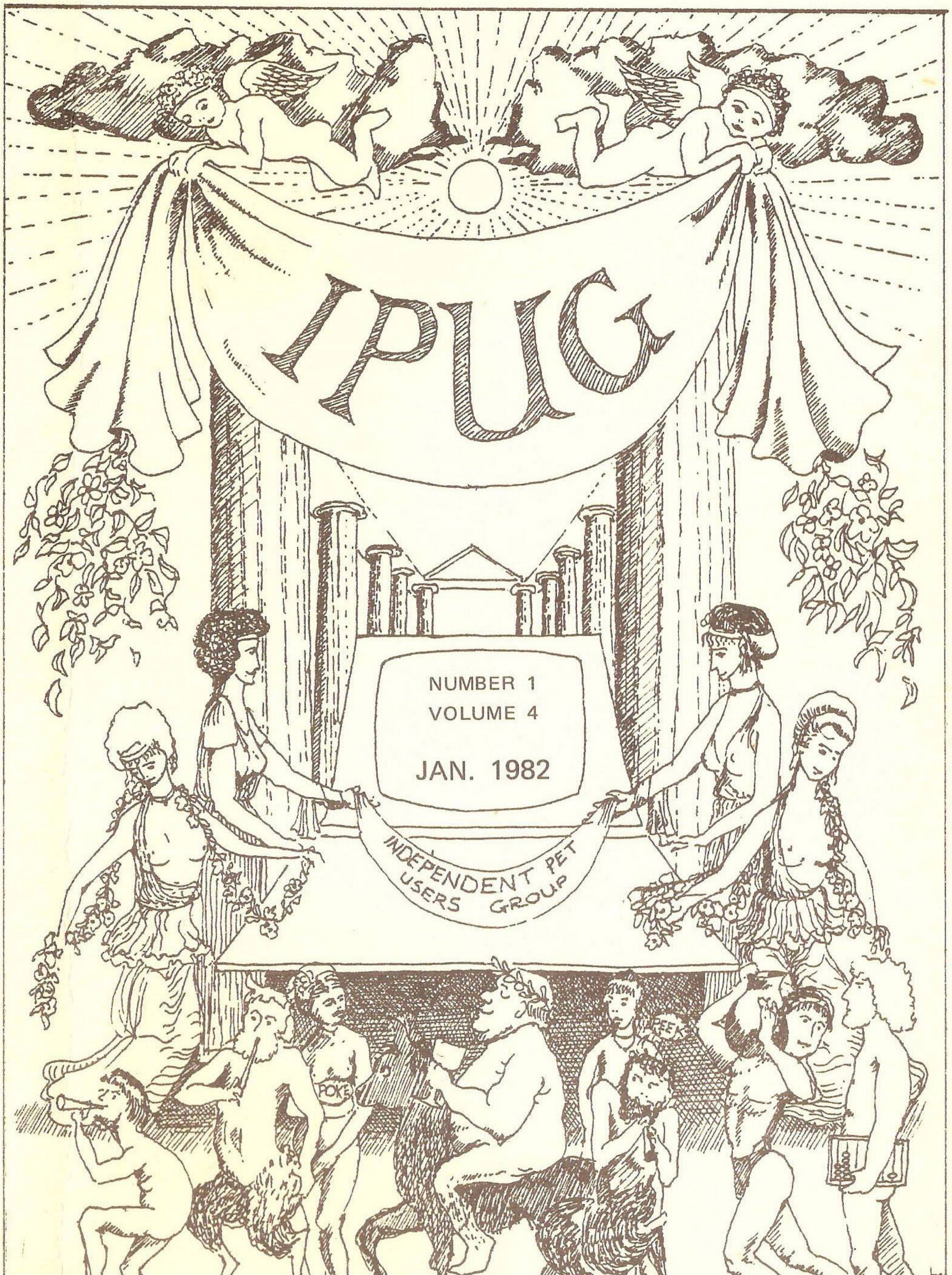


IPUG
NUMBER 1
VOLUME 4
JAN. 1982
INDEPENDENT PET USERS GROUP
POKE

NATIONAL OFFICIALS

Chairman:	Wing Cdr. Mick Ryan 164 Chesterfield Drive, Riverhead, Sevenoaks, Kent TN13 2EH Telephone: Sevenoaks (0732) 53530
General Secretary & Regional Co-ordinator:	Eli Pamphlet 7 Lower Green, Tewin, Welwyn Garden City, Herts AL6 0JX Telephone: Welwyn (043 871) 7325
Treasurer:	Joseph Gabbott
Software Librarian:	Bob Wood 13 Bowland Crescent, Ward Green, Barnsley, South Yorks S70 5JP Telephone: (0226) 811585
Membership Secretary:	Jack Cohen 30 Brancaster Road, Newbury Park, Ilford, Essex IG2 7EP Telephone: 01-597 1229
Editor:	Ron Geere 109 York Road, Farnborough, Hants GU14 6NQ
Sub-Editor:	Mike Todd 27 Nursery Gardens, Lodgefield, Welwyn Garden City, Herts AL7 1SF
Assistant Editors:	Ray Hunt Andy Rigby
Publicity Officer:	Alan Birks Flat 135, 3 Alexandra Mansions, Judd Street, London WC1H 9DQ

