low, and status. It's a little like JSR followed by a PHP, since we store address and status word. Note, however, that the address is the exact return address; with a JSR the address is one less than the return address.

An example: if the processor is executing a three-byte instruction at hex $1234 and an interrupt is signalled, address $1237 is pushed to the stack, followed by the status word. Later, when RTI is executed, the status word is restored and execution resumes at address $1237.

Finally, the BRK instruction (hex $00) causes an interrupt type of action, with this difference: the address which is placed on the stack is two locations behind the Break instruction. This is odd, since the BRK command is only one byte long. In this case, if we use an RTI to continue executing the code following the BRK, we'll skip one byte.

Tabular Summary

The following table summarizes the instructions that handle the stack.

Number of bytes stored or recalled	*Store command*	*Recall command*
1	*PHA*	*PLA*
1	*PHP*	*PLP*
2	*JSR*	*RTS*
3	*Interrupt*	*RTI*
3	*BRK*	

PAPERMATE PLUS

A very powerful word processor for all PETs with 16 or 32k memory. Easy to use, but packed full of useful functions it works with tape or disk. Only £45 or £46.50 on disk.

FLEX-FILE DATABASE

Flexible, friendly - this information storage system is for CBM disk owners only. Enter, update, delete or retrieve data; print mailing labels; or produce reports showing selected records and fields (with calculated values and column totals if you wish). Why pay £250 or more - FLEX-FILE costs just £65.

CAN YOU HELP?

As the leading distributors of software and accessories for the Commodore range we're always on the look-out for top quality programs and add-ons. If you have a PET or VIC product that we would be proud to sell contact Peter Calver at once!

MX-80 MIRACLE

The latest EPSON MX-80 F/T TYPE 3 is in stock now, price £450 including an Aculab addressable interface (with integral cable). If you already have the Type 1 or Type 2 we can supply the Type 3 upgrade Roms (phone for price).

The SUPERSOFT range of HIGH RESOLUTION GRAPHICS BOARDS represents a very economical way to upgrade your PET. For £149 you get 8k of additional Ram and 320 by 200 resolution. Specify the HR-40 for small screen machines, HR-40B for the large screen 4000 and HR-80 for your 8032 or 8096.

MX-LIST (£15) prints a PET Basic program with all cursor controls, graphics, and reverse field characters

MX-DUMP (£12) copies the PET's screen dot for dot onto paper

MX-SCREENDUMP (£12) will print the contents of your HR-40/HR-80 High Resolution screen

MICROSCRIPT — AVAILABLE NOW!

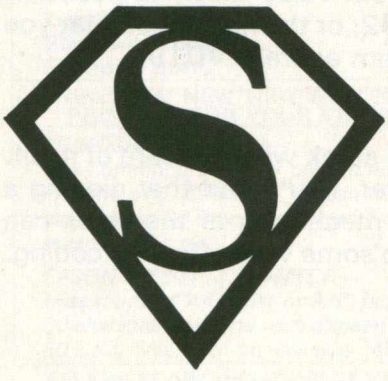
The new MicroScript word processor for 8000 series machines adds 46k of Rom and 2k of Ram to your computer. It even turns the two spare Rom sockets into four (so you can use Visicalc AND Command-O).

Powerful, but simple to learn, MicroScript has 30,000 bytes available for text - that's nearly three times the storage that other major programs offer. And the 'DataMerge' facility makes standard letters a doddle!

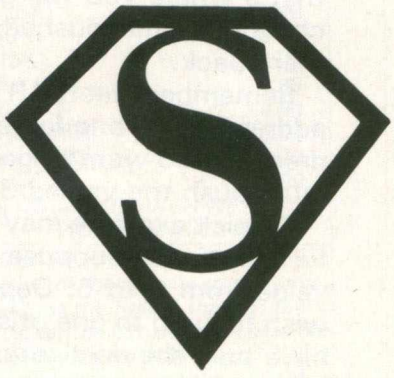
MicroScript costs £425 including manual and specially-designed ciruit board.

DEALER ENQUIRIES WELCOMED

PRE-PAID ORDERS POST FREE BUT ADD 15% VAT



SUPERSOFT



First Floor, 10—14 Canning Road, Wealdstone,
Harrow, Middlesex, HA3 7SJ, England
Telephone: 01-861 1166

ASK FOR OUR *FREE* CATALOGUE

Here's where it gets interesting. "Ordinary" programming assumes that you use the companion instruction to restore the stack. That is, if you used a PHA you should use a PLA to bring the information back. If you used a JSR, you should use an RTS. But by breaking this unwritten law, we can do some pretty fancy things. We must be careful, of course.

The Fun Begins

Suppose you are writing a subroutine. Normally, you'll want to return control to the calling point by giving the RTS command. On occasion, however, you don't want to go back; perhaps there's an error in the data so that the calling routine couldn't continue.

We can handle this. Just pop the return address from the stack with two PLA commands, and you'll never go back.

The Computed Jump

You can jump to any single location you choose by using the JMP instruction. There are times, however, when you want to jump to one of several locations depending on some value you have calculated.

For example, you might be writing a system which would jump to one routine if it detected an A (add) character, another routine for D (delete); a third for C (change), and so on.

This could be done, of course, with a series of compare, branch and jump instructions; but if the list is long, the whole thing becomes tedious and inefficient.

You can set up the equivalent of a very powerful computed jump by clever use of the stack. The principle is manufacture an address; push it to the stack with PHA... PHA; and then give RTS.

This seems puzzling at first. However can you return to a place you never came from? It works this way; by pushing the address to the stack, you stimulate a non-existent subroutine call. The stack doesn't care. If you issue an RTS instruction, the stack will deliver up that address, and that's where you will go. The stack ends up unchanged; it has pushed two values and delivered them back.

Remember that the RTS instruction expects the address to be one lower than the real return address. If you want to go address hex $3456, you must push the values 34 and 55.

A quick example may help illustrate this powerful technique. Suppose the X register contains a value from 0 to 5. Depending on this value, we wish to jump to one of six different locations. We have built the destination addresses into a set of address tables, each with six entries. The low order part of the addresses are in a table starting at hex $2320 and the high order part of the addresses are in a second table starting at hex $2326. We've carefully remembered to subtract one from each address, and the table like the following:—

```
2320 41 72 A3 C4 E5 F6
2326 24 25 27 29 2B 2C
```

If X contains zero, we want to jump to hex $2442; if one, we go to $2573; and so on. Let's do it.

```
BD 26 23    LDA $2326,X; high order first
48          PHA
BD 20 23    LDA $2320,X; low order last
48          PHA
60          RTS          ; go there
```

It's easy, it's fast, it's compact, and it's one of the most powerful tricks in the repertoire of the 6502 programmer. Microsoft Basic uses it to get to the various — PRINT, LET, FOR... etc. The Machine Language Monitor uses it to interpret it's commands — M.R., and so on.

The Stack Pointer

There are a couple of commands that tell you where the stack is working, to allow you to control where it is. You won't need to use them very often, but a little detail is worthwhile.

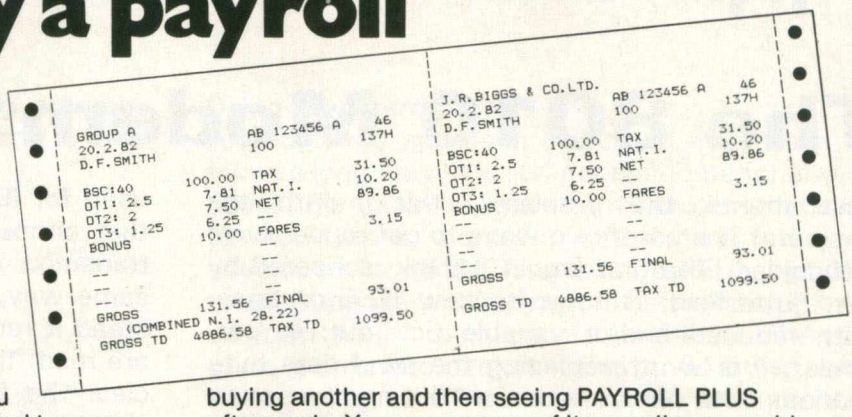
The stack works in a downwards direction. As you push things, the stack pointer gets lower. As you pull them back, the pointer goes back up. If the stack pointer gets down to its bottom value, pointing at address hex $0100, it will wrap around to $01FF if you try pushing more things in; but your program will be in serious trouble long before you reach this point.

The stack pointer indicates the next location that will be used. If the pointer has a value of 92, you know that the next value that you push will go into address $0192; or the next value that you will pull will come from address $0193.

Summary

Most of the time, the stack will take care of itself. Occasionally, however, you'll find that digging a little deeper into the mechanics of the stack can make it possible to do some very effective coding.

How to buy a payroll program...



GROUP A
20.2.82
D.F.SMITH

AB 123456 A 46
100 137H

BSC:40	100.00	TAX	31.50
OT1: 2.5	7.81	NAT.I.	10.20
OT2: 2	7.50	NET	89.86
OT3: 1.25	6.25	--	
BONUS	10.00	FARES	3.15
--		--	
--		--	
--		--	93.01
GROSS	131.56	FINAL	
(COMBINED N.I. 28.22)			
GROSS TD	4886.58	TAX TD	1099.50

J.R.BIGGS & CO.LTD.
20.2.82
D.F.SMITH

AB 123456 A 46
100 137H

BSC:40	100.00	TAX	31.50
OT1: 2.5	7.81	NAT.I.	10.20
OT2: 2	7.50	NET	89.86
OT3: 1.25	6.25	--	
BONUS	10.00	FARES	3.15
--		--	
--		--	
--		--	93.01
GROSS	131.56	FINAL	
GROSS TD	4886.58	TAX TD	1099.50

First, go to your CBM/PET dealer and ask to see at least two different payrolls. Second, make sure that one of those you see is the LANDSOFT 'PAYROLL PLUS'.

We are serious when we say you should see more than one. That way you are more likely to find the one most suited to your needs. However good PAYROLL PLUS is, there are certain things it won't do that other payrolls will — and vice versa.

Why do we say that you should insist that PAYROLL PLUS is among those you should see? Because it is so elegant in operation and so extremely easy to use. If you want a payroll that needs an expert 'computer person' to operate it, or if you enjoy spending hours trying to decipher an operating manual then PAYROLL PLUS is definitely NOT for you. You would find it too quick and easy to master.

So don't make the mistake of buying another and then seeing PAYROLL PLUS afterwards. Your awareness of its excellence could then be most frustrating!

PAYROLL PLUS is in use by a number of accountants and even bureaux.

Versions for 8032, 4032 and 3032 series CBM/PET **£150 + VAT.**

LandSoft

The Courtyard, 152-154 Ewell Road,
Surbiton, Surrey.
Telephone: 01-399 2476/7

SUPERIOR PROGRAMS FOR THE CBM/PET MICROCOMPUTER

CREAM COMPUTERS

380 STATION ROAD, HARROW, MIDDX. HA1 2DE
Telephone: 01-863 0833

HARDWARE

VIC - 20 Computer (Call now for new low low price)	
C2N Cassette Unit	39.99
1540 Floppy Unit	375.00
1515 Printer	214.95
Memory Expansion	110.75
3K Ram Expansion	27.50
8K Ram Expansion	39.98
16K Ram Expansion	68.99
Super Expander Cart	32.50
Progs Aid Cartridge	34.50
Vicmon Cartridge	30.00
Joystick	6.99
3K Store Board	51.00
8K Ram Chips	39.00
Vickit I	25.00
Vickit II	29.00
VIC - 20 Dust Cover	2.00
C2N Dust Cover	1.70
Light Pen	25.00
Rom Switch Board	42.00
Expansion Board	94.99

*NEW*NEW*NEW*NEW*NEW*NEW*NEW*
PRINTERS FOR YOUR VIC - 20

SEIKOSHA PRINTER –
The Commodore Printer plugs directly into the VIC - 20 uses 8" continuous paper up to 80 chars per line with a speed of 30 c.p.s. 214.95

EPSON MX80 F/T PRINTER –
Uses standard A4 paper or 9.5" paper. 80 characters per line with a speed of 80 c.p.s. (INC. VIC - 20 Interface) 458.99

'DISCOUNT PRICES ON QUALITY PRODUCTS'
THE VIC - 20 CENTRE

****** NEW ******

RABBIT WRITER (Wordprocessor)
Requires 8K or 16K Ram,
*includes - *mailmerge**
left and right hand justification
tape and disk format
prints up to 80 columns
uses 80 columns
uses all function keys for editing etc.
INTRO PRICE TAPE 9.99
DISK 12.50

RABBIT BASE
Requires 16K RAM Personal information master, ideal for mailing, personal records, stock control and filing.
INTRO PRICE TAPE 11.99
DISK 13.99

SPECIAL OFFERSPECIAL OFFER**
VIC - 20 PRINTER + 16K RAM PACK
+ RABBIT BASE + RABBIT WRITER
ALL THIS FOR ONLY 290.00

DON'T FORGET TO SEND OR PHONE FOR YOUR FULL FREE CATALOGUE

RABBIT SOFTWARE

Space Storm	6.99
Ski Run	4.99
Dune Buggy	4.99
Super Worm	4.99
Jungle	4.99
Cosmic Battle	4.99
Frogger	7.99
Rab Functions	4.99
Star Wars	6.99
Amok	6.99
Blitz	4.99
Invader Fall	6.99
Masterwits	6.99
Kiddie Checkers	6.99
Simple Simon	6.99
Alien Blitz	7.99
Functions	4.99
Code Breaker	4.99
Night Flight (3K)	4.99
The Alien (3K)	7.99
Amazing (3K)	6.99
Skymath (3K)	6.99
Spc Division (3K)	6.99
Charset 20 (3K)	4.99
Rabbitwriter (16K)	9.99
Rabbit Base (16K)	11.99
Rabbit Chase	4.99

ALL RUN ON BASIC MACHINE EXCEPT WHEN STATED

RABBIT CHASE
Rabbit version of the famous arcade game 'Gobbler'. Chase the Rabbit through the carrot fields, eat the lettuces and chase the farmers. Use Joystick or Keyboard.
only 4.99

ALL PRICES ARE INCLUSIVE OF VAT, POSTAGE & PACKING.
FOR FURTHER DETAILS ON ALL PRODUCTS, PLEASE CALL OR WRITE TO THE ABOVE ADDRESS.

The 8010 Modem Part Two

Last month the problems, this month the answers! There are two ways to get round these difficulties. The first (again, I think pioneered by Jim Butterfield) is to write new IEEE routines, with reduced and/or variable time-out. In particular, it is worth replacing the send data byte routines with ones which time out much sooner. This is not all that difficult, but it does involve low-level IEEE handling (sending listen attention sequence, secondary address, data byte(s), unlisten sequence) and I do not intend to discuss it further in this article.

The SRQ approach

The second technique is easier, though it only gets rid of the do-nothing delay on inputting characters from the modem. This is to make use of the SRQ (service request) line on the IEEE bus, which is implemented very nicely on the 8010 modem (though not so nicely on the PET, but you can't have everything). Whenever the modem has a character waiting to be input by the PET, it sets SRQ low to let the PET know about it. Thus there is no need for any timing out or use of ST on input — instead you need only check SRQ, and if it low handshake the waiting character straight in.

The catch, and the not-so-nice thing mentioned above, is that the designers of the PET were running a bit low on 6520 and 6522 lines when it came to fitting SRQ, in, so that it ended up somewhere a bit peculiar, though with compensating advantages. In fact it goes to the CB1 input of PIA = 2 the 6520 which lives between $E820 and $E823 in the PETs address space. An active transition on this line (high to low with PIA = 2 set up as normal in the PET) causes an interrupt flag to be set; this flag is the most significant bit of control register B of this chip, at memory location $E823. With PIA = 2 in its normal configuration, the setting of this flag does not cause a system interrupt; the flag can however easily be tested by a PEEK in BASIC, or the BIT instruction in machine code.

Readers are invited to take a stiff drink before joining me in the next paragraph.

Now, when MOS Technology designed the 6520, they certainly didn't intend CB1 for anything like SRQ. The line is designed primarily to handshake data into port B of the chip. (As used on the PET, port B is at $E822, and is actually used for IEEE output data). The result of this is that although the CB1 interrupt flag is set by a transition on the SRQ line, it is not cleared in the same way, nor can it be cleared by a POKE. Instead it remains set until the contents of port B are read; this must therefore be done in order to clear the flag, although the contents of port B should always be $FF, which is not of much interest.

The nasty thing about this (besides being jolly confusing) is that it is not standard IEEE. SRQ is supposed to be level sensitive, so that in the event of several devices requesting service (e.g. several modems) one can poll round them until all are satisfied and SRQ goes high. Instead one has an edge sensitive system, with a flag set by SRQ going low, and cleared by reading an otherwise irrelevant register. There is however a compensating advantage: by setting bit zero control register B at $E823, it can be arranged that the CB1 interrupt flag being set actually causes a CPU interrupt. The use of this is discussed later in this article.