INDEPENDENT PET USERS GROUP

Vol. 4, No. 1 **Newsletter** Jan. 1982

Europe's first independent magazine for PET users

Page	Contents
2	Editor's Notebook
3	Matters Arising
4	Review – Wideband 'Speakeasy'
5	Upgrading 16K to 32K
6	Review – 2532 EPROM Programmer
8	4022 Printer Notes
9	COMAL – the Lowdown
20	Sales & Wants
21	VIC Matters
25	Commodore Column
25	Peeking & Poking about
26	Shop Window
28	Discounts
29	Computed GOSUB & Line Header
31	Debug
32	Software Protection
35	4000/8000-Series Investigation
39	Disk Matters
40	Strictly for Beginners
43	Review – Faster Basic

EDITOR'S NOTEBOOK

Just as the last issue closed for printing some changes occured in the Group's officials printed on p142. Since our past treasurer left the country prematurely (not with the funds I hasten to add !) his work was temporarily undertaken by Joseph Gabbott. It was logical therefore that Joseph continue to be treasurer and that Alan Birks be appointed to fill the vacant position of publicity officer.

Devotees of the Newsletter will realise that this issue is the start of Volume 4 and for the benefit of more recent PET owners, what has taken place since issue 1 ? In the beginning there was Norman Fox. His advertisement resulted in the formation of IPUG and at the second meeting, held in London, I volunteered to produce a Newsletter. At the time we had 25 members, (I was number 19) and the Newsletter was produced by typing on to stencils and typically ran to 8 pages. Some of the original 25 are still with us. In those days Commodore was a different animal, perhaps best described as negative. The PET was by todays models primitive, but with potential, and peripherals non-existent. There was a second cassette deck (rarely seen) and an advertised printer (never seen !). Some members had one on order for over a year before it was displaced by a more conventional unit. One thing was sure. You never saw a second-hand PET, at least until the 16K & 32K models were launched. Shortly after, the screen phosphor was changed from blue to green. I have one of the few of those 32K models to have a blue phosphor.

I am not a lover of 'special issue' magazines but never-the-less the heavy dose of COMAL in this issue is included as many of you will have heard much of it and want to know what to make of it. COMAL is in the public domain and hence in the Group's Software Library.

Finally, if you haven't already done so, may I remind you that 1982 subscriptions are due for renewal, and if you have may I wish you all the best for the New Year.

R.D.G.

--oOo--

MATTERS ARISING...

It's good to know that there are people hovering over every written word, it makes sure I get it right (eventually). And so it was the Trott family quick to take me to task over the MIKROBE item (p135 - Andrew wrote MIKRO). In order to illustrate the bug I presented a minimal program and in so doing let an END directive occur before the true end and contrary to the instruction booklet. If, however, one were to use NOP or RTS in the example in place of END, the 'bug' is still illustrated. One should therefore ensure that equates define a label before use.

In MIKRO the END directive is not normally necessary, but there is one unusual instance where it is essential. MIKRO assembler has as I have mentioned (pp68 & 91) similarities with BASIC, sufficient for the two to be mixed in the same program. The only reservation is that MIKRO will not assemble BASIC and the RUN command will not execute assembler. The latter is ensured by a GOTO to circumvent the assembly code while the former is accommodated by the command !A<line number> to define the start line and a mandatory END to stop the process where BASIC follows. The implications of all this is program maintenance. BASIC programs with machine code routines can be saved with the source code and be properly documented.

The Hi-Res graphics board reviewed p158 is now available as the HR-40B from Supersoft for the 'Fat-40' and the HR-80 for the 8000-series. Resolution for the 80-column screen is still 320 x 200, because the dots are doubled up and look the same as the Fat-40. The result of this is that all three boards are almost totally compatible so far as the user is concerned (Peter Calvers words, it's the 'almost' that interests me though, since all three boards are totally different !).

R.D.G.

--oOo--

REVIEW

Wideband 'Speakeasy'   Intelligent Artifacts. £ 69.00

I have recently had for review the very latest offering in the speech synthesiser market. It is called the Wideband Speakeasy and is marketted by Intelligent Artifacts. This unit enables you to program the PET to produce synthesised speech output.

The unit consists of a speaker inside an enclosure which also contains a small PCB. A cable connects at one end to an edge connector within the speaker box and at the other end to the parallel user port of the PET. It can also be used with other computers (e.g. Atom. North Star Horizon so be sure to specify the PET connectors when you buy. Having connected the cable and connected the mains you are ready to make your PET speak.