A simple program to handle inputting from the modem using SRQ can be written in BASIC. There isn't much point to this as the speed of BASIC is such a limiting factor anyway; however the program serves as an illustration of what has to be done.

```
 10 CB=59427:PB=59426:T8=128
 20 OPEN5,5:GET#5,A$:X=PEEK(PB):PRINT"[CLR]ON LINE"
 30 PRINT"←[CL]";:GETA$:IFA$<>"" THEN PRINT#5,A$;
 40 IF (PEEK(CB)ANDT8) THEN GET#5,A$:PRINT" [CL]";A$;:X=PEEK(PB)
 50 GOTO 30
READY.
```

In line ten we set up some constants for later use, in order to speed program execution. In line 20 we open to the modem, get a character — this is necessary to get the modem set up with SRQ functioning correctly — then read port B to clear the interrupt flag. In 30 we print a cursor, then check the keyboard for user input, and if found sent to the modem. In 40 we check if the most significant bit of $E823 is set — i.e. if the SRQ interrupt flag has been set. If so there is a character waiting in the modem, so we get it, print it, then peek port B to clear the interrupt flag. Then we loop back and repeat.

This approach makes a lot more sense if implemented in machine code. Again, the following bit of code is intended only to demonstrate the

idea, and not as a complete working program. CRB and PORTB are labels corresponding to location $E822 and $E823 respectively:

```
10 CB=59427:PB=59426:T8=128
20 OPEN 5,5:GET#5,A$:X=PEEK(PB):PRINT "(CLR)ON LINE"
30 PRINT "(BACK ARROW, CL)";:GET A$:IF A$<>"" THEN PRINT#5,A$;
40 IF (PEEK(CB)ANDT8) THEN GET#5,A$:PRINT " (CL) ";A$;:X=PEEK(PB)
50 GOTO 30
```

```
2000  MAIN    JSR CGETL      ; CHARACTER FROM KEYBOARD
2010          BNE MAIN10
2020          PHA            ; IF SO SAVE CHARACTER ON STACK
2030          LDX #$05       ;       SET OUTPUT TO MODEM
2040          JSR COOUT
2050          PLA            ;       PULL CHARACTER OFF STACK
2060          JSR OUTCH      ;       OUTPUT TO MODEM
2070          JSR CLSCHN     ;       RESTORE NORMAL I/O
2080 MAIN10   BIT CRB        ; IS SRQ INTERRUPT FLAG SET
2090          BPL MAIN       ; IF NOT, NOTHING TO INPUT
2100          LDX #$05       ; ELSE SET INPUT CHANNEL TO MODEM
2110          JSR COIN
2120          JSR INCHR      ; GET WAITING CHARACTER
2130          JSR OUTCH      ; OUTPUT TO SCREEN
2140          JSR CLSCHN     ; RESTORE NORMAL I/O
2150          LDA PORTB      ; CLEAR SRQ INTERRUPT FLAG
2160          JMP MAIN       ; AND REPEAT


1000          LDA #<NEWIRQ   ; POINT AT NEW ROUTINE
1010          LDY #>NEWIRQ
1020          SEI            ; DISABLE INTERRUPTS
1030          STA IRQVEC     ; ALTER INTERRUPT VECTOR
1040          STY IRQVEC+1
1050          CLI            ; ENABLE NEW INTERRUPTS


2000          JSR CHKIRQ     ; CHECK FOR INPUT FROM MODEM
2010          JMP IRQSUB     ; THEN DO NORMAL INTERRUPT


3000 CHKIRQ   BIT CRB        ; IS SRQ INTERRUPT FLAG SET
3010          BPL CIRQ10     ; IF NOT, EXIT WITH N FLAG CLEAR
3020          LDX #$05       ; ELSE SET INPUT CHANNEL TO MODEM
3030          JSR COIN
3040          JSR INCHR      ; GET WAITING CHARACTER
3050          LDX BUFPTR     ; X=POINTER INTO BUFFER
3060          STA BUFFER,X   ; STORE CHARACTER IN BUFFER
3070          JSR CLSCHN     ; RESTORE NORMAL I/O
3080          INC BUFPTR     ; INCREMENT POINTER INTO BUFFER
3090          LDA PORTB      ; CLEAR SRQ FLAG, SET N FLAG
3100 CIRQ10   RTS            ; AND EXIT


1500          LDA #$3D
1510          STA CRB


2000 NEWIRQ   JSR CHKIRQ     ; WAS INTERRUPT CAUSED BY SRQ
2010          BMI NIRQ10     ; IF SO, N FLAG WILL BE SET
2020          JMP IRQSUB     ; IF NOT, NORMAL CLOCKED INTERRUPT
2030 NIRQ10   JMP IRQEND     ; ELSE RESTORE REGS AND RTI


6000          PHA            ; SAVE CHAR ON STACK
6010          LDX #$08
6020          SEI            ; DISABLE INTERRUPTS
6030          JSR COOUT      ; SET OUTPUT TO CHANNEL 8
6040          PLA            ; PULL CHARACTER OFF STACK
6050          JSR OUTCH      ; OUTPUT TO MODEM
6060          JSR CLSCHN     ; RESTORE NORMAL I/O
6070          JSR CHKSRQ     ; CHECK FOR INPUT FROM MODEM
6080          CLI            ; RE-ENABLE INTERRUPTS
```

This program can be expanded to allow a lot more processing to take place within the loop without any loss of characters from the modem, making it possible to add a cursor, control key, error checking, disk output, and even word-processor-style screen tricks like bidirectional scrolling. A very simple complete program of this variety is given as an appendix to this article; next month more sophisticated varieties are available from Ariadne Software Ltd. at 56 Chandos Road, London NW2, tel 01-452 4390.

Using the interrupts

Although the approach discussed above can be taken a long way, it is still not adequate for all purposes. For example, a well error protected micro-to-mainframe communication system, as developed for PETNET, uses a packet-based protocol. Suppose a long program file is to be uploaded from a PET to the host mainframe. The program is split into packets of up to 64 bytes, consisting of up to 57 data bytes and seven control bytes. The latter comprise start and end of packet bytes, a packet type identifier, the number of characters, a sequence number, and a CRC check-word. At the start of transmission, four 'slots' in the PET are filled with the first four packets in sequence, which are then transmitted to the mainframe.

When the mainframe receives a packet whose CRC checks out, it sends back to the PET a special ACK-type packet, also protected by a CRC, identifying the sequence number of the correctly-received packet.

Having sent four packets, the PET checks if the first of them has been acknowledged by the mainframe. If so, it starts sending the next packet in sequence; otherwise it starts retransmitting. This process continues until all packets have been acknowledged by the mainframe.

To implement this sort of system, the PET needs to deal with constructing output data packets with CRCs and sending them to the modem, while simultaneously watching for and performing CRC checks on ACK packets coming the other way. The exact time of arrival of these packets is unpredictable, as they are subject to a varying delay due to jobs being 'swapped' in and out of a timesharing mainframe.

The best way of doing this is to use the interrupts, so that the modem can be dealt with in the background of the main program. Very clever people (a big hello to Harry Broomhall) do both input and output to the modem as background processes using the interrupts. For our purposes however it was sufficient to deal with only input of ACK packets in the background, while the main program looks after constructing and sending the data packets.

There are two ways of doing this, which are very similar in principle. They both involve the use of a buffer, into which characters are put by the interrupt routines for later retrieval by the main code. A number of complications arise, due to the necessity of checking that buffer capacity is not exceeded, and that the interrupt-code and main program routines do not contend over the use of buffer pointers etc. These are not included in the

example code following, the object again being to demonstrate principles rather than to present complete programs.

The first method of using the interrupts for background input from the modem is to use the normal PET clocked interrupts, adding a small extra bit of code to check for input from the modem, before going on to perform the usual functions of updating the clock, scanning the keyboard etc. This can be done by altering the interrupt vector on page zero in locations $90 to $91 (IRQVEC and IRQVC+1).

The new interrupts can call a routine to check for input from the modem, then jump to the normal interrupt routine at $E455 (IRQSUB).

The routine to check for modem input can be roughly as follows:

The second method is a slight variation on this, and involves setting bit zero of control register b of PIA = 2, so that SRQ going low actually causes a processor interrupt:

The code to change the interrupt vector and the CHKIRQ subroutine are the same as before; the routine NEWIRQ subroutine are the same as before; the routine NEWIRQ however now wants to call CHKIRQ to see if the interrupt came from the modem, then either perform normal clock interrupts or restore registers and RTI (IRQWND).

With a 300-baud modem, there is nothing really to choose between the two approaches, though the first is probably simpler. (If the modem was a lot faster, characters would come in faster than the clocked interrupts (60 per second), and the second approach would have to be taken.)

By using these routines, input from the modem becomes a background operation. The only thing to be careful about is to make sure that the interrupt routines don't try and get characters from the modem while the main code is half-way through an IEEE output sequence. This can be done by disabling interrupts while outputting on the IEEE (eg to modem or disk drive), then calling CHKSRQ immediately to check if any characters came in while the output was in progress. For example, to output a character to channel 8, which could be a disk file:

These techniques are far from the last word in modem processing; one can do better by handling i/o at a lower level, and altering the normal PET IEEE routines. However, by using them some quite sophisticated communication software can be written — I wish you the very best of luck.

ACK packets

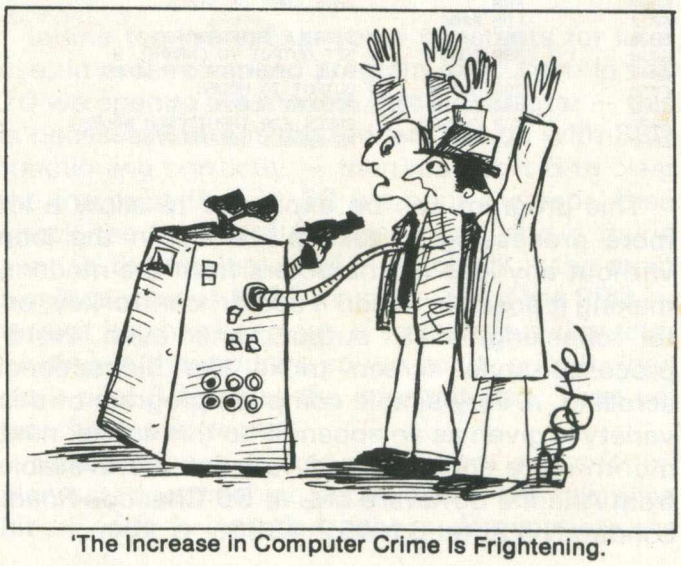
Harry Broomhall of Heronview corrected a number of misconceptions about the 8010 (those remaining are entirely our responsibility).

Nigel Bufton and Jon Rutter of ADP Network Services taught us how the big boys handle data transmission.

Nick Green of Commodore got us involved in the whole business in the first place.

Appendix

Next month we show an assembler listing and hex dump for a complete 'glass teletype' modem program, with cursor, control key, escape and delete keys. As well as some of the techniques discussed above, the program illustrates interconversion between true ASCII, the variant PETSCII, and PET screen character codes.



'The Increase in Computer Crime Is Frightening.'

HI-RESOLUTION GRAPHICS

A high resolution graphics board that gives a 64,000 dot (320×200) resolution. Versions available for any dynamic ram Pet, BASIC 2,3,4, FAT40 & 80 columns. No soldering or track cutting required, supplied complete with fast GRAPHIX software in ROM and full fitting & operating instructions. **£149.00**

SUPER ROM-SELECTOR

A high quality printed circuit board giving 64K of utility ROM space, software selectable!!! Allows 16×4K ROMs/EPROMs to reside in the expansion area of your PET. One 'POKE' enables any two ROMs at a time. Suitable for bank-switched software. **£75.00**

PET UPGRADES

WHILE-U-WAIT service! (Dynamic ram pets only).
Memory expansion:
8K-32K.....£59.90
16K-32K.....from £48.70
8K-16K.....£38.26
40 column (12in VDU only) to 80 column conversion.
40-80 column.....£89.00
40-80 column switchable (two machines in one!).....£105.00
Full keyboard functions: i.e. TAB, ESC, REPEAT, SCROLL up/down, define WINDOW, lowercase/graphics mode and DELETE from/to cursor. All available in direct or program mode.

REPAIRS AND SERVICING

Fast and efficient repairs to all Commodore computers at reasonable prices. WHILE-U-WAIT service whenever possible.

Please add VAT to the prices shown at the current rate.

Telephone Mick Bignell for more details at:

MICROSERVE
7 Clydesdale Close,
Borehamwood, Herts. WD6 2SD.
Tel: 01-953 8385

PCS

Programming Aids
UN—NEW: If you typed NEW and lost your program without saving, UN-NEW will let you recover it. Price: £10.00.
DUMP: You just touch a single key and the entire screen is copied onto the printer. This can happen even while running a program. Price: £10.00.
DISK UN-NEW 8050: Recover any scratched program on your 8050. Price: £12.00.
TINY BASIC COMPILER: A true compiler that turns your BASIC programs into fast machine code programs. For old/new/e.0 Roms. Price: £30.00.
VIGIL: Exciting, new games interactive language. Easy to learn with 60+ powerful commands. For all PETs. Also available for VIC. Price: £30.00.

Hardware
CB2-SOUNDBOX: de luxe Soundbox for Arcade games with programming manual. Price: £17.50.
TV INTERFACE: for both 40 and 80 col. PETs. Video out RF out for connection to home TV. Price: £35.00.
TELEPHONE DIALLER: use your PET to dial your telephone. Stores 40 names and 'phone numbers. More than one screen page of names and numbers possible. 40 names and numbers per page. Price: £25.00.
AD100: single board analogue to digital converter with on-board supply, 4-bit output port and drive program. Price: £55.00.
AD200: 8-channel analogue to digital converter assembled and ready for use. Comes complete with drive program and well documented handbook. Features include 8-bit output port, precision reference voltage, and many others. Price: £150.00.
PET ROM WRITER: now you can program DOS and other utilities into 2K and 4K EPROMs. Price: £55.00.

Books
THE DR. WATSON BOOK OF ASSEMBLY LANGUAGE PROGRAMMING for all PETs. Price: £10.00.
THE DR. WATSON BOOK OF ASSEMBLY LANGUAGE PROGRAMMING for VIC. Includes Assembler Cassette. Price: £17.00.
PET MACHINE LANGUAGE GUIDE. Price: £8.00.
PET REVEALED: Price: £10.00.
LIBRARY OF PET SUBROUTINES. Price: £10.00.
THE VIC REVEALED. Price: £10.00.

VIC Programs
BIG FOUR: 4 games on one cassette: Price £5.00. (3.5K).
STARTREK: fighting in the galaxy. Price: £5.00. (3K & 8K).
DATABASE: Retrieval system. Price: £7.00. (8K & above).
HI-RES CHARACTER GENERATOR: Price: £7.00. (8K & above).
THE CYBER MEN: Try to destroy them. Price: £7.00. (3.5K), (machine code).
VIC DEFENDER: Destroy fighter ships. Price £7.00. (3.5L), (machine code).

Please add 15% VAT (except books) + £0.50 p&p (£1.50 hardware only) on all orders.
Please add £1.50 if you require Programming Aids on disk. Specify 3040/4040 or 80 Format.
We also stock floppy disks, printer ribbons & cassettes. Send s.a.e. for catalogue.

PEDRO COMPUTER SERVICES

4 Cowcross Street, London EC1. Telephone: 01-250 1481.

Wego Computers Ltd

BackPack – A standby power supply that fits internally within the PET and Disk Drive. It gives at least 15 minutes to close the system down in an orderly fashion.
£140 + VAT

The Card Reader is now available with an optical read head, which can also read punch cards.
from £650 + VAT

The Sequential Switching Unit for powering up and down various devices in the correct sequence.
£75 + VAT

The TNW 1000 – an output only addressable serial interface IEEE to RS232C.
£110 + VAT

Wego Computers Ltd
22a. High Street, Caterham, Surrey CR3 5UA
Tel: Caterham 49235 Telex 933660 WEGO-G

Pet Talker

The project was originated from an idea by the 'A' level group in the school's 6th Form.

We have so far made it possible for two CBM 4032's to talk to each other using a control key. (The OFF/RVS key).

Within this documentation, we have included a hardcopy of the program (which must be in both machines), to enable you to set up your own 'mini-network'.

To implement the software however you will need to build your own USER-PORT connector. We constructed our own as we do not know of any such connector available on the market. We used standard Commodore supplied plug-in pins and soldered the wires to these pins. Enclosed is a diagram which shows how to connect the wire.

Hardware requirements
2 CBM 4032 MICROS

1 USER PORT CONNECTOR

Software requirements
Each CBM must have the program in memory.

Acknowledgements
The software and hardware were designed in December 1981 at the Pingle School, Swadlincote by the following:—
Neil Hudson
Neil Dutton
Nigel West
John Cantrill
Sean Hancock
Adrian Lakin
Computer Studies Dept. the Pringle School, Swadlincote, Burton-on-Trent, Staffs DE11 0QA.
WITH VERY SPECIAL THANKS TO MR. BARRY CLAYDON FOR HIS WORK DURING THE DEVELOPMENT OF THE PROJECT.

User's Guide
1. Load the program in the normal way (tape or disk) into both machines.
2. Connect the connector to the User-Port as described in that section of the documentation.
3. RUN the program on both machines.
4. There will be no response until one of the users presses the control key i.e. the OFF/RVS key.
5. Pressing this key will set that machine into 'TALK-MODE'. The other machine automatically becomes a 'LISTENER'.
6. The 'TALKER' may now send his message down the 8-bit bus to the 'LISTENER'.
7. During this time the 'LISTENER' must not press the control key otherwise a collision will occur on the bus!!
8. To get out of 'TALK-MODE', press the OFF/RVS key again. This resets both machines.
9. If the 'LISTENER' now wishes to talk he simply presses the OFF/RVS key and he will become the 'TALKER'.
10. Simple really, isn't it?

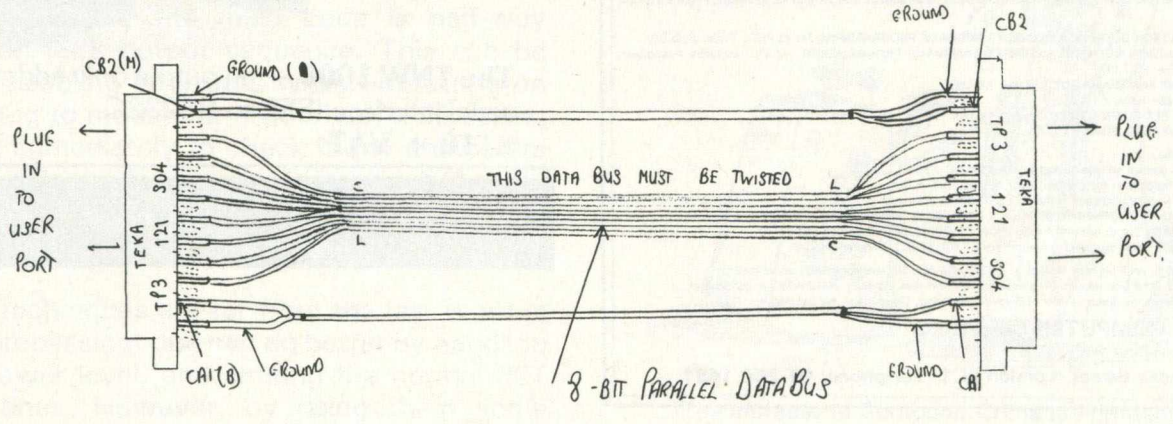
The Connector
Requirements (not very expensive)
1 plug-pins (TEKA TP3 121 S04)
1 8-bit parallel data bus (a sample is included)
2 Handshake lines which consist of two wires which connect to the CA1/CB2/GROUND pins (depending on their position).

How to construct it
1. Connect the 8-bit data bus to the middle eight pins on the lower pins of the plug-pins on each plug-pin by soldering preferably.

Note: The bus must be twisted so as the wires connect as shown (e.g. A to A,B to B etc.)

2. Connect the 'Handshake lines' which consist of two wires each to the outer pins of the plug-pins.

Note: Ground wire connects to end pin in both cases, and the CA1 line must connect to the CB2 line of the other machine. (See diagram).



Pet Talker Listing

```
100 DA=59459 : REM DATA DIRECTION REG.
110 :
120 PA=59457 : REM PORT A WITH HANDSHAKING
130 :
140 AX=59467 : REM AUXILLARY CONTROL REG.
150 :
160 PR=59468 : REM PERIPHERAL CONTROL REG.
170 :
180 IA=59469 : REM INTERRUPT REG.
190 :
200 IE=59470 : REM INTERRUPT ENABLE REG.
210 :
220 POKE PR,(PEEK(PR)AND254)
230 POKE AX,(PEEK(AX)AND227)
240 POKE PR,(PEEK(PR)OR224)
250 POKE IE,2
260 POKE AX,(PEEK(AX)OR1)
270 POKE DA,0
280 POKE IA,2
290 GET A$
300 :
310 IF A$="" THEN 370
320 :
330 IF ASC(A$)=18 THEN GOSUB 440
340 :
350 :
360 A=PEEK(PA)
370 IF PEEK(IA)AND2 = 0 THEN 400
380 IF PEEK(PA) = 0 THEN GOSUB 1020
390 :
400 GOTO 290
410 :
420 :
430 :
440 REM DEDICATED TALKER
450 :
460 REM GRAB BUS
470 :
480 POKE DA,255
490 :
500 POKE PA,0
510 :
520 POKE PR,(PEEK(PR)OR224)
530 POKE PR,(PEEK(PR)AND223)
540 :
550 T=TI
560 :
570 C=PEEK(IA)AND2
580 :
590 IF (C<>2) AND ((TI-T)<120) THEN 570
600 :
610 IF (TI-T)>=120 THEN GOTO 670
620 :
630 REM PROCESS TALK
640 :
650 GOSUB 700
660 RETURN
670 PRINT "COLLISION"
680 POKE DA,0
690 RETURN
700 REM ******************************
710 REM ******************************
720 REM ****************************
730 REM
740 REM WAIT FOR KEY TO BE PRESSED.
750 REM
760 REM ****************************
770 :
780 A$="":GETA$:IFA$=""THEN 780
790 IF ASC(A$)=18 THEN A$=CHR$(1):GOTO 870
800 PRINT A$;
810 REM ******************************
820 REM
830 REM PLACE DATA ONTO DATA BUS
840 REM
850 REM ******************************
860 :
870 POKE 59457,(ASC(A$))
880 POKE 59468,(PEEK(59468)OR224)
890 POKE 59468,(PEEK(59468)AND223)
900 :
910 :
920 :
930 :
940 :
950 :
960 LET C=PEEK(59469)
970 LET C=C AND 2
980 IF C=0 THEN 960
990 IF ASC(A$)<>1 THEN 780
1000 POKE DA,0
1010 RETURN
1020 REM      LISTENER
1030 :
1040 POKE 59459,0
1050 LET C=PEEK(59469)
1060 LET C=C AND 2
1070 :
1080 IF C=0 THEN 1050
1090 :
1100 LET D=PEEK(59457)
1110 IF D=1 THEN 1160
1120 :
1130 PRINT CHR$(0);
1140 :
1150 :
1160 POKE 59468,(PEEK(59468)OR224)
1170 :
1180 POKE 59468,(PEEK(59468)AND223)
1190 IF D<>1 THEN 1050
1200 RETURN
```

```
0010: 033A           SGL*II ORG   $033A
0020: 033A           INDEXL *     $0062
0030: 033A           INDEXH *     INDEXL -01
0040: 033A           ACCELL *     INDEXL -02
0050: 033A           ACCELH *     INDEXL -03
0060: 033A           SLOW   *     INDEXL +01
0070: 033A           PIA    *     $E840
0080: 033A           PORTA  *     PIA    +0F
0090: 033A           IER    *     PIA    +0E
0100: 033A           ACR    *     PIA    +0B
0110: 033A           PCR    *     PIA    +0C
0120: 033A           TCL    *     PIA    +06
0130: 033A           TCH    *     PIA    +05
0140: 033A           IFR    *     PIA    +0D
0150: 033A           DDRA   *     PIA    +03
0160: 033A           FLPINT *     $D09A
0170: 033A           TIME   *     $7FFF
0180:
0190: 033A 20 9A D0          JSR   FLPINT ;CONV.TO INTERGER
0200: 033D 78                SEI
0210: 033E A0 FF             LDYIM $FF
0220: 0340 8C 43 E8          STY   DDRA
0230: 0343 A9 00             LDAIM $00
0240: 0345 8D 4F E8          STA   PORTA  ;PTA O/P'S LOW
0250: 0348 A9 C0             LDAIM $C0
0260: 034A 85 63             STAZ  SLOW
0270: 034C A5 61             LDAZ  INDEXH
0280: 034E 10 16             BPL   POS    ;+VE INDEX
0290: 0350 A9 08             LDAIM $08
0300: 0352 8D 4F E8          STA   PORTA  ;SET -VE
0310: 0355 98                TYA
0320: 0356 45 62             EORZ  INDEXL
0330: 0358 18                CLC
0340: 0359 69 01             ADCIM $01
0350: 035B 85 62             STAZ  INDEXL ;COMP.& +1
0360: 035D 98                TYA
0370: 035E 45 61             EORZ  INDEXH
0380: 0360 90 02             BCC   SKIP
0390: 0362 69 00             ADCIM $00    ;ADD CARRY
0400: 0364 85 61      SKIP   STAZ  INDEXH
0410: 0366 A5 62      POS    LDAZ  INDEXL
0420: 0368 85 60             STAZ  ACCELL
0430: 036A A5 61             LDAZ  INDEXH
0440: 036C 85 5F             STAZ  ACCELH
0450: 036E D0 0F             BNE   LLP    ;LARGE INDEX
0460: 0370 A5 62             LDAZ  INDEXL
0470: 0372 D0 03             BNE   CONT   ;NOT ZERO
0480: 0374 4C DE 03          JMP   FINISH
0490: 0377 C9 50      CONT   CMPIM $50
0500: 0379 B0 04             BCS   LLP    ;)80 STEP INDEX
0510: 037B A9 40             LDAIM $40
0520: 037D 85 63             STAZ  SLOW   ;'S' ONLY
0530: 037F AD 4C E8   LLP    LDA   PCR
0540: 0382 09 01             ORAIM $01
0550: 0384 8D 4C E8          STA   PCR    ;CA1 +VE EDGE
0560: 0387 AD 4B E8          LDA   ACR
0570: 038A 29 3F             ANDIM $3F
0580: 038C 8D 4B E8          STA   ACR    ;MONO TIMER
0590: 038F AD F5 03          LDA   TL
0600: 0392 8D 46 E8          STA   TCL
0610: 0395 AD F6 03          LDA   TH
0620: 0398 8D 45 E8          STA   TCH
0630: 039B AD 4F E8          LDA   PORTA
0640: 039E 05 63             ORAZ  SLOW
0650: 03A0 8D 4F E8          STA   PORTA  ;START INDEX
0651:
0652:                  ;INDEXING
0653:
0660: 03A3 A9 02      START  LDAIM $02
0670: 03A5 8D 4D E8          STA   IFR
0680: 03A8 2C 4D E8   LP     BIT   IFR
0690: 03AB F0 FB             BEQ   LP     ;TEST PULSE
0700: 03AD A6 62             LDXZ  INDEXL
0710: 03AF D0 02             BNE   NOCRY  ;'XL' NOT ZERO
0720: 03B1 C6 61             DECZ  INDEXH ;DEC.'XH'
0730: 03B3 C6 62      NOCRY  DECZ  INDEXL ;DEC. 'XL'
0740: 03B5 D0 04             BNE   NOTZ
0750: 03B7 A5 61             LDAZ  INDEXH
0760: 03B9 F0 23             BEQ   FINISH ;ALL OVER
0770: 03BB AD 4F E8   NOTZ   LDA   PORTA
0780: 03BE 29 80             ANDIM $80
0790: 03C0 10 E1             BPL   START  ;'F' OFF
0800: 03C2 A9 40             LDAIM $40
```

```
0810: 03C4 2C 4D E8         BIT   IFR
0820: 03C7 70 05            BVS   STONE  ;ACC.TIME OVER
0830: 03C9 20 E8 03         JSR   DECACC ;DEC.ACCEL
0840: 03CC F0 05            BEQ   NFAST
0850: 03CE 20 E8 03  STONE  JSR   DECACC
0860: 03D1 D0 D0            BNE   START
0870: 03D3 AD 4F E8  NFAST  LDA   PORTA
0880: 03D6 29 7F            ANDIM $7F
0890: 03D8 8D 4F E8         STA   PORTA  ;'F' OFF
0900: 03DB 4C A3 03         JMP   START
0901:
0910: 03DE AD 4F E8  FINISH LDA   PORTA
0920: 03E1 29 3F            ANDIM $3F
0930: 03E3 8D 4F E8         STA   PORTA  ;'F&S' OFF
0940: 03E6 58               CLI
0950: 03E7 60               RTS
0951:
0952:                    ;DEC. 2 BYTES ACC.
0953:
0960: 03E8 A6 60     DECACC LDXZ  ACCELL
0970: 03EA D0 02            BNE   NC
0980: 03EC C6 5F            DECZ  ACCELH
0990: 03EE C6 60     NC     DECZ  ACCELL
1000: 03F0 D0 02            BNE   NZ
1010: 03F2 A5 5F            LDAZ  ACCELH
1041: 03F4 60        NZ     RTS
1042:
1050: 03F5 FF        TL     =     TIME
1060: 03F6 7F        TH     =     TIME   /256
0010: 1800           DBL*II ORG   $1800
0020: 1800           SLOW   *     $0054
0030: 1800           ACCXH  *     SLOW   +01
0040: 1800           ACCXL  *     SLOW   +02
0050: 1800           INDXH  *     SLOW   +03
0060: 1800           INDXL  *     SLOW   +04
0070: 1800           ACCYH  *     SLOW   +05
0080: 1800           ACCYL  *     SLOW   +06
0090: 1800           INDYH  *     SLOW   +07
0100: 1800           INDYL  *     SLOW   +08
0110: 1800           PIA    *     $E840
0120: 1800           PORTA  *     PIA    +0F
0130: 1800           IER    *     PIA    +0E
0140: 1800           ACR    *     PIA    +0B
0150: 1800           PCR    *     PIA    +0C
0160: 1800           TCL    *     PIA    +06
0170: 1800           TCH    *     PIA    +05
0180: 1800           IFR    *     PIA    +0D
0190: 1800           DDRA   *     PIA    +03
0200: 1800           TIME   *     $7FFF
0210:
0220: 1800 78                SEI
0230: 1801 A0 FF             LDYIM $FF
0240: 1803 8C 43 E8          STY   DDRA
0250: 1806 A9 00             LDAIM $00
0260: 1808 8D 4F E8          STA   PORTA  ;PTA O/PS LOW
0270: 180B A5 54             LDAZ  SLOW
0280: 180D 09 F0             ORAIM $F0
0290: 180F 85 54             STAZ  SLOW   ;X&Y F&S ON
0291: 1811 AD 26 19          LDA   XL
0300: 1814 85 58             STAZ  INDXL
0310: 1816 85 56             STAZ  ACCXL
0311: 1818 AD 27 19          LDA   XH
0320: 181B 85 57             STAZ  INDXH
0330: 181D 85 55             STAZ  ACCXH
0340: 181F D0 16             BNE   YIND   ;LARGE INDEX
0350: 1821 A5 58             LDAZ  INDXL
0360: 1823 D0 08             BNE   CONT   ;NOT 0 INDEX
0370: 1825 A5 54             LDAZ  SLOW
0380: 1827 29 3F             ANDIM $3F
0390: 1829 85 54             STAZ  SLOW   ;0 INDEX
0400: 182B D0 0A             BNE   YIND
0410: 182D C9 50     CONT    CMPIM $50
0420: 182F B0 06             BCS   YIND   ;)80 INDEX
0430: 1831 A5 54             LDAZ  SLOW
0440: 1833 29 7F             ANDIM $7F    ;NO FAST
0450: 1835 85 54             STAZ  SLOW
0451: 1837 AD 28 19  YIND    LDA   YL
0460: 183A 85 5C             STAZ  INDYL
0470: 183C 85 5A             STAZ  ACCYL
0471: 183E AD 29 19          LDA   YH
0480: 1841 85 5B             STAZ  INDYH
0490: 1843 85 59             STAZ  ACCYH
0500: 1845 D0 1D             BNE   LLP
0510: 1847 A5 5C             LDAZ  INDYL
0520: 1849 D0 0F             BNE   PLS    ;NOT 0 INDEX
0530: 184B A5 54             LDAZ  SLOW
0540: 184D 29 CF             ANDIM $CF
0550: 184F 85 54             STAZ  SLOW
0560: 1851 29 C0             ANDIM $C0
0570: 1853 D0 03             BNE   PL     ;X NOT 0
0580: 1855 4C 15 19          JMP   FINISH ;BOTH ZERO
0590: 1858 A5 5C     PL      LDAZ  INDYL
0600: 185A C9 50     PLS     CMPIM $50
0610: 185C B0 06             BCS   LLP    ;)80 INDEX
0620: 185E A5 54             LDAZ  SLOW
0630: 1860 29 DF             ANDIM $DF
0640: 1862 85 54             STAZ  SLOW   ;NO FAST
0650: 1864 AD 4C E8  LLP     LDA   PCR
0660: 1867 09 11             ORAIM $11    ;CA1,CB1 + EDGE
0670: 1869 8D 4C E8          STA   PCR
0680: 186C AD 4B E8          LDA   ACR
0690: 186F 29 3F             ANDIM $3F
0700: 1871 8D 4B E8          STA   ACR
0710: 1874 AD 24 19          LDA   TL
0720: 1877 8D 46 E8          STA   TCL
0730: 187A AD 25 19          LDA   TH
0740: 187D 8D 45 E8          STA   TCH
0750: 1880 AD 4F E8          LDA   PORTA
0760: 1883 05 54             ORAZ  SLOW
0770: 1885 8D 4F E8          STA   PORTA  ;START INDEXING
0780:
0790:                    ;INDEXING
0800:
0810: 1888 A9 12     START   LDAIM $12
0820: 188A 8D 4D E8          STA   IFR
0821: 188D A9 12     LP      LDAIM $12
0830: 188F 2C 4D E8  L       BIT   IFR    ;PULSE ?
0840: 1892 F0 FB             BEQ   L
0850: 1894 AD 4D E8          LDA   IFR
0860: 1897 29 02             ANDIM $02
0870: 1899 F0 2F             BEQ   Y      ;'Y'PULSE
0880: 189B 8D 4D E8          STA   IFR
0890: 189E A2 04             LDXIM $04
0900: 18A0 20 17 19          JSR   DECR   ;DEC. 'X'
0910: 18A3 F0 56             BEQ   FINX   ;'X'OVER
0911: 18A5 A2 02             LDXIM $02
0920: 18A7 AD 4F E8          LDA   PORTA
0930: 18AA 29 80             ANDIM $80
0940: 18AC 10 DF             BPL   LP     ;NO FAST
0950: 18AE A9 40             LDAIM $40
0960: 18B0 2C 4D E8          BIT   IFR
0970: 18B3 70 05             BVS   STONE  ;ACC.OVER
0990: 18B5 20 17 19          JSR   DECR   ;DEC. ACC.
1000: 18B8 F0 05             BEQ   NFAST  ;HALF WAY
1020: 18BA 20 17 19  STONE   JSR   DECR
1030: 18BD D0 CE             BNE   LP     ;NEXT
1040: 18BF AD 4F E8  NFAST   LDA   PORTA
1050: 18C2 29 7F             ANDIM $7F
1060: 18C4 8D 4F E8          STA   PORTA  ;FAST OFF
1070: 18C7 4C 8D 18          JMP   LP
1080:
1090:                    ;'Y' INDEX
1100:
1110: 18CA A9 10     Y       LDAIM $10
1120: 18CC 8D 4D E8          STA   IFR
1130: 18CF A2 08             LDXIM $08
1140: 18D1 20 17 19          JSR   DECR   ;DEC. 'Y'
1150: 18D4 F0 34             BEQ   FINY   ;'Y'OVER
1151: 18D6 A2 06             LDXIM $06
1160: 18D8 AD 4F E8          LDA   PORTA
1170: 18DB 29 20             ANDIM $20
1180: 18DD F0 AE             BEQ   LP     ;NO FAST
1190: 18DF A9 40             LDAIM $40
1200: 18E1 2C 4D E8          BIT   IFR
1210: 18E4 70 05             BVS   SONE   ;ACC.OVER
1230: 18E6 20 17 19          JSR   DECR   ;DEC. ACC.
1240: 18E9 F0 05             BEQ   NF     ;HALF WAY
1260: 18EB 20 17 19  SONE    JSR   DECR
1270: 18EE D0 9D             BNE   LP     ;NEXT
1280: 18F0 AD 4F E8  NF      LDA   PORTA
1290: 18F3 29 DF             ANDIM $DF
1300: 18F5 8D 4F E8          STA   PORTA  ;FAST OFF
1310: 18F8 4C 8D 18          JMP   LP     ;NEXT PULSE
1320:
1330:                    ;END OF 'X'
1340:
1350: 18FB AD 4F E8  FINX    LDA   PORTA
1360: 18FE 29 3F             ANDIM $3F    ;'F'&'S' OFF
1370: 1900 8D 4F E8          STA   PORTA
1380: 1903 29 F0     TEST    ANDIM $F0
1390: 1905 F0 0E             BEQ   FINISH ;BOTH OVER
1400: 1907 4C 8D 18          JMP   LP
1410:
1420:                    ;END OF 'Y'
1430:
1440: 190A AD 4F E8  FINY    LDA   PORTA
1450: 190D 29 CF             ANDIM $CF
1460: 190F 8D 4F E8          STA   PORTA  ;'F'&'S' OFF
1470: 1912 4C 03 19          JMP   TEST   ;ALL OVER?
1480: 1915 58        FINISH  CLI
1490: 1916 60                RTS
1500:
1510:                    ;DEC. 2 BYTES
1520:
1530: 1917 B5 54     DECR    LDAZX SLOW
1540: 1919 D0 02             BNE   NC
1550: 191B D6 53             DECZX SLOW   -01
1560: 191D D6 54     NC      DECZX SLOW
1570: 191F D0 02             BNE   NZ
1580: 1921 B5 53             LDAZX SLOW   -01
1590: 1923 60        NZ      RTS
1600:
1610: 1924 FF        TL      =     TIME
1620: 1925 7F        TH      =     TIME   /256
1630: 1926 30        XL      =     '0
1640: 1927 30        XH      =     '0
1650: 1928 30        YL      =     '0
```