The Speakeasy uses the formant synthesis method of speech production in which sounds are built up using an electronic model of the vocal tract. A voice pattern for individual sounds is called a phoneme and words are made by combining different phonemes. Thus the limit of words that can be spoken is dependent on the number of different phonemes available. As there are 64 phonemes which can be produced by the Speakeasy. the vocabulary is very large indeed. In addition to this one can add up to 3 levels of stress or inflection to each phoneme which helps to offset the mechanical sound of some synthesisers.

Basically the sound is produced by converting the phoneme and its stress level to a code number which is then POKEd to the user port. Although it is slightly more than this it is well explained in the user manual and you should have your PET speaking very quickly.

Intelligent Artifacts also sell a program which will accept phonemes as input and code them up and POKE the user port. I combined their program with my own text processor and made a speech processor. With this I could experiment with forming phrases and editing them before coding and

POKEing the user port. When I was happy with the speech I noted the codes and added them to the program that was to do the talking ! It is all very simple once you are used to spelling words using phonemes. For example the word 'computer' becomes K.UH1.M.P.Y1.IU.U1.T.ER. It's easy, honest ! The manual has a dictionary of over a thousand words which I found very handy. If I couldn't find the actual word I needed I found one near enough to change easily.

To sum up. I found the unit neatly packaged, had an excellent manual and was very easy to use. The price is £ 69 + VAT for the unit. For orders of 3 or more the cost is £ 59 + VAT so I'm sure our discount schemer, Dave Annals, can arrange something. Very good value for money - buy it and turn your PET into ORAC !

Brian Grainger

--oOo--

UPGRADING 16K TO 32K

From letters received, many 16K PET owners feel competant to perform the upgrade to 32K, given the requisite information. Now some 16K machines have sockets which house sixteen 4115-type 1K dynamic RAM chips. Replace them with sixteen 4116 2K chips and then change the two sets of jumper links (you'd wondered about those, hadn't you ?).

The memory chips are at the front, and jumper A with 10 links is towards the rear and are linked 0110010101, where '0' is open and '1' is linked. Change this linking to 1110000101. Jumper B with 6 links is roughly central on the board. Change the pattern 100010 to 001001. When completed, power up and all being well, you should see 31743 BYTES FREE - but don't come whining to me if you don't, check what you have done (you may have some duff chips if they were cheap) !

R.D.G.

--oOo--

REVIEW

Model 2532 EPROM Programmer £47.50 incl.
Computer Interface Designs. 4. Albert Rd.
Margate. Kent. CT9 5AN.
Tel: 0843 294648

This unit was recently purchased but it was found that it works quite well but lacks adequate documentation. The following notes may be of use to users:

The unit comes complete with cassette software. but minus mains plug. A sketch on the instruction sheet shows how to connect the unit to the PET IEEE port. After loading the program from tape. enter RUN. One is then prompted to key '7' for the 2716 EPROM or key '5' for the 2532. after which a menu appears. Do NOT try these options in numerical order. Insert a blank EPROM. switch the unit to the type used. and set the switch to 'READ'. Power up the unit and select '3' on the menu. This can only be done when there is no DATA stored at $1000 to $2000 in memory. or it will give 'EPROM DATA WRONG'. If the EPROM is unprogrammed (all addresses = $FF). it will indicate 'EPROM DATA CORRECT' and after about three seconds. it will return to the menu. Select 2 and follow screen. To read the contents of an EPROM. switch off the unit. insert the EPROM. set to 'READ' and restore power to the unit. Press '1' and the EPROM is read to PET memory starting at $1000 up. The menu reappears after about 1 second. press '7'. then key M 1000 2000 return.

Select 4 on menu to place $FF from $1000 to $2000. Select '7' on the menu then enter 'M 1000 1080 return. Now edit all the FF's to the code you wish to save on EPROM. If not completed. save it on tape or disk using the normal 'monitor' .S save format. eg .S"TITLE",01,1000,1081 - note the last number is one more than the last address to be saved. Subsequently it can be loaded as usual and continued. When all data is in PET memory. select '6' on the menu. switch the unit off. insert blank EPROM. switch to 'PROGRAM'. switch unit back on. Press '1' to burn in EPROM. During this the yellow LED will flash. After four minutes for a 2K 2716 EPROM the 'PROGRAMMING' display says

'NOW SWITCH TO READ'. After doing this. press '1' and the menu returns. Select 5. press 1 and data in EPROM is 'verified' with that in memory and 'DATA IS CORRECT' should appear for about 3 seconds. then the menu. Press 8 to exit the program. If you select 1 you will again be given the request to key 7 for a 2716 EPROM.

Note: if you use 1K of input, providing all the other bytes are set to $FF you can add more data later.

Leslie Reeve

--oOo--

MAIL SPOT

At the end of the third week of October, I 'phoned Microtech of Pennsylvania for details of 'Petdiskii', and on the same day. ACM of North Cheam for details of two of their products. so far without result. Also on the same day I ordered a 15-A-28 BASIC Switch from AMS of Indiana, which finally arrived in the last week of October. It was installed in about an hour and I now have a small-keyboard 8K old-ROM PET, with key-tops worn to stubs, updated to 32K with BASIC Switch containing original & Upgrade ROMs and with 'ARROW' and 'FAST BASIC' customised on a 4K chip in the 15th socket (I do not have disk drives).