Sound and Vision

Another Voice for the VIC-20

Normally, your VIC has 4 musical voices . . . three music registers and a white noise register. But by connecting a small amplifier to the User Port, and doing a little programming, you can get another musical voice.

The User Port on the VIC is very similar to the PET User Port. This makes it easy to adapt some of the PETs music methods to the VIC-20.

Background — Adding Sound to Older PET/CBMs

Before Commodore introduced the CBM 8032 with a built-in speaker, most PET/CBM users had to develop their own means of getting their computers to squeak, hum, whistle and sing. They came up with the idea of using the User Port to send square waves through an external amplifier/speaker combination. The shift register could be programmed through BASIC, giving a wide variety of squeals, pops, sirens, etc.

Theory

Most music is made up of square waves of different amplitudes and frequencies. One of the functions of the 6522 chip is to generate square waves through the CB2 line. If we connect the CB2 line to a speaker, we will be able to hear the square waves generated by the VIC.

NOTE: Connecting a speaker directly to CB2 may damage your VIC and void your warranty. You must connect the speaker through an amplifier to protect the VIC.

BASIC Program Steps

1. Set the 6522 shift register to free running mode by typing:

POKE 37147, 16

2. Set the shift rate by typing:

POKE 37144, C

. . . where C is an integer from 0 to 255. C is the note to be played.

3. Load the shift register by typing:

POKE 37146, D

. . . where D = 15, 51 or 85 for a true square wave. This step also sets the octave for the note.

This step must be done last, since as soon as it is set, the VIC starts generating the square waves.

The frequency of the square wave can be found by the following formula:

$$FREQ = \frac{50000\ Hz}{(C+2)*(D1)}$$

where D1=8 when D=15
D1=4 when D=51
D1=2 when D=85

When you're in this mode, the VIC will not read or write to cassette. To restore normal operation, just type:

POKE 37147, 0

Here are some pre-calculated values for C:

```
Bb = 251        B   = 124
C  = 237        C1  = 117
C# = 224        C1# = 111
D  = 211        D1  = 104
D# = 199        D1# =  99
E  = 188        E1  =  93
F  = 177        F1  =  88
F# = 167        F1# =  83
G  = 157        G1  =  78
G# = 149        G1# =  73
A  = 140        A1  =  69
A# = 132
```

The following short program demonstrates music using this method. By hitting a letter, a note will be played.

```
10 PRINT "MUSIC USING CB2"
20 REM A TO G IS ONE OCTAVE, SHIFT A TO G IS ANOTHER
30 PRINT "HIT + TO GO UP AN OCTAVE, - TO GO DOWN"
40 PRINT "[DN] USE ! TO EXIT."
50 POKE 37147, 16
60 DIM A(14)
70 FOR I=1 TO 14
80 READ A(I)
90 NEXT
100 DATA 124, 117, 104,  93,  88,  78,  69
110 DATA 251, 237, 211, 188, 177, 157, 140
200 GET A$ : IF A$="" THEN 200
210 IF A$="!" THEN POKE 37147, 0 : END
220 IF A$="+" THEN SF=SF-(SF<2) : GOTO 200
230 IF A$="-" THEN SF=SF+(SF>0) : GOTO 200
240 A=ASC(A$)-64+(ASC(A$)>192)*121
250 IF A>14 OR A<1 THEN 200
260 POKE 37144, A(A)
270 POKE 37146, -(SF=0)*15-(SF=1)*51-(SF=2)*85
280 GOTO 200
```

BASIC Plotter

This program will plot random lines using the "quarter-square" graphics characters. Although it's a program in itself, it could easily be made into a subroutine.

The program has been set up for 80 column screens (line 9040). Notice "LL" (Line Length) is multiplied by 2 in lines 2020 & 2030? Since the quarter squares use up half a character space in the "x" direction, an 80 column screen can have up to 160 "half-characters" horizontally. Similarly, on 25 lines there can be up to 50 half characters vertically ("y" direction). For 40 column screens you'll need to change LL to 40; the

second parameter remains the same since both have 25 lines.

Line 2000 clears the window (if one set), the screen, and sets graphics mode (no gap between lines). If you like, substitute CHR$(142) with 'esc-rvs-N' and stick it inside the quotes.

```
2000 PRINT"[HM HM CLR]"CHR$(142)
2010 GOSUB 9000
2020 X1=INT(RND(TI)*LL*2) : Y1=INT(RND(TI)*50)
2030 X2=INT(RND(TI)*LL*2) : Y2=INT(RND(TI)*50)
2040 GOSUB 3000 : Y1=Y2 : X1=X2 : GOTO 2030
3000 REM **********  PLOT A LINE  **********
3010 DX=X2-X1 : DY=Y2-Y1 : X=X1 : Y=Y1
3020 L=SQR(DX*DX+DY*DY) : IF L=0 THEN 3040
3030 XI=DX/L : YI=DY/L
3040 GOSUB 8000 : IF (ABS(X2-X)<=ABS(XI)) AND
     (ABS(Y2-Y)<=ABS(YI)) THEN RETURN
3050 X=X+XI : Y=Y+YI : GOTO 3040
8000 REM **********   PLOT X, Y   **********
8010 TX=INT(X+IR):TY=INT(Y+IR) :SQ=AM(TX AND AM, TY AND AM)
8020 P=BS+TX/DV-INT(TY/DV)*LL : POKE P, C(I(PEEK(P))OR SQ)
     : RETURN
9000 REM **********      SETUP      **********
9010 DIM C(15), I(255), AM(1,1)
9020 FOR I=0 TO 15 : READ C(I) : I(C(I))=I : NEXT
9030 FOR I=0TO1 : FOR J=0TO1 : AM(J,I)=(J+1)*4↑I : NEXTJ,I
9040 LL=80 : BS=32768+24*LL : DV=2 : AM=1 : IR=.5
9050 DATA  32, 123, 108,  98, 126,  97, 127, 252,
         124, 255, 225, 254, 226, 236, 251, 160
9060 RETURN
```

The subroutine at 9000 sets up an array with the 16 possible combinations of the quarter squares. BS is the base address of the POKE address of the bottom left corner of the screen.

All plotting efforts are performed by the two subroutines at 3000 & 8000. Subroutine 3000 plots a line from x1,y1 to x2,y2 by plotting several points (sub 8000). At the same time, subroutine 8000 must determine if there is already a point in a character space. If there is, the POKE information must not interfere with existing points.

Lines 200X are used for plot criteria generation. The above merely plots random lines. For something more meaningful, try substituting with these:

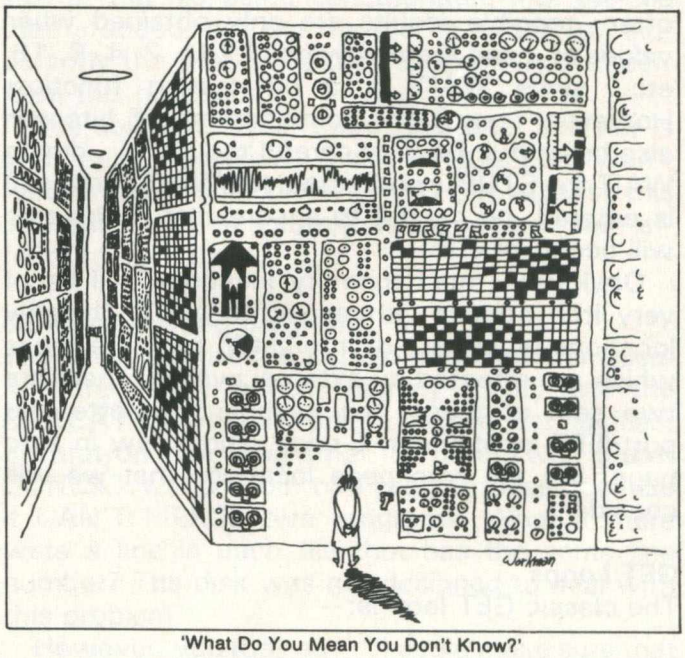
```
2020 X1=0 : Y1=1
2025 FOR X2=0 TO 159
2030 Y2=EXP (X2/31.4)
2040 GOSUB 3000 : Y1=Y2 : X1=X2 : NEXT : END

2020 N=6 : C=3.1415926/160 : X1=0 : Y1=25
2025 FOR X2=0 TO 159
2030 Y2=25 + 24 * SIN(X2 * N * C)
2040 GOSUB 3000 : Y1=Y2 : X1=X2 : NEXT : END

2020 N=8 : C=3.1415926/160 : X1=0 : Y1=50 : DC=100
2025 FOR X2=0 TO 159
2030 Y2=25 + 24 * COS(X2 * N * C) * EXP(-X/DC)
2040 GOSUB 3000 : Y1=Y2 : X1=X2 : NEXT : END
```

The first plots an exponential curve. Notice the Y origin is set to 1 rather than 0. This accounts for a slight inaccuracy as the plotter draws horizontal lines using the top ''half-character'' rather than the bottom half-character. This could be changed by modifying the character table at 9050.

The second draws a SINE curve starting half way up the screen (Y1 = 25). The variable N represents the number of half cycles displayed (N = 6 will draw 3 complete cycles).

The last one is a decaying COSINE wave, origin at top-left (Y1 = 50). For higher decay rates, use lower values in DC.

Finally, with little effort you could use the plotter routine to draw axes for your functions.



'What Do You Mean You Don't Know?'

Programming Techniques and Tips

This month we feature not one long article, but rather a collection of facts and figures that people have discovered along the way. We've tried, as far as possible, to cover the whole range of Commodore machines, but in some cases we have to admit to failure. If anyone wishes to write in with the equivalent version for another machine, who are we to argue? (And we pay, as well . . .!).

Fun with WAIT Statements — Henry Troup
Most of us find that the WAIT statement is of limited use. Until recently, the only use I ever had found was:—

```
WAIT 59411, 8, 8
```

to wait for the cassette recorder play switch to be pressed. But I did later find some amusing and useful applications for WAIT.

But first, a quick review.

The statement WAIT I, J, K causes the value of location I to be exclusively OR'ed with K, and AND'ed with J. If the result is 0, the process repeats until a non-zero result is obtained. Most often, tangible results are only obtained when values of J and K are powers of 2 (1, 2, 4, 8, 16, etc.) since WAIT is a bit testing function. However, testing for combinations of bits can also be useful. Be very careful though . . . during WAIT the STOP is not tested. If a WAIT command is entered, be careful to ensure that a non-zero will occur, or else!

Obviously, most memory locations will be of very little interest with respect to WAIT. The only locations which are of interest in fact, are those which are affected by external events. There are two sets of these: the keyboard/cassette/user port/IEEE locations in E-page, and a few in zero page. It's the zero page locations that we will consider.

GET Loops
The classic GET loop is:—

```
10 GET A$ : IF A$ = "" THEN 10
```

which loops until a non-null input is received. The same affect can be obtained by WAITing for the keyboard buffer pointer:—

```
10 WAIT 157, 127 : GET A$
```

This waits until the keyboard buffer count (decimal 158 for BASIC 2 and 4, 525 for BASIC 1, 198 on the VIC) is non-zero. It's a little harder to understand, but shorter and probably slightly faster. For experimentation, try replacing the GET command with INPUT, and the 127 with 2, 4 and 8.

WAITing for a key
Very often a GET loop is used on a 'Push any key to continue' basis. One interesting alternative is to use:—

```
WAIT 152,1
```

This waits for the shift key to be pressed (516 on BASIC 1, 657 on the VIC). The advantage of this is that nothing is put into the keyboard buffer, so you don't have to clear it first.

Or, if you want to have fun, try experimenting with WAITing for location 151 — key held down (515 on BASIC 1, 197 on the VIC). WAIT 151, 127, 255 will wait for any key. Specific keys are harder to WAIT for, since WAIT will only wait on one bit at a time. Remember that we're talking about undecoded keyboard values here.

WAITing for the Clock
The real time clock occupies locations 141-143 (512-514 on BASIC 1, 160-162 on the VIC) in zero page. WAITing for one particular bit in the clock to change state gives an interesting delay effect. For example, WAIT 142, 1, 1 will wait for the rightmost bit of the second byte. This bit chages state every 256th jiffy, or slightly over 4 seconds. WAIT 143, 1, 1 will wait till the start of the next jiffy.

While some of these are not particularly useful, playing with the WAIT statement is quite a bit of fun. If anyone finds any more useful or interesting locations, we'll be WAITing to hear from you.

Screen Loading
All you need is a "screen-set-up" routine to "draw" your screen out, and this program will store it on disk:—

```
100 REM SCREEN SAVER
110 OPEN 8, 8, 8, "0:SCREEN NAME, P, W"
120 PRINT#8, CHR$(0)CHR$(128);
125 SS = 32767
130 EN = 33767 : IF PEEK (50003) = 160 THEN EN = 34767 : GOTO140
135 IF PEEK (50003) = 194 THEN SS = 7680 : EN = 8186
140 FOR J = SS TO EN
150 PRINT#8, CHR$ ( PEEK (J) ) ;
160 NEXT
170 CLOSE 8
180 END
```

Line 125 sets the screen start address for the PETs, and 130 sets the screen end address for 40 or 80 column machines. Line 135 sets start and end for the standard VIC. SAVE this program and do a NEW. Now enter:—

```
10 ON X GOTO 120
100 PRINT "(clr)";
110 X = 1 : LOAD "0:SCREEN NAME", 8
```

RUN this and the old screen should pop back on the screen as fast as loading the same size program from disk. The cursor will remain in the home position, since nothing is actually printed. No pointers or variables are changed since it was a "dynamic load". But the loader program would RUN from the beginning, hence the ON X GOTO statement. This could be expanded to accommodate more screen loads simply by adding more GOTO data to line 10, and setting X appropriately prior to the load. The screen saver program could also be modified to store only a portion of the screen. But don't forget to change the load address in line 120, or else the files will always load back to screen starting at HOME.

This way we have a means of producing animation on your computer!

Sys'em!

Two useful SYS addresses to note:—

SYS 64790 and SYS54386

The first does a jump to warm start : like turning the machine off and on again, but without the power interruption. On the VIC it seems that almost any SYS call will produce the same result! The second is extremely handy when you want to send a machine language monitor memory dump to the printer, as the old faithful SYS 4 on BASIC 4.0 machines cancels any CMD commands previously set up.

Extra Linefeeds

In BASIC 2.0, the PRINT= command always wanted to send a linefeed CHR$(10) after the carriage return CHR$(13). As a lot of disk users will know, this can be a pain! But not always . . . some printers that don't automatically do a line advance require that linefeed character to be sent (e.g. LIST to printer). So when Commodore altered this for BASIC 4.0 a re-think was necessary. The engineers decided that logical file numbers greater than or equal to 128 would send the line feed, whilst numbers below would not. With PRINT= to the disk, you would usually opt to supress line feeds, while you could OPEN 128, 4 to do double spacing, or follow that with CMD 128 to LIST to a printer without a hardware line advance.

Harmless Bugs

It had to be Jim Butterfield! He's discovered what could possibly be rated as the most insignificant bug in DOS. He found that after using APPEND= to add a small bit of data to a very small sequential file, the block count was increased from 1 block to 2. Now this isn't possible, since the total amount of data involved was less than 60 bytes, which is nowhere near the 254 byte capacity of a block. The answer ! A Bug! It seems that DOS just assumes that the result of an append will increase any file size by at least one block. But the 'blocks free' count didn't change, indicating that the disk hadn't really used an extra block but simply incremented the block count that's stored in the directory along with the filename.

APPEN=ing large amounts of data won't cause this to happen. Evidently it only happens when the result of the append do NOT warrant the use of an extra block. When extra blocks are required for the appended data, the DOS correctly increments the block count before updating the directory.

The same bug may surface after a CONCAT of two files, depending, reasonably enough, on the size of the file being concatenated (i.e. the file that is added, not the file that is added to). Apparently the DOS uses the same routines to perform this operation.

The solution? Well, there isn't one, and nor is one really necessary. Even a COLLECT won't restore the proper block count, but at least this bug will cause absolutely no damage or side effects to your disk!

CONCATenating programs

The preceding item brings to mind another question frequently posed, namely "Why will the CONCAT command work on two sequential files, but not on two programs?". The answer is that CONCAT will not join two program files because it CAN'T MERGE two programs. What if there were a line in each file that has the same line number? The disk was not designed to deal with this problem.

However, you may say, "I could make sure that all line numbers in the file to be concatenated are high than the line numbers in the first file". The problem is . . . that is not the problem!

All PET/CBM/VIC program files end with three binary zeroes. This is so the LIST command

Programming Tips

knows when to stop listing. GOTO and GOSUB also look for these zeroes when looking for a line. If the line was not found before encountering '00 00 00', an UNDEF'D STATEMENT ERROR occurs. If you could concatenate two program files, the three zeroes that belong to the first program reside in memory ahead of code that was concatenated. LIST, GOTO and GOSUB would never look past this point.

For those doing a lot of program merging, it might be an idea to consider acquiring one of the many 'toolkit's on the market that include this function.

Turning Sound Off

If you break into a program using sound, and are being driven round the bend trying to work out how, or what, turns the sound off, here's a simple answer (12'' PETs only 'though). Simply cursor right until you get to the point on the screen that sounds the bell, and voila! After the jingle, CB2 sound is de-activated.

Cassette Notes

Jeff Kriss of Toronto has submitted the POKEs for turning the cassette motors on or off for BASIC 4.0 machines, as these are different from before. You now need two POKEs to turn them off.

Cassette = 1 :	OFF	POKE 249, 52 POKE 59411, 61
	ON	POKE 249, 0
Cassette 32 :	OFF	POKE 250, 52 POKE 59465, 61
	ON	POKE 250, 0

Another bit of tape information, this time from Ernest Blascke, also of Toronto.

When loading a program or reading a data-file from tape, quite often I forget to press the cassette deck STOP button after the tape has stopped moving. This can result in dire consequences when, later in the program, a file is opened for writing on tape, and yet the cassette is still on PLAY rather than PLAY & RECORD. As a safeguard against this happening, include a line in your program as follows (if anyone finds out the values for the VIC, can you write and let us know, and we'll include them in a later edition):—

```
10 IF (PEEK (59408) AND 16) = 0 THEN PRINT
''STOP TAPE : WAIT 59408, 16
```

Anyone using two tape drives will need these two lines:—

```
10 P9 = PEEK (241) : P8 = 59408
20 IF PEEK (P8) AND 16 * P9) = 0 THEN PRINT
STOP TAPE ='';P9 : WAIT P8, 16 * P9
```

This will eliminate any potential problems.
The above is for BASIC 1.0 ROMs. For BASIC 2.0 and 4.0 the 59408 location stays the same, but change the 241 in line 10 to a 212.

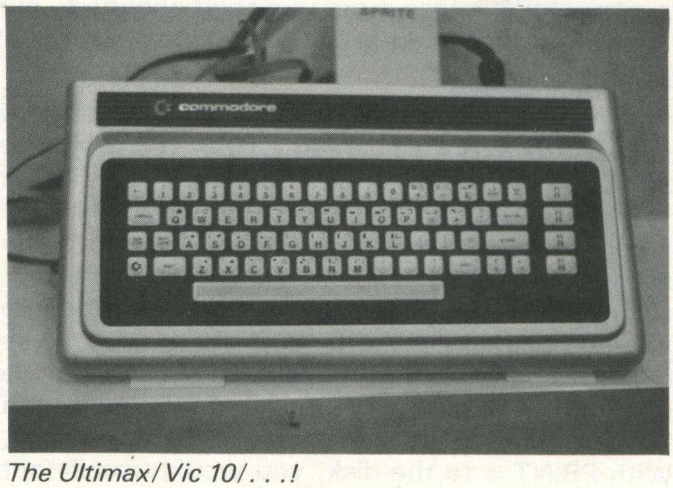
Bits and Pieces

In Midnight Software Gazette a POKE was published to suppress the question mark that follows an INPUT command prompt. Try this short program:—

```
10 POKE 16, 1 (BASIC 2.0 : POKE 41, 1)
20 INPUT ''DATA'';A$
30 PRINT A$
```

Note that line 20 prompts for 'DATA' with no '?' following, but when you hit RETURN after typing some characters, line 30 prints this string on the same line. This is a residual affect of the POKE in line 10. You might be able to use this to your advantage, but to get a line feed between lines 20 and 30 you'll have to do an extra PRINT. Subsequent INPUT commands will also have the '?' suppressed. You can get this back with POKE 16, 0. Thus the program now becomes:—

```
10 POKE 16, 1 (BASIC 2.0 : POKE 14, 1)
20 INPUT ''DATA'';A$
30 POKE 16, 0 (BASIC 2.0 : POKE 14, 0)
40 PRINT
50 PRINT A$
```



The Ultimax/Vic 10/. . .!

Pet Fourier Transplants

In part two of this two-part series on Fourier analysis and synthesis using the Commodore Pet computer and Basic, we look at how a waveform dissected using the program in Part One can be reconstituted, albeit imperfectly.

In Part One we discussed the idea that any periodic waveform can be described as a series of sinewaves, each of whose frequencies is at an integer multiple of the fundamental. Tones containing many harmonics tend to sound more interesting than those with only a few.

Compare, for instance, the timbre of a flute, which is almost pure sinewave, with that of an oboe, in which the opening and closing of the reed produces a wide range of harmonics. It is a general rule that waveforms in which there is an abrupt change in level, such as a square wave, ramp or pulse train, tend to contain a greater proportion of harmonics at the higher frequencies than those in which the changes are gradual writes *Nick Hampshire.*
That, then, is the effect of the reed snapping shut due to the back pressure in the body of the instrument. The length and volume of the instrument determines how the pressure builds-up and hence the frequency of the note we hear. The situation is complicated further by the general shape and design of the instrument which accentuates some of the harmonics while attenuating others, leading to distinctive ''colourations'' in the final tone.

Infinite range

There is an infinite range of possible periodic waveforms, but three parameters describe completely any one in terms of its harmonic sinusoidal content. First, the number of harmonics which constitute the wave form. A sine or cosine wave has only one harmonic, the fundamental. Most waveforms will be composed of an infinite series of harmonics. Fortunately, the low order ones usually contribute most to the final shape.

Some wave-shapes contain a high proportion of their energy at the higher harmonics and they will suffer more distortion in passing through a limited bandwidth amplifier than a waveform whose harmonics trail-off quickly.

Secondly, the relative amplitude of each harmonic is an essential factor in calculating the form of the result. Most ''artificial'' waveforms, such as square, triangle, ramp and rectified sinewaves, will show a progressive reduction in effect of the higher harmonics.

While the harmonics of a square wave drop off almost linearly, those for a triangle wave decrease according to square law. So the third harmonic of a square law — there is no second — will be 30 percent of the fundamental but only 11 percent in the triangle.

''Natural'' waveforms, such as the human voice or musical instruments, seldom show a neat geometric shape to their harmonic series, but will be more interesting because of it. The third and final parameter is the phase angle of each harmonic. As each new harmonic is added to the current waveform, every point where two troughs or two peaks super-impose, the resulting waveform will be accentuated; where a trough and peak overlap, the result is diminished.

So, for the synthesiser program one must first load into the memory the harmonic aplitude and phase angle of the first n harmonics, where n is sufficiently large to give a good approximation to the desired result. The next stage is successively to add each harmonic to the output waveform buffer (WV) until it is fully synthesised.

At each step it is useful to be able to plot the resulting waveform on the Pet screen. Also, as an option, to print the part-synthesised result to a hard-copy printer via the IEEE port. As our main interest is to investigate audio waveforms, a further option allows the user to POKE the resultant waveforms into a buffer in the Pet memory and then to play it back through a suitable D-A converter and amplifier.

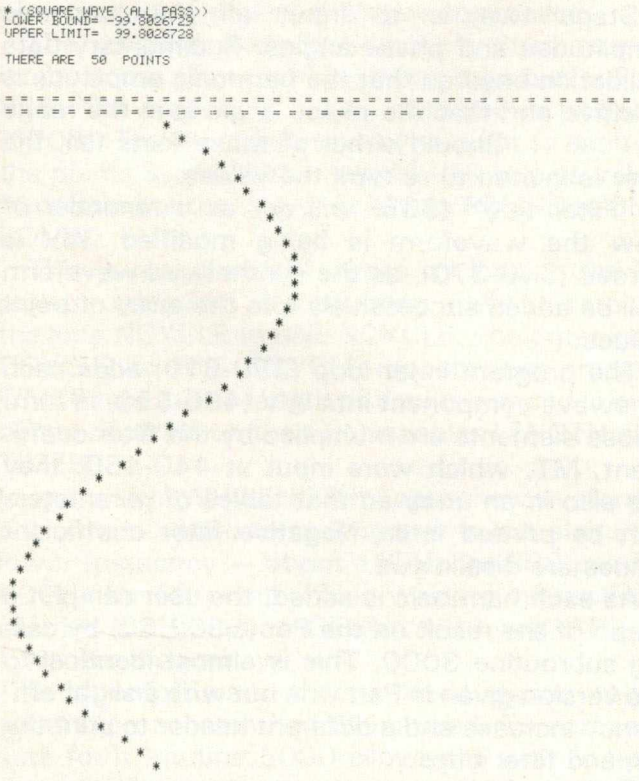
At the end of the run the user has the option to print to the hard-copy device a list of the harmonic parameters and a bar chart showing their relative amplitudes.

Primary aim

As the primary aim of this program is to show the effects of filtering on a waveform, the user is asked to enter a filter coefficient before each harmonic is added. If the filter coefficient is in the range zero to one, the harmonic is attenuated. If it is greater than one, it is accentuated or amplified.

When either the harmonic amplitude or the filter coefficient is zero, the result contains none of that harmonic and the program invites the user to proceed immediately to the next harmonic in the series. To facilitate experimentation the user may synthesise a new waveform with different filter parameters using the same harmonic amplitudes and phase angles.

Figures 1 to 4 show the system in action. There



are two methods to obtain the raw data — amplitude and angle. The first is to use analysis program given in Part One. The second is to calculate it from the series formula, which for a square wave is:

$$\frac{4}{\pi}\left(\frac{\sin x}{1}+\frac{\sin 3x}{3}+\frac{\sin 5x}{5}+\ldots.\right)$$

The harmonic amplitudes are calculated easily as being one unit of the fundamental (sin x), one-third of this at the third harmonic (sin 3x), one-fifth at the fifth, and so on. If we take the amplitude of the fundamental as 100, then it will be 33·3 for the third, 20 for the fifth.

The sample contains up to the 29th harmonic (3·448), as shown in table 1. It is easy to calculate any amplitude with this method (99th = 1·01).

Phase angles are not so obvious from the series formula but they are all the same for a square wave, and the waveform is synthesised with all phase angles set to zero.

Figure 1 shows the fundamental frequency. It is, of course, a sinewave, with a period equal to that of the final waveform. This is equivalent to passing a square wave through a perfect low pass filter with the cut-off set just between the fundamental and the third harmonic frequency.

Figure 2 shows the effect of adding the third harmonic. The peaks have been flattened-out and the sides steepened. By the time the ninth harmonic is added, the square wave is recognised easily, even though there is a considerable amount of ripple (figure 3).

After all 29 harmonics (figure 4) the tops are nearly flattened, with slight over-shoot and ripple. One could go on adding harmonics forever. A reasonable time to stop, however, is when the resolution of the graph falls below the value of the component added, or, as in this case, sooner.

People familiar with audio filters will notice that the effects of the low-pass filtering program are not identical to those observed with a conventional low-pass filter. There are many reasons for this. No electronic filter has an infinite attenuation at an arbitrary cut-off point. So some higher-frequency component will always leak through.

Furthermore, electronic filters invariably shift the phase angle of the various components. It would require only a small modification to the program to investigate the effect of phase shift in the synthesised waveforms.