Snags ? Everything worked first time, except Fast Load and Save. which is what I bought the gear for in the first place ! But then I remembered trouble with 'RABBIT' (p30, Jan '81), and sure enough my cassette deck was 1977 vintage, but a quick swop with a co-operating dealer enabled Fast Load, etc.

Advantages ? Programs load on average 12 times as fast. and run 37% faster.

E.W.Thomas, 47, South Road,
Weston-super-Mare, BS23 2LX.

--oOo--

4022 PRINTER NOTES

By Nick Higham

The 4022 printer superseded the 3022 several months ago. There are many differences between the two, the most obvious being the more smoothly styled appearance of the 4022. The 4022 has a lift-up perspex cover which allows easy access to the ribbon and tractor guides - by keeping the cover up, the output can be seen as it being printed.

The 4022 is slower than the 3022, but it makes up for this with the following features:
(1) It is possible to switch the printer into and out of upper/lower case mode by:

```
          POKE59468,14:OPEN7,4,7:PRINT#7:CLOSE7 and
          POKE59468,12:OPEN8,4,8:PRINT#8:CLOSE8.
```

In each case the printer will print in the same mode as the screen.
(2) The printer can be reset with:

```
          OPEN10,4,10:PRINT#10:CLOSE10
```

(3) An 8x6 dot matrix is used (3022 had 7 x 6). True lower case descenders are printed, as are all 8 of the vertical lines (e.g. shift-T). The characters on the whole are a bit 'fatter' than those on the 3022 and the full stop and semi-colon are heavier.

Bugs: (1) PRINT#4,CHR$(254) produces a space instead of the shifted '>' graphic character.
(2) When the printer is in lower case mode, key and shift-key are swapped round, where 'key' is [ ] ← or ↑.
Several other differences are catalogued in the Sept. 1981 issue of Commodore Club News.

Note that the 4022 manual is WRONG in several places. In particular for setting the line spacing with OPEN6,4,6:PRINT#6,CHR$(N):CLOSE6, N=24 gives vertically contiguous graphics and N=36 is the default value. Also, the explanation of the programmable character on p29 has not been updated to take into account the extra row of dots on the 4022 matrix head !

In summary, the 4022 is a very pleasant printer to use and some users may find it worth upgrading from the 3022.

--oOo--

COMAL - THE PROGRAMS

By Brian Grainger

The purpose of these articles on COMAL is threefold. Firstly to indicate how to use the various COMAL programs that are circulated by IPUG. Secondly to show how to use COMAL80 with emphasis on the differences from BASIC. Finally I will identify problems I have found with the current version of COMAL80 (0.11). In these articles I will go through use of all the COMAL commands which are different in use from BASIC. Because space is short in the newsletter I will give COMAL examples with BASIC equivalents and leave you to experiment. Any commands I do not mention in the articles can be assumed to be used as in BASIC.

THE PROGRAMS.

COMAL80 is a BASIC4 version of the complete COMAL interpreter and executer. To use the program just LOAD and RUN and PET is turned into a COMAL machine with just under 5000 bytes free.

COMAL80IN is a BASIC4 version of the COMAL input module. To use the program just LOAD and RUN. PET will allow input of COMAL programs with about 15000 bytes available for program and variable storage. When the COMAL program has been input type RUN and the program will be stored on disk. You will then be asked if you wish to execute the program. If you answer NO, you can return to input more COMAL programs. If you answer YES, then COMAL80EX will be loaded from disk and executed automatically.

COMAL80EX is a BASIC4 version of the COMAL executer module. It is used to execute COMAL programs prepared by COMAL80IN. To use the program just LOAD and RUN. You will be asked for the filename of the COMAL program file. This file will then be loaded from disk and executed.

COMALERRORS is a file which is accessed from disk whenever the COMAL interpreter or executer wants to print an error message. This file can be created by running the BASIC program COMALGENERRORS.

COMALB3 is the BASIC3 version of COMAL80. COMALB3IN is the BASIC3 version of COMAL80IN. COMALB3EX is the BASIC3 version of COMAL80EX -
NOTE:- COMAL for BASIC3 has some BASIC4 PET routines added to it which do not exist in BASIC3 PETs. Consequently storage space for programs and data is less than that with BASIC4 COMAL (not much less!). More important the start of program space for BASIC3 COMAL is different for that of BASIC4 COMAL. This means that programs SAVEd from BASIC4 COMAL cannot be LOADed from BASIC3 COMAL and vice versa. Provided the program length is OK one should LIST the programs with BASIC4 COMAL. The program can then be loaded with BASIC3 COMAL by using the ENTER command.