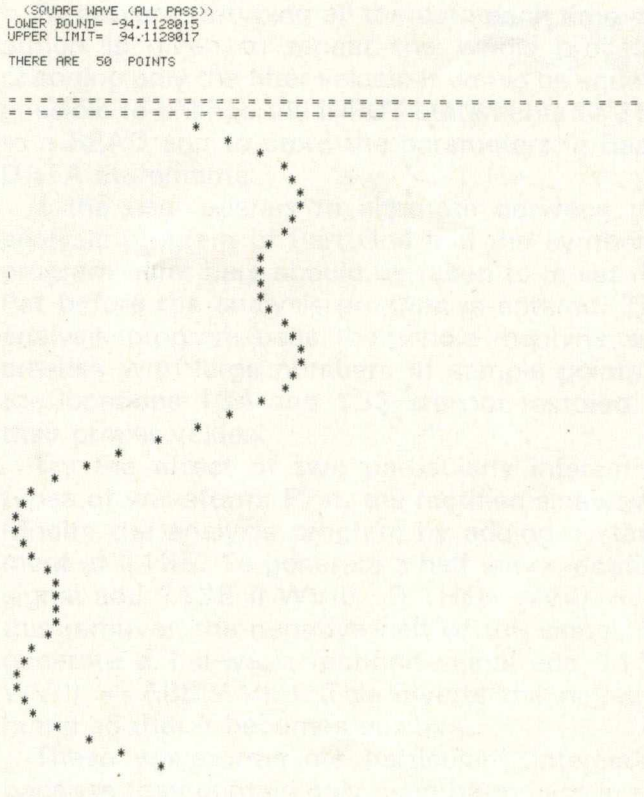


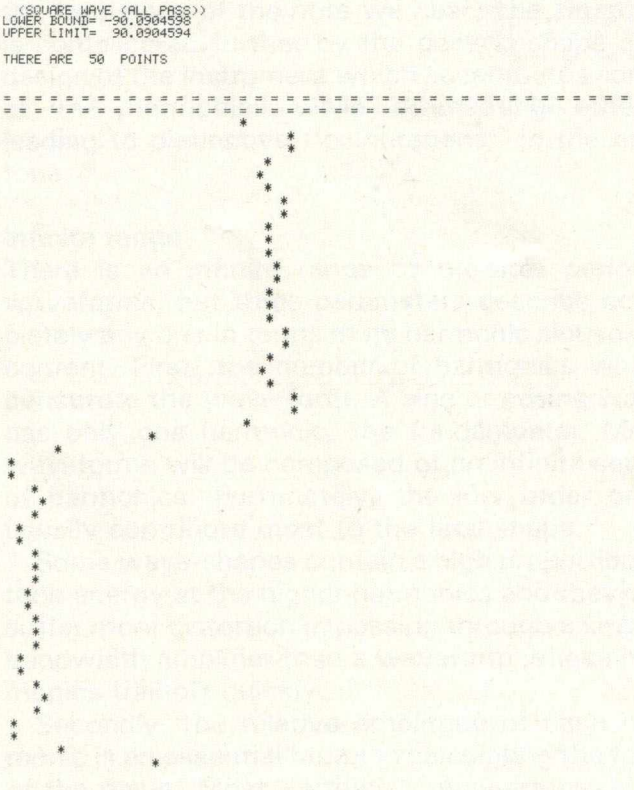
Figure 5 shows the effect of a high-pass filter on the same data. There is zero harmonic content — total attenuation — at the fundamental, third, fifth and seventh harmonic, and then zero attenuation from the ninth to the 29th, where the sample ends.

Predictable

The effect is predictable, since the higher-frequency components constitute those portions of the waveform which change most rapidly. They are the sides of the squarewave and it has been effectively differentiated into a sharp pulse. The points have been joined by hand to improve legibility. The total energy of the waveform has been reduced considerably; the points cluster about the zero line in the centre of the graph.

Overall amplitude is diminished from −90 to +90 to −42.4 to +42.4, although the plotting routine always normalises the smallest value at the bottom and the largest to the top of the graph.

The main program runs from statement 130 to 710. The user first sets-up the data arrays by specifying how many sample points the waveform is to have — this need not be the same as in the analysis program — and the total number of harmonics. This should always be fewer than half the number of points. ''Run title'' is a string assigned to the name of the waveform being investigated.



Stage two is to input all the harmonic amplitudes and phase angles. Rudimentary data validation ensures that the harmonic amplitude is positive and that the phase angle is in the range − + . Should either of those tests fail, the user is invited to re-type the values.

''Filter title'' (335) will act as a reminder of how the waveform is being modified. WV is zeroed (340-370), as the synthesised waveform will be added successively into the array of point values.

The program inner loop (380-610) adds each sinewave component into WV (490-530) in turn. Those elements are multiplied by the filter coefficient, MT, which were input at 440-450, they are also in an array so that tables of parameters can be printed later. Negative filter coefficient values are disallowed.

As each harmonic is added, the user can plot a graph of the result on the Pet (550-560) by calling subroutine 3000. This is almost identical to the version given in Part One but with a slight efficiency increase and a different header to print the run and filter titles.

As before, the user may also print a hard-copy graph (510-580), again using code (subroutine 4000) similar to that in Part One. Figures 1 to 6 are examples of graphs produced in this way.

By checking only the first character of the answers to each of the option questions Y$, both ''Y'' and ''YES'' — and ''X YETI'' probably — are taken as affirmatives and any other character or string as the negative. This is a great improvement over the ''YES'' answer required always by the code in part one.

Being able to hear the waveform produced is a bonus; subroutine 5000 is new and worth looking at in some detail. An area at the top of user memory has been reserved as a buffer into which the elements of WV will be placed, having been normalised suitably (5010-5120) to be between zero and 255.

Usually the top of memory is used by the Basic interpreter to store strings used in the program. When the Pet is first powered-up, a test pattern of bytes is stored into memory from the lowest locations until the first location which fails to return the value written into it. This location is then taken as the top of user memory. Top-of-memory address is stored in locations 124 and 135. The Pet can be fooled into thinking it is a smaller machine than it is by changing the value stored in those two locations (1-3).

Having set up the buffer, a call is made to a short machine code routine (5160) with a SYS() call which dumps the buffer out repeatedly to the

user port for digital-to-analogue conversion.

The machine code (listing 2) is three nested loops, and is stored in the unused tape buffer 2 at 826. The waveform length is stored in location (6704). The X-register is used to point to each of the points in the buffer from 6705 and its value transferred to the user port in loop ROUND to BNE ROUND.

This is repeated 255 times by loading the Y-register with 255 and counting down to zero, in the loop NCYCLE to BNE NCYCLE. The outer loop OCYCLE to BNE OCYCLE repeats that process TIMER times. The length of time the sound is produced depends on the two variables LENGTH and TIMER.

Larger arrays WV(LENGTH) cause the waveforms not only to sound for a longer time but also at a lower frequency — about 150Hz for 200 points, 600Hz for 50 points. Time-wasting instructions can be put between DELAY NOP and RTS to lower the frequency.

It is unfortunate that a fixed period tone-generation scheme had to be used. To compensate for it, routine 5000 allows the user to hear the waveform as many times as is desired. Our first idea was to make the output routine repeat forever but to check periodically the contents of Pet location 525.

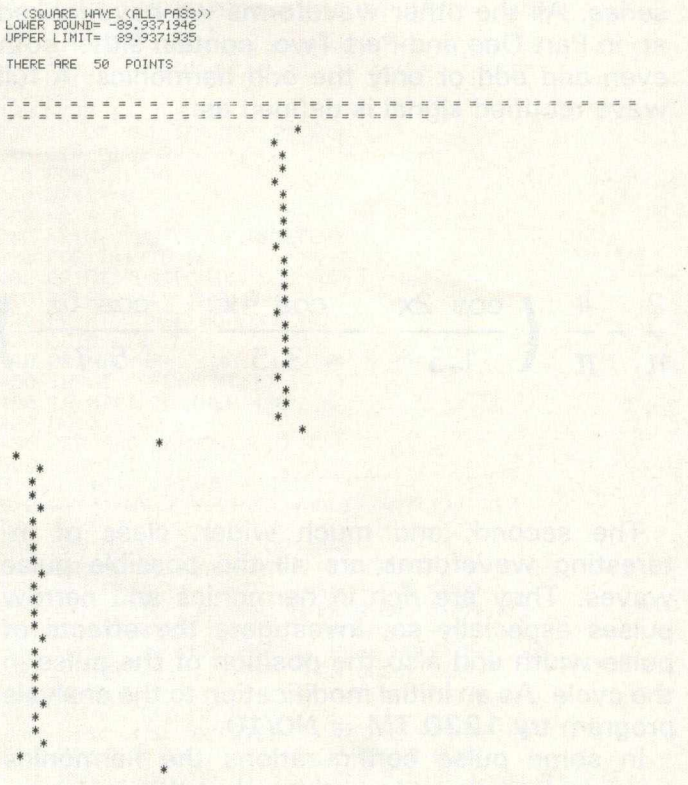
Location 525 indicates to the system how many characters still remain in the keyboard input buffer. If this is set to zero before the machine code is called, one could interrupt the waveform by pressing any key, thereby incrementing location 525. The disadvantage of this scheme is that the keyboard is scanned during an interrupt routine every 1/50th of a second as the monitor re-scans. This, in turn, means that the tone would be modulated with a 50Hz signal.

To cure this, all interrupts are switched-off in the 6502 processor with an SEI instruction. A CLI instruction at the end of the code restores the processor to its proper status.

When all the harmonics are added and the waveform is synthesised totally, the user has two more options before quitting the program altogether (710) or trying-out the data with a new filter-envelope shape (670-700).

First, a resume may be printed (630-640) which contains a list of all the harmonic amplitudes, phase angles and filter coefficients used in the last run. Table one shows an example of this routine (6000).

The slightly unusual form of the print statements is designed to protect the user from vagaries in the IEEE to RS232 interface. The second option is to print, on the hard-copy device, a



bar chart showing the relative amplitudes of all the harmonics after filtering, as in figure 7 (650-660).

Subroutine 6500 generates the bar chart in a similar manner to the code in Part One which displayed the relative amplitudes there.

To prevent re-typing all the data each time, an option is given to repeat the whole process, changing only the filter values. It would be equally possible to change the INPUT statements of 250 to a READ and to store the parameters in Basic DATA statements.

If the user wishes to alternate between the analysis program of Part One and the synthesis program here, care should be taken to re-set the Pet before the analysis program is entered. The analysis program uses the whole machine and crashes with large numbers of sample points if the locations 134 and 135 are not restored to their proper values.

Try the effect of two particularly interesting types of waveform. First, the rectified sinewave. Modify the analysis program by adding a statement at 1135. To generate a half wave rectified signal add 1135 II WV(I) 0 THEN WV(I) = 0; this removes the negative half of the signal. To generate a full-wave rectified signal add 1135 WV(I) = ABS(WV(I)). This inverts the negative hump so that it becomes positive.

These waveforms are particularly interesting because they contain only even harmonics in the

series. All the other waveforms we have looked at, in Part One and Part Two, contain either both even and odd or only the odd harmonics. A full wave rectified signal is defined as:

$$\frac{2}{\pi} - \frac{4}{\pi}\left(\frac{\cos 2x}{1.3} - \frac{\cos 4x}{3.5} + \frac{\cos 6x}{5.7}\right)$$

The second, and much wider, class of interesting waveforms are all the possible pulse waves. They are rich in harmonics and narrow pulses especially so. Investigate the effects of pulse-width and also the position of the pulse in the cycle. As an initial modification to the analysis program try 1220 TM = N0/10.

In some pulse configurations the harmonics seem to trail away to nothing, but this is deceptive; they are, in fact, part of a comb-spectrum, one which "bounces" like a rubber ball thrown obliquely against the ground.

As a final experiment, we decided to see if it would be feasible to synthesise a human voice sound The vowel sound "e" was chosen and we adopted a much- simplified model of the vocal system. Firstly, there is a voicing sound from the vocal cords — simulated here by the harmonics of a square wave.

Two hundred points were taken to give an output frequency of about 150Hz, approximately that of a man's voice.

Two filter formats were superimposed on this, one around the seveth harmonic and one around the 22nd. This looks satisfactory on paper but it has to be said that the result did not sound much like the human voice.

There are several possible explanations. A square wave is a bit too rich in harmonics, since the glottal waveform is nearer a ramp with the bottom chopped off. Also, the filter bands are in the wrong place — 1KHz and 3 · 3KHz as opposed to about 400KHz and 2KHz. More work is obviously required in this direction but once it is successful, other vowels should be simulated easily by shifting the relative positions of the lower and higher band-pass filters.

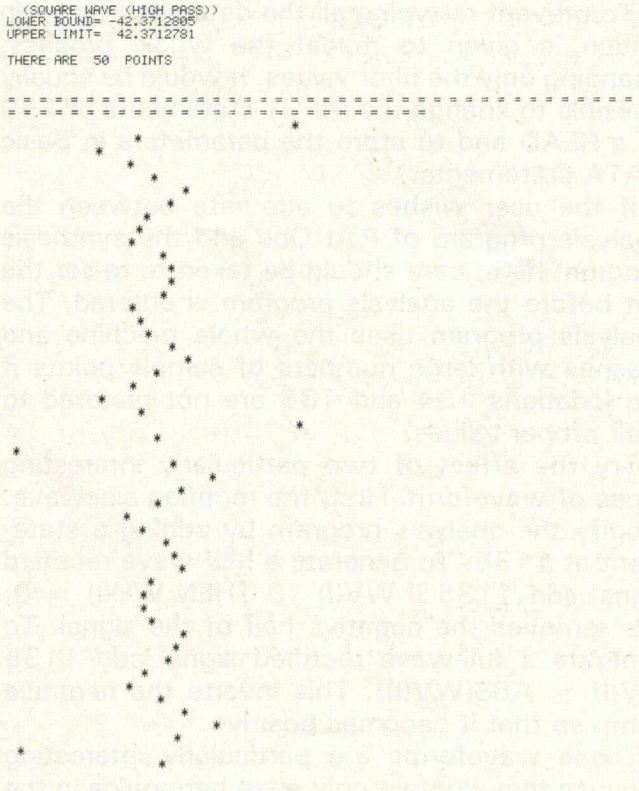


Table One

DATA FOR SQUARE WAVE (VOWEL 'E')

HARMONIC	AMPLITUDE	PHASE ANGLE	FILTER
1	100	0	0
2	0	0	0
3	33.3	0	0
4	0	0	0
5	20	0	.25
6	0	0	0
7	14.3	0	1
8	0	0	0
9	11.11	0	.5
10	0	0	0
11	9.1	0	.25
12	0	0	0
13	7.692	0	0
14	0	0	0
15	6.667	0	0
16	0	0	0
17	5.882	0	0
18	0	0	0
19	5.263	0	.25
20	0	0	0
21	4.762	0	.75
22	0	0	0
23	4.348	0	1
24	0	0	0
25	4	0	.75
26	0	0	0
27	3.7	0	.25
28	0	0	0
29	3.45	0	0
30	0	0	0

Listing 2
=========

```
                    ; CODE TO POKE 'LENGTH' BYTES IN
                    ; A BUFFER STARTING AT 6705 TO
                    ; THE USER OUTPUT PORT AT AUDIO
                    ; FREQUENCIES
                    ; PORT DD  = 59459
                    ; PORT IO  = 59471
                    ; TIMER    = 6703
                    ; LENGTH   = 6704
                    ; SBUFF    = 6705
                    ; START OF PROGRAM
0000                            *=$033A
033A   78           START   SEI  ; DISABLE INTERRUPTS
033B   A9 02                LDA #2
033D   8D 2F 1A             STA TIMER
0340   A0 FF        DCYCLE  LDY #$FF
0342   A2 00        NCYCLE  LDX #0
0344   BD 31 1A     ROUND   LDA SBUFF,X
0347   8D 4F EB             STA PORTIO
034A   20 60 03             JSR DELAY
034D   E8                   INX
034E   EC 30 1A             CPX LENGTH
0351   D0 F1                BNE ROUND
0353   00                   DEY
0354   D0 EC                BNE NCYCLE
0356   CE 2F 1A             DEC TIMER
0359   AD 2F 1A             LDA TIMER
035C   D0 E2                BNE OCYCLE
035E   58                   CLI
035F   60                   RTS
                    ; USE DELAY TO LOWER FREQUENCY
0360   EA           DELAY   NOP
0361   60                   RTS
0362

SECOND PASS FINISHED O.K.

SYMBOL TABLE
      10

PORTDD  E843     PORTIO  E84F
TIMER   1A2F     LENGTH  1A30     SBUFF   1A31
START   033A     OCYCLE  0340     NCYCLE  0342
ROUND   0344     DELAY   0360

END OF ASSEMBLY
```

FOURIER LISTING

```
1 REM LIMIT PET TO 6.7K MACHINE
2 POKE 134,44:REM POKE 52,44 FOR BASIC 2 OR 4 MACHINE
3 POKE 135,26:REM POKE 53,26 FOR BASIC 2 OR 4 MACHINE
4 DATA 826
5 DATA 120,169,5,141,47,26,160,255,162,0
6 DATA 189,49,26,141,79,232,32,96,3,232,236,48,26
7 DATA 208,241,136,208,236,206,47,26,173,47,26
8 DATA 208,226,88,96,234,96
9 DATA -1
19 PI=3.14159265
20 REM MACHINE CODE LOADER
30 READ AD
40 READ BY
50 IF BY<0 THEN 100
60 POKE AD,BY
70 AD=AD+1
80 GOTO 40
100 REM SET-UP
110 CH=4:REM PRINTER CONTROL
120 PW=67:REM PRINTER WIDTH
130 PRINT "FOURIER SYNTHESIS PROGRAM"
140 PRINT "======= ========= ======="
150 INPUT "RUN TITLE";RT$
160 INPUT "NUMBER OF POINTS";NO
170 INPUT "NUMBER OF HARMONICS";NH
180 DIM WV(NO),HA(NH),PA(NH),MT(NH),TP(NH)
200 PRINT "FOR EACH HARMONIC INPUT:"
210 PRINT "1ST - HARMONIC AMPLITUDE (>=0)"
220 PRINT "2ND - PHASE ANGLE (=-PI TO +PI)"
230 FOR I=1 TO NH
240 PRINT I;" HA, PA";
250 INPUT HA(I),PA(I)
260 REM VALIDATE DATA ITEMS
270 IF HA(I)>=0 THEN 300
280 PRINT "NEGATIVE HARMONIC AMPLITUDE - REDO"
290 GOTO 240
300 IF PA(I)>=-PI AND PA(I)<=PI THEN 330
310 PRINT "PHASE ANGLE OUT OF RANGE - REDO"
320 GOTO 240
330 NEXT I
335 INPUT "FILTER TITLE";FT$
340 REM ZERO WV
350 FOR I=1 TO NO
360 WV(I)=0
370 NEXT I
380 PRINT "BUILD UP WAVEFORM"
390 FOR I=1 TO NH
400 PRINT "HARMONIC ";I;
410 IF HA(I)>0 THEN 440
420 PRINT " HAS NO COMPONENT"
430 GOTO 610
440 PRINT "HA= ";HA(I);" PA= ";PA(I);
450 INPUT " FE=";MT(I)
460 IF MT(I)>0 THEN 490
480 GOTO 610
490 REM ADD HARMONIC TO WV
500 FOR J=1 TO NO
510 Q=I*J*(2*PI/NO)+PA(I)
520 WV(J)=WV(J)+(SIN(Q)*HA(I))*MT(I)
530 NEXT J
540 REM USER DISPLAY OPTIONS
550 INPUT "DO YOU WANT A PET GRAPH";Y$
560 IF LEFT$(Y$,1)="Y" THEN GOSUB 3000
570 INPUT "DO YOU WANT A PRINTER GRAPH";Y$
580 IF LEFT$(Y$,1)="Y" THEN GOSUB 4000
590 INPUT "DO YOU WANT TO HEAR WAVEFORM";Y$
600 IF LEFT$(Y$,1)="Y" THEN GOSUB 5000
610 NEXT I
620 PRINT "END OF RUN"
630 INPUT "DO YOU WISH TO RESUME PRINT";Y$
640 IF LEFT$(Y$,1)="Y" THEN GOSUB 6000
650 INPUT "DO YOU WISH TO SEE FILTER ENVELOPE";Y$
660 IF LEFT$(Y$,1)="Y" THEN GOSUB 6500
```

```
4050 PRINT#CH
4060 PRINT#CH,"THERE ARE ";NO;" POINTS"
4070 PRINT#CH
4080 PRINT#CH
4090 FOR L=1 TO PW
4100 PRINT#CH,"= ";
4110 NEXT L
4120 PRINT#CH
4130 TW=MX-MN
4140 FOR L=1 TO NO
4150 SP=INT(((WV(L)-MN)/TW*36)+0.5)
4160 IF SP>0 THEN 4190
4170 PRINT#CH,"*"
4180 GOTO 4200
4190 PRINT#CH,SPC(SP);"*"
4200 NEXT L
4205 CLOSE CH
4210 RETURN
4500 REM FIND LARGEST (MX) & SMALLEST
4501 REM (MN) VALUES IN WV
4510 MX=WV(1)
4520 MN=WV(1)
4530 FOR L=1 TO NO
4540 IF WV(L)>MX THEN MX=WV(L)
4550 IF WV(L)<MN THEN MN=WV(L)
4560 NEXT L
4570 RETURN
5000 REM SOUND WAVEFORM IN WV
5010 GOSUB 4500
5020 POKE 59459,255 : REM PORT OUTPUT
5030 BF=6704
5040 REM LOADS POINTS TO RAM AT 6705
5050 TW=MX-MN
5060 FOR K=1 TO NO
5070 VL=INT(((WV(K)-MN)/TW*254)+0.5)
5080 IF VL>=0 AND VL<=255 THEN 5110
5090 PRINT "POINT ";K;" OUT OF RANGE ";VL;" ERROR"
5100 RETURN
5110 POKE BF+K,VL
670 PRINT "DO YOU WISH TO TRY A DIFFERENT"
680 PRINT "FILTER CONFIGURATION ON THIS DATA";
690 INPUT Y$
700 IF LEFT$(Y$,1)="Y" THEN 335
710 END
3000 REM SUBROUTINE TO PLOT PET GRAPH
3010 GOSUB 4500
3020 PRINT RT$;" (";FT$;")"
3030 PRINT "LOWER BOUND= ";MN
3040 PRINT "UPPER LIMIT= ";MX
3050 FOR L=1 TO 39
3060 PRINT "#";
3070 NEXT L
3080 PRINT
3090 TW=MX-MN
3100 FOR L=1 TO NO
3110 PRINT "#";
3120 SP=INT(((WV(L)-MN)/TW*36)+0.5)
3130 IF SP>0 THEN 3160
3140 PRINT "*"
3150 GOTO 3170
3160 PRINT SPC(SP);"*"
3170 NEXT L
3180 RETURN
4000 REM SUBTROUTINE TO PLOT PRINTER GRAPH
4005 OPEN CH,CH
4010 GOSUB 4500
4020 PRINT#CH,RT$;" (";FT$;")"
4030 PRINT#CH,"LOWER BOUND= ";MN
4040 PRINT#CH,"UPPER LIMIT= ";MX
4050 PRINT#CH
4060 PRINT#CH,"THERE ARE ";NO;" POINTS"
4070 PRINT#CH
4080 PRINT#CH
4090 FOR L=1 TO PW
4100 PRINT#CH,"= ";
4110 NEXT L
4120 PRINT#CH
4130 TW=MX-MN
4140 FOR L=1 TO NO
4150 SP=INT(((WV(L)-MN)/TW*36)+0.5)
4160 IF SP>0 THEN 4190
4170 PRINT#CH,"*"
4180 GOTO 4200
4190 PRINT#CH,SPC(SP);"*"
4200 NEXT L
4205 CLOSE CH
4210 RETURN
5120 NEXT K
5130 REM NO OF POINTS TO LENGTH
5140 POKE BF,NO
5150 REM JUMP TO ROUTINE
5160 SYS826
5170 INPUT "DO YOU WANT TO HEAR IT AGAIN";Y$
5180 IF LEFT$(Y$,1)="Y" THEN 5160
5190 RETURN
6000 REM PRINT HA, PA & FILTER CO-EFF
6005 OPEN CH,CH
6010 PRINT#CH,"DATA FOR ";RT$;" (";FT$;")"
6020 PRINT#CH," HARMONIC    AMPLITUDE  ";
6030 PRINT#CH," PHASE ANGLE    FILTER"
6040 PRINT#CH
6050 FOR L=1 TO NH
6055 PRINT#CH,"      "
6060 PRINT#CH,LEFT$(STR$(L)+"           ",10);
6070 PRINT#CH,LEFT$(STR$(HA(L))+"                ",14);
6080 PRINT#CH,LEFT$(STR$(PA(L))+"                ",14);
6090 PRINT#CH,LEFT$(STR$(MT(L))+"                ",14)
6100 NEXT L
6110 CLOSE CH
6120 RETURN
6500 REM DISPLAY FILTER/HARMONIC BAR CHART
6505 OPEN CH,CH
6510 PRINT#CH,"HARMONIC/FILTER SPECTRUM OF:-"
6515 PRINT#CH,"   ";RT$;" (";FT$;")"
6520 PRINT#CH
6530 TP(1)=HA(1)*MT(1)
6540 MX=TP(1)
6550 FOR L=2 TO NH
6560 TP(L)=HA(L)*MT(L)
6570 IF TP(L)>MX THEN MX=TP(L)
6580 NEXT L
6590 FOR L=1 TO NH
6600 IF TP(L)>0 THEN 6630
6610 PRINT#CH,"0"
6620 GOTO 6680
6630 SP=INT(((TP(L)/MX)*PW)+0.5)
6640 FOR N=1 TO SP
6650 PRINT#CH,"*" ";
6660 NEXT N
6670 PRINT#CH
6680 NEXT L
6690 CLOSE CH
6700 RETURN
READY.
```

'Here You See Our Entry to the Far East Market.'

Machine Language Auto-Location

When a program like Supermon or Tinymon loads into its computer and RUN is given, it builds a copy of the "real" program is high memory. There's a need to do this: different computers have different memory sizes, and we want to find the top memory wherever it is. More: the computer might already have something else near the top of memory (such as a wedge program) and we want the new program to fit neatly below it.

This calls for an auto-location program. The object program must be packed into high memory. This is often more than just moving the program, since some things may need to be changed with the move. If you have a program that uses only branches — no jumps, no in-program subroutines, no tables — you may be able to get away with a simple move operation. But any instruction that uses an in-program absolute address: jumps, subroutine calls, and tables — will need to be adjusted.

We need to build a relocatable program module. Something that says, "This byte is normal so we may just move it; but that pair of bytes is an address and must be recalculated for the new location".

Ground Rules

We need a scheme which marks addresses so that the proper arithmetic may be performed. There's one requirement as to how to write the program: it may be summarized as "all addresses must be in one piece" . . .

The rule makes sense: it would be difficult to perform arithmetic on an address whose two bytes were scattered in different parts of the program. For users with assemblers, the rule translates to: never use the or functions for high and low byte.

So if we wanted to place the address of TABLE into indirect address INDAD, we would avoid coding: LDA = TABLE : STA INDAD : LDA = TABLE : STA INDAD+1. Instead, we'd define the table address in memory with TABLAD. WORD TABLE and perform the above setup with LDA TABLAD : STA INDAD : LDA TABLAD+1 : STA INDAD+1. We've used four more bytes but gained a major benefit: the two bytes representing the address of TABLE are now stored together (at TABLAD) and we can adjust this address easily when we wish to relocate.

The Method

The way we build a relocatable module is quite easy. Any time we see an address that will need relocation, we place a zero above it. As we repack the program (from the top down) the zero will signal that a relocatable address follows.

That's all very well, but, what do we do with real zeros? There will be many zeros in the program itself, and we don't want them to trigger a false relocation calculation. In this case, we change the zero to two zeros in the relocatable package. The relocation program will spot this and change it back to a single zero.

In order to do arithmetic on the addresses, we need to know where they are pointed in the first place. To relocate from $1000 to $4000, for example, we need to add $3000; but we must know that we are starting from $1000. I use the following convention: addresses are written so that the top of the program plus one is $0000 — that is, the last byte of the relocatable program is $FFFF. The program can't really go there, since that's ROM space, but it makes the arithmetic easy. We can look at an address in the relocation package as a signed number: address $FFC0 can be viewed as "64 bytes from the top of the program". If our real top-of-program turned out to be $8000, which would be correct for a 32K machine, we would translate the sequence 20 C0 FF 00 to 20 C0 7F . . . note that the zero disappears; it's the relocation flag. How did we get the new address $7FC0? By adding the relocation address, $FFC0, to the top-of-program, $8000.

Generating the Relocatable Program

How do we manufacture this package with zeros added and addresses recalculated, ready for relocation? With an assembler it's quite easy.

First, we assemble two versions of the program at two different locations. That's easy enough to do: we just change the * = statement at the start of our source code.

Then we run a simple compare program which compares the two object programs we have assembled, starting from the top. Each matching byte is copied into the relocation area unchanged; if it's a zero, an extra zero is added. If the bytes don't match, we have a relocatable address: in this case, we insert the zero plus the recalculated address into the relocation package. It's an easy

job: my "relocate builder" is a BASIC program of about a dozen lines.

Stopping

As we work down from the top we need to detect when we have reached the end of the program: this is true of both the relocate builder and the relocating program itself. There are many easy ways of doing it. The program can test to see if the last address has been reached. Alternatively, we can put some sort of "flag" into the coding itself to detect the end. In TINYMON, I use a value $BF which is never used in the program as a simple detection. A more complete method might be to use a zero with a value of 1 stored below it. It's up to you: whatever works is OK.

VIC Note

In the VIC, we have one more problem to solve. We can find the top memory (locations $37 and $38) but our program might fall into different space, depending on what's plugged in. Use pointers to find your own program (try $2D and $2E) and everything should work out nicely.

Summary

You can pick apart the code of SUPERMON or TINYMON and see how it's done. You can develop your own programs. But if you understand the principles of a relocating program package, you can develop significantly more useful programs which will adapt to a wider variety of machine configurations.

Editor's Note

The machine code dissassembly to follow is Jim Butterfields relocator modified slightly by Dave Hook. Dave eliminated the JMPs and JSRs in Jim's original utility so that the relocator can be relocated. For Vicloader, it starts at $0640, but you can move it anywhere; higher if you want more BASIC underneath it, or lower for larger object programs.

Notice that the relocator starts with the end of the object program since this will be the first byte to be packed into high memory. This is conveniently pointed at by the Start of Variables pointer minus 1, which is set on completion of the LOAD (provided it is .Saved properly).

```
0400-063F   BASIC portion (title, sys address, etc)

0640 A5 2A      LDA $2A         ;store copy of
0642 85 1F      STA $1F         ;Start of Variables
0644 A5 2B      LDA $2B         ;pointer (last byte of
0646 85 20      STA $20         ;object program + 1).
0648 A5 34      LDA $34         ;store copy of
064A 85 21      STA $21         ;Top of Memory
064C A5 35      LDA #35         ;pointer (MemTop)
064E 85 22      STA $22
0650 A0 00      LDY #$00        ;zeroise Y index
0652 A5 1F      LDA $1F         ;dec pointer to last
0654 D0 02      BNE $0658       ;byte of object prog.
0656 C6 20      DEC $20         ;(lst byte to be
0658 C6 1F      DEC $1F         ;packed)
065A B1 1F      LDA ($1F),Y     ;get obj. prog. byte
065C D0 3C      BNE $069A       ;not 0, goto $069A
065E A5 1F      LDA $1F         ;if 0, dec pointer
0660 D0 02      BNE $0664
0662 C6 20      DEC $20
0664 C6 1F      DEC $1F
0666 B1 1F      LDA ($1F),Y     ;and get next byte
0668 F0 21      BEQ $068B       ;0? yes, true zero *
066A 85 23      STA $23         ;no, relocatable addr
066C A5 1F      LDA $1F         ;store high byte in
066E D0 02      BNE $0672       ;$23.  dec pointer
0670 C6 20      DEC $20         ;and
0672 C6 1F      DEC $1F
0674 B1 1F      LDA ($1F),Y     ;get next byte
0676 18         CLC             ;recalculate lo addr
0677 65 21      ADC $21         ;using MemTop lo
0679 AA         TAX             ;result in .X
067A A5 23      LDA $23         ;recalculate hi addr
067C 65 22      ADC $22         ;using MemTop hi
067E 48         PHA             ;result on stack
067F A5 34      LDA $34         ;dec MemTop
0681 D0 02      BNE $0685
0683 C6 35      DEC $35
0685 C6 34      DEC $34
0687 68         PLA             ;retrieve hi addr
0688 91 34      STA ($34),Y     ;pack at ($Memtop) .Y=0
068A 8A         TXA             ;retrieve lo addr
068B 48         PHA             ;* save on stack
068C A5 34      LDA $34         ;dec MemTop
068E D0 02      BNE $0692
0690 C6 35      DEC $35
0692 C6 34      DEC $34
0694 68         PLA             ;retrieve byte
0695 91 34      STA ($34),Y     ;pack at ($MemTop) .Y=0
0697 18         CLC             ;rather than
0698 90 B6      BCC $0650       ;a JMP
069A C9 BF      CMP #$BF        ;last byte?
069C D0 ED      BNE $068B       ;no, goto $068B *
069E A5 34      LDA $34         ;yes, set
06A0 85 30      STA $30         ;Bottom of Strings
06A2 A5 35      LDA $35         ;= MemTop
06A4 85 31      STA $31         ;pointer
06A6 6C 34 00   JMP ($0034)     ;jmp to program
06A9 BF                         ;end detector of obj prog
06AA ...                        ;start of object prog
```

Quest for Pet. . .

NETKIT II
Universal Communications for the Commodore PET

Why buy an expensive terminal when you can use a Commodore **PET** and still be able to run normal business software.

NETKIT II is being used in many varied and diverse applications, not only as an intelligent or dumb terminal to **MINI** or **MAINFRAME COMPUTERS**, but also to support **PET** to **PET** communication with shared processing and transfer of programs and data. Numerous other applications are in use including interfacing to **NC MACHINES, PAPER TAPE PUNCHES, TELEX EQUIPMENT, HAND-HELD DATA CAPTURE TERMINALS** and other industrial and scientific equipment.

NETKIT II is the completely re-designed and upgraded version of the best selling **NETKIT** communications interface.

Unlike other software communication packages which are available **NETKIT II** is a combined **HARDWARE** and **SOFTWARE** package which provides the **PET** with an RS 232 interface and new powerful communication commands. As the software is contained in a 4K EPROM within **NETKIT II** a disk drive is not a necessity.

NETKIT II is now available for all series of **PETS** and is supplied with a comprehensive operation and applications manual.

ONLY £150 + VAT.

Yorkshire Microcomputers

28 RAMSHILL ROAD SCARBOROUGH NORTH YORKSHIRE YO11 2QF
TEL: (0723) 78136 TELEX: 527579

KINGSLEY COMPUTERS LTD.
132 Desborough Road
HIGH WYCOMBE, BUCKS HP11 2PU

CBM BUSINESS SYSTEMS

VIC HOME COMPUTERS

COMPUTER ACCESSORIES AND SUPPLIES

AGENTS FOR CBM APPROVED PRODUCTS

COMMODORE SERVICE CENTRE

Happy Birthday, KRAM!