As indicated above COMAL accesses a disk to print error messages. I have prepared some versions of COMAL80 and COMALB3 which print an error number instead of calling a disk. This will allow those with cassette based PET systems to try out COMAL. If anybody should wish a copy please send me a blank cassette, the stamps for return postage and tell me what BASIC your PET uses (new ROM BASIC3 or BASIC4). I will send the copy by return. The address below can also be used if you have queries on use of COMAL or have found some interesting ideas in using COMAL.

Please write to me at 73, Minehead Way, Stevenage, Herts. SG1 2HZ

--oOo--

THIS ISSUE'S THOUGHT

If builders built buildings the way that programmers wrote programs, then the first woodpecker that came along would destroy civilisation.

--oOo--

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 than A	N.B. This uses same storage 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 the BASIC equivalents. Some of the COMAL commands shown above have extra uses. See the article on features unique to COMAL.

--oOo--

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 COLOUR$ OF
0020 WHEN "RED","YELLOW","BLUE"
0030  PRINT COLOUR$;"IS A PRIMARY COLOUR"
0040 OTHERWISE
0050  PRINT COLOUR$;"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 NAME$ OF 10 declares a string of maximum length 10 characters.
e.g. DIM NAME$(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 I, B 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 NAME$ OF 9
          0010 NAME$:="FREDERICK"
          0020 A:="RICK" IN NAME$
          0030 B:="DAVE" IN NAME$
          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 list of strings are being input, it does mean that each variable must be separated by <return>.

11) The command LABEL will cause a label to be placed on a line. For example you can type either of the 2 statements below and the COMAL interpreter will accept it in the same way:

```
          0010 LABEL START or
          0010 START:
```

When listing the program it is the second form that will be listed. We can now use the label START as a reference in a goto statement:
N.B. No other statements can occur on a labelled line.

12) The LIST command under COMAL has 2 special features. Firstly it invokes automatic indenting of lines for readability. It is not necessary to type the lines indented it is done automatically. Secondly one can LIST to a file on the disk or cassette. This file can subsequently be merged into any existing program by the ENTER command:
LIST"FILENAME" lists the current program to disk under the name FILENAME.
LIST"FILENAME",1 lists the current program to cassette under the name FILENAME.
LIST 10-50,"FILENAME" will list lines 10 to 50 inclusive to disk under the name FILENAME.

13) The RENUM command will renumber an entire program. It has the same syntax as the AUTO command.

14) The RUN command is similar to that of BASIC with one notable exception. Variables are NOT reset to zero or null. Thus no assumptions should be made on the values of undeclared variables in a program.

15) In comparing the BASIC PRINT command with the COMAL equivalent, COMAL looks clumsy. In reality COMAL is much more flexible. BASIC has fixed print zones of length 10. In COMAL one can vary the zone length by the ZONE command:
e.g. ZONE:=5 will cause items separated by commas in print lists to be printed in zones of 5-character length.

The default value of ZONE length is 0 and not 10 as in BASIC. Thus a comma will cause no separation of variables until zone length has been redefined. In COMAL a semicolon between print items will ALWAYS print 1 space between those items, whether they are numeric or string. It is a true space unlike BASIC which for numerals will print a cursor right after the value. This appears like a space but in reality is not.

16) In COMAL variable names or labels can be up to 16 characters in length. The first character must be a letter and all 16 are recognised unlike BASIC where only the first 2 are recognised. In COMAL it does not matter if a variable name includes a COMAL keyword. Thus variable names can be chosen which have meaning. One side effect of this is that COMAL keywords must be typed with a following space. Otherwise it will be regarded as a variable name and a syntax error will probably result. One other COMAL restriction on variable names is that the same name CANNOT be used for two different variable types. For example the variables A and A$ cannot appear in the same segment of program.

17) I have mentioned above that one need not type the spaces necessary to cause line indentation. They are added automatically when the program is LISTed. There are other things that need not be typed as well. Here is the complete list:
The ':' preceding the '=' in assignment statements.
The 'OF' in 'CASE...OF' statements.
The 'DO' in 'FOR...DO' or 'WHILE...DO' statements provided the 'DO' is the last word in the statement.
The 'THEN' in 'IF...THEN' or 'ELIF...THEN' statements provided the 'THEN' is the last word in the statement.
The procedure name following the 'ENDPROC' statement.
In all the above cases the COMAL interpreter will automatically add the relevant words if they have been omitted.

18) The 'REM' command is accepted in COMAL but the interpreter will replace it with //. REM or // can be used at the end of a COMAL statement or be on a line of its own.

19) It has been stated above that variables cannot be assumed to be null at the start of a program. There is therefore a need for a command which will set a string variable to a blank. The following example will illustrate:
SPACE$(1:60):="" will set the string SPACE$ of sixty

characters length to a blank. Another facility on string handling is that if a single character is to be referenced it can be done simply as the following example shows:
NAME$(5) refers to the 5th character of NAME$. It is not necessary to say NAME$(5:1). Thus when the length value is 1 it may be omitted.