Two years ago we introduced KRAM to the UK market, and since then several thousand copies have been sold world-wide. KRAM is approved by Commodore, and adds ten new commands to any Commodore Basic, to allow "keyed access" to disk data. You can read through KRAM files in ALPHABETICAL order of key, either forwards or backwards. For any CBM/PET with CBM floppy disk drive. (MAIOR hard disk version under test). Includes a Rom (for UD4/UD11), 40-page User Manual and demo mailing list program on disk, at £86.95.

Superkram

SUPERKRAM has all the facilities of KRAM, as well as a secondary MULTI-KEY facility allowing retrieval of data based on the contents of the RECORDS! The SUPERKRAM package includes a Rom (for the UD4/UD11 slot), a User Reference manual and demo mailing list program on disk, £146.95 (3040 disk only - 8050 available soon).

Command-o

COMMAND-O is a 4K ROM that adds 39 NEW FUNCTIONS to Basic IV Pets, including improved 'Toolkit' commands (AUTO, DUMP, DELETE, FIND, HELP, TRACE, RENUMBER), PRINT USING, MERGE, USER-DEFINABLE keys, PROGRAM scrolling, and disk commands to replace Dos Support, SCROLL, MOVE, OUT, BEEP, KILL, 8032 control chars on key. Comes with Rom (for UD3/UD12), 80-page User Manual and Quick Reference Card at £59.95. Models for 80-column, 9-inch & 12-inch 40-column.

Disk-o-pro

DISK-O-PRO for 2000/3000 series Pets is similar to Command-o, but the Toolkit functions are replaced by the complete Basic IV command set. Unlike the standard Basic IV Upgrade, most existing programs and Roms (including Toolkit) will work as before. Rom (for UD3), 80-page User Manual & Quick Reference Card at £59.95.

DTL Compiler

The DTL Compiler is the ONLY Pet Compiler that works with Basic extensions (e.g. Disk-o-pro, Command-o, Kram), and which is also completely compatible with Pet Basic. A compiled program will run up to 20 times faster, and use less memory. RRP £300 or £360 (Basic II or IV). OUR MAIL ORDER SPECIAL OFFER - £230 or £275!

Wordpro Plus

The WORDPRO range meets, and often exceeds, the facilities available on dedicated word-processing machines costing several thousand pounds or more. All our own sales literature (including this ad) are produced using WORDPRO IV PLUS. The Rom fits in the UD4/UD11 slot. Wordpro III Plus (40 column) RRP is £275.00, Wordpro IV Plus or V Plus (80-column, 32K or 96K) RRP is £395.00: OUR MAIL ORDER SPECIAL OFFER - £206.25 or £296.25!

Visicalc

VISICALC, a 'wordprocessor with numbers', is now the most popular software package ever. As well as being an excellent business aid, Visicalc is also invaluable in any organisation large enough to have a budgeting or planning activity. RRP £148 (32K) OR £170 (CBM 8096): OUR MAIL ORDER SPECIAL OFFER - £130 or £150!

Spacemaker

SPACEMAKER keeps you ahead in the Rom Race! The new Quad model allows up to four Roms to be mounted into one Rom slot, with selection of Roms from an external 4-way switch, RRP £29.95

ORDERING INFORMATION: Add 15% VAT. For same-day Access/Barclaycard service: call 01-546-7256. To order by post: write to Calco Software, FREEPOST, Kingston-upon-Thames, Surrey KT2 7BR (no stamp required). For over-the-counter sales: see your Commodore dealer. You may also order through PRESTEL. Prepaid orders are despatched by 1st Class Post, free of charge. OVERSEAS ORDERS: no VAT charge, but add £3.00 Airmail P&P (Europe) or £5.00 (outside Europe).

WRITE OR TELEPHONE FOR OUR COMPREHENSIVE DATA SHEETS

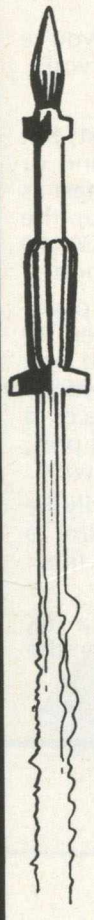
Calco Software

Lakeside House, Kingston Hill, Surrey KT2 7QT 01-546-7256

audiogenic

PROCEP CURSOR CREATIVE SOFTWARE

PET PACK SOFTWARE Ltd. Commodore COMPUTER UMI



AMOK . . . The halls of AMOK are populated by robots out of control and out to get you. To save yourself you must be quick on the draw and fast on your feet. **VP010 *£6.99.***

SIMPLE SIMON . . . Puts your dexterity and memory to test. You have to follow a sequence of flashing colour bars and tones. **VP011 *£6.99.***

VICalc . . . Turns the VIC into an easy to use programmable calculator. Ten memories are displayed on the screen along with four working registers. Using single keystrokes all normal addition, multiplication etc. and scientific functions may be undertaken as well as compound interest and percentage. Precision may be set to round up the last digit from 0 to 9 decimal places. **VP012 *£8.99.***

A-MAZ-ING . . . Needs 3K expansion. Fast action gobbler game - eat the dots and avoid the nasty ghosts. You can eat the ghosts when they change colour. **VP016 *£6.99.***

MASTERWITS . . . Try to deduce the pattern of four markers from the six colours at your disposal, while Masterwits gives you the clues you need. Rumour has it that seven tries is average!? **VP023 *£6.99.***

KIDDIE CHECKERS . . . Teaches small children to play the game of draughts. Only looks one move ahead. **VP024 *£6.99.***

WALL STREET . . . The stock market comes to life. Follow the market's daily rise and fall with cash, shares and hope. Sharp trading will win. **VP025 *£6.99.***

ALIEN BLITZ . . . How good are you at blasting aliens from the sky? Find out on the superhuman ninth level. **VP026 *£7.99.***

SKYMATH . . . Needs 3K expansion. Specially for young children! Addition and subtraction problems set in a highly entertaining audio-visual presentation. The sound and visuals provide the incentive to learn. **VP029 *£6.99.***

SPACE DIVISION (Level 1) . . . Needs 3K expansion. Another one for kids! This program sets division questions in the context of a rocket countdown. As the questions are answered correctly, the rocket prepares for blast off, until finally the child is rewarded by seeing the rocket take off. **VP030 *£6.99.***

INVADER FALL . . . Another variation on the Invader theme. This time the unpredictable aliens fall from the sky at different points across the screen. You must shoot them before they touch ground. Accuracy and quick reflexes essential! **VP032 *£6.99.***

THE ALIEN . . . Needs 3K expansion. You are the Alien. You have landed in a maze inhabited by unfriendly creatures. You must trap the creatures in your inflatable traps to avoid being eaten by them. A totally original game concept! **VP033 *£7.99.***

STAR WARS . . . A game for all you budding Skywalkers! You are the space gunner fighting off the interceptor ships of the evil Empire. **VP034 *£6.99.***

HANGMAN-HANGMATH . . . HANGMAN . . . As it suggests. You must guess the word before you get hung from the gallows. HANGMATH . . . The same except that you have to solve a mathematical problem. **VP044 *£7.99.***

MATH HURDLER-MONSTER MAZE . . . MATH HURDLER . . . Is designed to teach basic maths. If you answer wrongly the hurdler crashes. MONSTER-MAZE . . . You are in a maze and have to escape by the exit without crashing into walls or being caught by the monster. **VP045 *£7.99.***

SEAWOLF-BOUNCE OUT-VIC TRAP . . . SEAWOLF . . . You are a submarine and you have to sink as many enemy ships as possible in 60 seconds. BOUNCE-OUT . . . Full colour 'Breakout' type game. Keep the ball in play while breaking down the wall. VIC TRAP . . . A battle between you and the VIC. Try to cross the screen whilst enticing the VIC to cross your path to score extra points. **VP046 *£8.99***

CODE MAKER-CODE BREAKER . . . CODE BREAKER . . . The VIC has to guess what code or pattern you have made. CODE MAKER . . . You have to guess the code or pattern that the VIC has produced. You are rewarded with black or white tokens as to the number of guesses you have made. **VP047 *£7.99.***

NEW **KOSMIC KAMIKAZE** . . . Requires 3K or 8K expansion. Destroy the suicidal aliens as they try to land on Earth. Avoid the deadly beams from the Mothership. A fast-action Invader Fall type game. **VP053 *£7.99***

NEW **MINIKIT** . . . The low-cost high power VIC toolkit program for any memory size! This cassette program automatically relocates itself at the top of memory, taking up about 1K. Using Minikit you can enter the common Basic Keywords with just one keystroke. Toolkit commands include Find, Kill, Delete, Auto and Trace. **VP054 *£7.99***

NEW **GOLF** . . . At last - Golf in the privacy of your own home! Set your handicap and tee off - with choice of clubs and nine different holes with hazards, bunkers, water, etc. About the only thing it doesn't do is serve drinks in the clubhouse afterwards! **VP055 *£7.99***

CARTRIDGE PACKS

SPIDERS OF MARS . . . The most incredible VIC game yet! You are a trapped fly and you must shoot your way across Mars avoiding the dreaded Spiders and the other flying creatures. With classical music! **VP014 *£24.99.***

NEW **CLOUDBURST** . . . Save the Earth from the downpour of Acid Raindrops and the invasion of the mutant Cloud Hoppers. Speed and skill are essential! With original music and ten levels of play. **VP048 *£19.99.***

NEW **RENAISSANCE** . . . The ultimate Othello package for the VIC! This age-old game has been brought right up to date. You can play against the VIC with eight playing levels, or use the VIC as the board to play another person. During the games you can change sides and playing level, take back moves, set up and play special games, and even save and recall games to and from tape! **VP049 *£24.99.***

NEW **SATELLITES AND METEORITES** . . . The amazing Asteroids type game cartridge for the VIC. All the facilities of the famous Arcade game, plus . . . It uses the full screen! No more borders - the final frontier overcome! **VP050 *£24.99.***

NEW **METEOR RUN**. . . Pilot your spaceship through the Meteor belt, blasting your way through the rocks and Alien saucers. A linear type of Asteroids game! Includes early warning radar display. **VP051 *£24.99.***

NEW **BUTI** . . . The VIC programmers utility cartridge. Commands include AUTO, DELETE, DUMP, EDIT, FIND, HELP, KILL, OFF, RENUMBER, REPEAT, STEP, TRACE, UNNEW, plus Hex to Decimal (and vice-versa) converter and special VIC command which reconfigures the memory. All this PLUS A FREE 3K MEMORY EXPANSION BOARD INCLUDED! How do we do it??? **VP052 *29.99***

Now also representing HES *Human Engineered Software*

Programs include Assembler-Editor, Text-Editor,
Pet Vic commnications package and yet more games.

VIXEL from THE CODE WORKS

DINERS

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

AMERICAN EXPRESS VISA

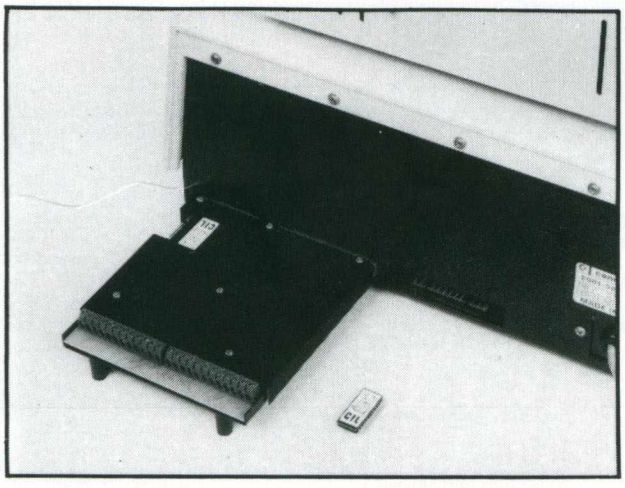
AUDIOGENIC, P.O. Box 88, Reading, Berks. Tel: Reading (0734) 586334

ADVERTISER'S INDEX

A NOTICE TO ALL

PET LOVERS

INTRODUCE YOUR PETS TO OURS
AND SOLVE YOUR INTERFACE PROBLEMS



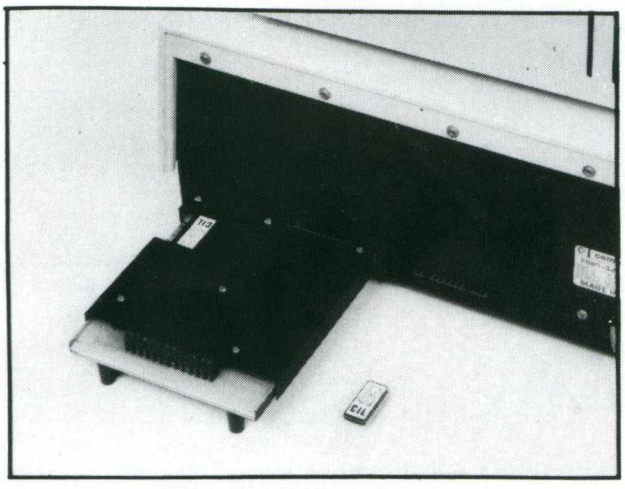
Only £195.00

ANALOGUE/DIGITAL I/O

THE PUPI

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

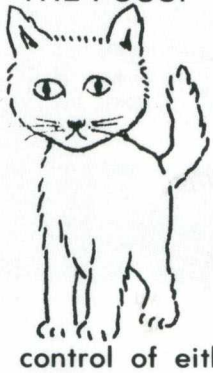
When connected to the "PET" User Port the PUPI gives you all the above features together with an operating system in EPROM, which interacts with Basic's variables, giving extremely simple operation. Inputs and outputs are ±10V and relays are rated at 10VA. Logic inputs can be used for microswitch sensing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

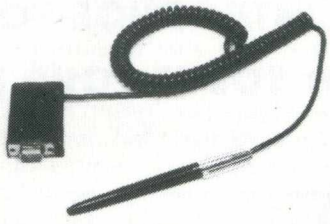
VIC-20 ACCESSORIES FROM STACK

STACK STOREBOARD (memory expansion unit)

Power up your VIC-20 to a MASSIVE 32k COMPUTER!!

only **£49.00** (plus VAT) for 3k and expandable to 27k on the same board.

STACK LIGHTPEN



Allows you to use VIC-20 without keyboard entry by simple programming. Sensor in pen sees the TV screen! Ideal for education, games, menu selection etc.

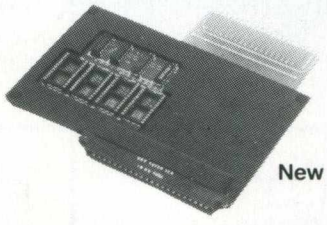
only **£25.00** (plus VAT)

STACK 8k RAMPACK

Use this upgrade pack to increase memory size on Stack Storeboard by 8k a time.

NEW LOWER PRICE!! only £29.00 (plus VAT)

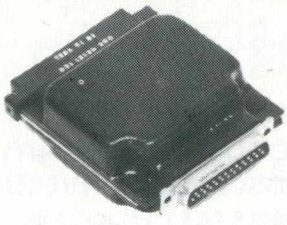
STACK ROM SWITCHBOARD



Use up to 4 ROMs at once! eg. games, ROMs, VICKIT, VICKIT II etc.

New Lower Price!! £29.00 (plus VAT)

STACK LOW COST RS232 INTERFACE



Allows you to use a serial printer with your VIC-20.

£22.99 (plus VAT)

STACK VICKIT SERIES

A series of ROMs which greatly simplifies programming and enhance the qualities of your VIC-20. Fits into Stack ROM SWITCHBOARD or Stack STOREBOARD.

VICKIT

Offers HELP to programmers....it also offers AUTO, DELETE, DUMP, FIND, OFF, RENUMBER, STEP, TRACE.

£25.00 (plus VAT)

Special Offer Price if Purchased with STOREBOARD only **£15.00** (plus VAT)

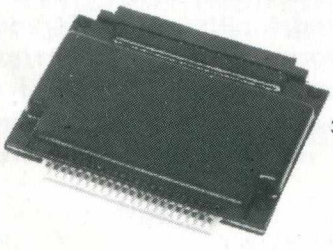
VICKIT II

A 4k ROM offering ALL THE FEATURES OF VICKIT plus.....TEXT, GRAPHICS, LINE, CLEAR, DRAW, PUT, FILL, SET, POINT

only **£29.00** (plus VAT)

Other exciting additions to the VICKIT series due soon!

STACK LOW COST 3k MEMORY



The lowest costing memory addition gives you 6½k of user memory on your VIC-20. Also allows you to use those quality games which demand 3k of Hi-Res Graphics! Socket at rear allows you to stack up further accessories.

only **£25.99** (plus VAT)

Contact your local Commodore VIC dealer for details.

Stack Computer Services Limited, 290-298 Derby Road, Bootle, Merseyside. 051-933 5511. Telex: 627026.