As can be seen from the above COMAL has many features that BASIC does not have. I have left until last one of the most important features:
There is syntax checking on input as there is with ZX80/81 BASIC (wash my mouth out) ! The interpreter will even leave the cursor at the point where the error is recognised so that the line can be modified quickly before continuing. This means all the statements of a COMAL program are checked for syntax before it is RUN. In BASIC they are not checked until run time and it is up to the programmer to ensure all the lines are executed and checked. This is not at all easy on long and complicated BASIC programs with many loops and branches. To sum up COMAL is an extremely powerful language compared with BASIC which results in easy to read, easy to maintain programs.

B.D.Grainger

--oOo--

COMAL - BASIC COMMANDS WITHOUT COMAL EQUIVALENTS

By Brian Grainger

In this article on COMAL I want to do two things. Firstly I want to identify those BASIC commands which are not included in COMAL and do not have COMAL equivalents. Secondly I want to identify some problems I have found in using COMAL (rev 0.11) whether in using the BASIC2 or BASIC4 version.

Here is a list of BASIC commands not supported by COMAL:

BACKUP, CLR, CMD, COLLECT, CONCAT, COPY, GET, HEADER,
ON...GOTO, RENAME, SCRATCH, VERIFY, WAIT,
POS, STR$, TIME, TIME$, USR & VAL.

In addition to the above commands. shortforms are not allowed in COMAL. In particular '?' does not mean PRINT.

It will be seen that most of the commands not implemented are BASIC4 disk management commands. The equivalent BASIC2 sequences also do not appear to exist. I find this somewhat surprising as the means to decode the commands is available in the BASIC ROM.

Here is list of oddities I have found in using COMAL:
1) The command BASIC causes a system error although it is a valid COMAL command.
2) PRINT USING is not implemented. In my opinion the saddest point about using COMAL.
3) SAVE or LOAD to cassette does not work. The commands function but on reloading the file reads as gibberish.
4) When a program with indented lines is ENTERed an error occurs on lines with DIM statements. Just ignore the error. Type <return> over the displayed line and all will continue normally.
5) When a program is ENTERed from tape an EOF error occurs as a matter of course. It is not really an error.
6) ZONE:=1 does not work correctly (it works as if ZONE:=2). Use ZONE:=0 and it works like ZONE:=1 !
7) Do NOT send disk commands when a disk is not switched on. A system crash will occur if you do !
8) DEBUG, while being a COMAL command causes a system error. If anybody finds any other problems or has any suggestions as to resolving the above or why they happen, please let me know. Write to me at 73, Minehead Way, Stevenage, Herts. SG1 2HZ.

--oOo--

MEMBERS SALES & WANTS

Small keyboard PET for sale, upgraded to 32K from 8K. Enquiries to Jeremy Jacob, Computer Department, Sigma Resources Inc., 30, New Bond Street, London, W1Y 9HD. or telephone during office hours: (01) 499 0963.

--oOo--

VIC MATTERS

by Mike Todd

Well, it's 1982 - a year in which there should be many developments in the VIC world. 40 columns, disk, printer and a many add-ons should be available before the year is over.

The first news is about the VIC itself - Commodore say that there should now be over 10.000 VICs in the UK, and by the time you read this dealers should be in a position to sell them off-the-shelf. They also tell me that the VIC disk drive is now available (395 pounds) and the good news is that it is basically half a 4040 which means that its disks should be totally compatable with the PET. It is unfortunate that the VIC can only support one drive, although the add on IEEE interface should allow several drives to be connected just like on the PET. The 80 column VIC printer should also be available this month at about 200 pounds.

The other good news is that Commodore hope to have a 40-column add on by the time you read this. It is not designed by me (despite what you may have read in the press !) and will not convert the VIC into a true 40 column machine. It is marketed by B&B Computers of Bolton and will cost just under a hundred pounds. It does not replace the VIC chip but is instead an add on box. The operating system will still think it's got 22 columns and that is not easily changed without a major rewrite of the screen handling routines. The 40 column board will be the heart of a VIC/PRESTEL package and is a Commodore approved product.

This is being written at the begining of December as rumours are flying that there will be a 40-column VIC some time in 1982. However, the upgrade is expected to be more than a simple upgrade of screen width - several other features are expected to be added.

The VIC chip itself is one of a family of video controller chips. The 6560 and 6561 are the current VIC chips, the former being NTSC (for USA use) and the other for PAL (Europe); there are two other chips in the range, the 6562 and 6563 (again NTSC & PAL) which provide a screen resolution of 300x200 instead of 190x200 in the 6560/1.

Following my column last issue, I had expected some response from IPUG members who have VICs. Although I received a large number of letters, not one related to the VIC! Is it that IPUG members don't have VICs? Don't forget that it is hoped that local PET user groups will act (at least initially) as VIC user groups and that we hope to have VIC user groups active fairly soon. If there is no VIC interest in IPUG then so be it, but if there is, please let us know!

In my last column I mentioned the odd memory map for the VIC; well. it comes in three varieties:

```
$0000------------    $0000------------    $0000------------
     workspace            workspace            workspace
$0400------------    $0400------------    $0400------------
      empty                                      unused
$1000------------         BASIC TEXT       $1000------------
     BASIC TEXT                                 video RAM
                                            $1200------------
$1E00------------    $1E00------------           BASIC TEXT
     video RAM            video RAM
$1FFF------------    $1FFF------------    /////////////////
```

These memory maps show the three possible combinations - the first is the bare VIC; you will see that there is a gap from $0400 to $0FFF. In the second, this space has been filled with the simplest memory expansion of 3k. The third shows what happens as soon as RAM is added after $2000 - the video RAM immediately moves to $1000 and BASIC text now starts at $1200. All this is set up on reset when a RAM check (non-destructive, unlike the PET) identifies the lower and upper limits of RAM and sets the map accordingly. It would appear not to allow for RAM at $0400-$0FFF if RAM is added above $2000. Although this is not accessible to BASIC, it could provide a useful home for machine code. If you are in any doubt as to what RAM is where, the following locations may help:

$0281/2	(641/2)	- address (lo/hi) of lowest RAM location
$0283/4	(643/4)	- address (lo/hi) of highest RAM location
$0288	(648)	- page number (ie high byte) of screen RAM
$2B/C	(43/4)	- address (lo/hi) of start of BASIC RAM
$37/8	(55/5)	- address (lo/hi) of end of BASIC RAM

The uncertainty of which set up any particular VIC may have will cause problems when accessing RAM directly, not only in machine code programs but if programs access the video (or indeed the colour screen RAM which also moves!) then they should determine where it starts before PEEKing or POKEing. Note that if the video RAM starts at $1000, then the colour RAM starts at $9600 - if $1E00 then it starts at $9400.

LOADing programs will also cause problems since they may have been SAVEd on a machine with a different configuration. To get round this problem, all normal LOADs from cassette or the serial port (LOAD from RS232 is not allowed) will begin a LOAD at the start of BASIC TEXT. However, this is not much good for machine code programs or other LOADs which are memory position dependant. To allow for this, the format of LOAD is - LOAD"filename",device,mode where mode is '0' for normal "relocate" LOAD (the default mode) or '1' if you want the program to LOAD at the same address it was SAVEd from.

To make life a little easier, SAVEing to cassette allows for the programmer to specify if a relocated LOAD or an absolute LOAD is required. To do this, the same format is used as for LOAD, except that it only works for cassettes. Mode 0 allows a simple LOAD to relocate the program, while mode 1 forces an absolute LOAD.

This is done by using a new program header identifier. At the start of every program on cassette is a header, with the filename, start and end addresses, and a program identifer which is normally 01 on the PET or if SAVEd on the VIC using mode 1. If a LOAD finds the identifier to be 1 then it will perform an absolute LOAD. However, a new identifer (03) is used to indicate that a relocated LOAD is required and this is the identifer used for a normal SAVE on the VIC.

It is important to note that the PET always SAVEs using the identifier 01, which means that any programs developed on the PET will always relocate unless the absolute LOAD mode is used. Similarly, any programs SAVEd on the VIC using normal SAVE will not LOAD on the PET since the PET does not recognise the 03 identifier.

Turning now to the SYS command, the VIC has a very useful facility which allows the X, Y, accumulator and status registers to be passed to and from BASIC. This is done by POKEing 780 with the accumulator value, 781 with the X, 782 with the Y and 783 with the status register. As soon as the SYS command is entered, the registers are loaded with these values, the routine executed and then the new values returned in the same locations. Thus it is possible to pass parameters relatively easily to VIC routines without having to write machine code to do it !

Finally. two problems with the VIC that may be of interest. The first concerns a slight misunderstanding of the function keys. These are not soft-keys as you may have been led to believe. Instead, they merely generate ASCII characters which have to be detected through the GET command, although machine code software could allow these keys to be actioned directly. The ASCII codes for the function keys are:

f1=133 f2=137 f3=134 f4=138 f5=135 f6=139 f7=136 f8=140

One serious bug has been reported by Commodore and that is in the RS232 handling software. Release versions of the VIC with KERNEL ROM number 07 have this problem, although Commodore hope to release a new ROM when all the other problems are ironed out. The bug occurs during multi-line handshaking (the 3-line handshaking is fine) and when a specific sequence of signals occurs, the VIC can hang waiting for a handshake signal that will never occur.

Well, that's it for this issue - don't forget, if you've anything you want to say about the VIC, or just want to borrow IPUGs VIC for a regional group demonstration, or want to start your own VIC user group - then drop me a line, my address is inside the front cover.

--oOo--

COMMODORE COLUMN

The big gap between storage capacities of floppy disks and the 22M-byte 8422 Winchester doesn't look like being filled until the Spring when CBM launch a US-built 5M-byte 5¼ inch Winchester with a built-in 1M-byte floppy for back-up.

Odd that the MMF9000 Micro-Mainframe, which CBM have dubbed the 'SuperPET' because that's what everyone else refers to the 8032 as, does not appear in the current price lists, even though it has a brochure of its own. I understand the price is around £ 1,500.

Official figures now put the number of CBM systems installed in the UK as over 40,000.

Latest piece of hot news (well it was when I heard it) is that the 8024 printer, that is the 132 column, 160 160-char/sec job, and the 8027 daisy-wheel printer has been discontinued, with the 8026 keyboard version to follow suit. According to my reckoning that puts us back almost to square one leaving the somewhat dubious 4022 matrix printer. Now if that isn't enough, the 8010 modem looks doomed since Livermore, the manufacturer, has been taken over. It seems likely that the new owners will deem the modem insufficiently profitable.

R.D.G.

--oOo--

PEEKING & POKING ABOUT

Temptation. I am sure was never meant to be resisted, so try this. LOAD a BASIC program, then POKE19,157:LIST - users with old ROMs do not get the same effect poking location 101. Like it ? Will explain further next issue....

--oOo--

SHOP WINDOW

If you took a fancy to the Machsize Mini-rack (photo p113 Sept '81) there is now available a cased version. Looks remarkably like the case of a 3040/4040 disk unit in style. The system (see p109) has been expanded, the card frame being available as 5- or 10-slot, and the range of plug-in board options increased. Tel: (0926) 312542/32399.

The robot arm from Colne Robotics briefly mentioned on p110 and known as the Armdroid has been joined by a new arm able to lift 1lb rather than 10oz. The Armdroid costs £ 199 in kit form, the new arm is 'under £1,000 in kit form. Colne Robotics are at 1, Station Road, Twickenham, Middlesex. Tel: 01-892 7044.

Smart-Arms is the name given to Systems Control's range of robot arms having models handing up to 2Kgm in a price range £ 430 to £ 2560. Software is available for the CBM/PET together with an interface equipped with servo-motor drivers and additional I/O ports. Eight models to choose from, contact Systems Control, 30, Thirsk Road, Northallerton, North Yorkshire, DL6 1PH. Tel: (0609) 70643.

Another robot, known as the 'Small Arm' comes from Sands-Whiteley and is priced at just under £ 500. It has been designed for the PET and the BBC micro. Sands-Whiteley Research & Development Ltd., are at Royston, Herts. Lifting capability is 250g and it can pick & place to within 1mm.

The Model 845T from Wessex Electronics is a programmable digitally synthesised crystal-controlled oscillator, controllable from the IEEE-488 bus. Details from Wessex at 114-116, North Street, Downend, Bristol, BS16 5SE. Tel: (0272) 571404.

A networking system known as Hydra developed by Wordcraft links up to 225 PETs and allows disk drives and printers connected to one of them to be shared. Programs may be LOADed or SAVEd and screen displays may be interchanged. The network link operates over a distance of 1km at up to 250K baud. Hydra comes as a plug-in board and costs £ 125 + VAT for each PET in the network. Details on Derby (0332) 760127.

Small Systems Engineering have now added to their CP/M adapter a UNIX adapter. The CP/M unit uses a Z80 processor, whereas the UNIX adapter uses the 8088. The adapter plugs into the PET and provides 64K of memory while using the PET's existing 32K as a data buffer. Price about £ 1,000. Address: 2 - 4, Canfield Place, London, NW6 3BT. Tel: 01-328 7145/6.

Urwick Dynamics is a name to watch out for, they have developed a BASIC code generator for the PET (The next one ?). Unlike 'the other one', publicity has been minimal, but it was launched at the Pergamon Infotech State-of-the-Art review in London recently.

Qwerty Computer Services have a range of useful knick-knacks, covering light-pens, joy-sticks, sound generator (programmable), disk power-break protector, TV/video interface, character generator ROMs - software selectable, and a video RAM duplicator board. Details from 20, Worcester Road, Newton Hall, Durham. Tel: (0385) 67045.

Old-ROM PET users will have noticed how difficult/expensive it is to get replacement 6550 RAM chips. Now you can replace up to eight 6550 RAMs with low-cost 2114s one at a time. Two units are required for total memory replacement. Contact Optimized Data Systems, P.O.Box 595, Placentia, CA 92670, USA.

Not enough room in your 8096 ? UPSYS enables you to upgrade your machine to 256K in units of 64K. UPSYS is an intelligent RAM expansion organised in banks of 32K with an operating system installed at $E900 to $EFFF. Available for both BASIC 2 and BASIC 4 from Contract Trade Ltd., 788-790, Finchley Road, London, NW11 7UR.

The 220M mini digital recorder provides a compromise cost storage medium between the operation of disks and the cheapness of cassettes. Up to 64K per side can be stored on a system with software selectable options. In addition to read, write and verify are back-space, search, write-protect file, load/save directory. Note these recorders are PET-compatible and can be obtained by bona-fide IPUG members at £ 150 each if ordered in multiples of five. Contact David Annals for discount arrangements (p142 November), or Currah Computer Components Ltd., Graythorp Industrial Estate, Hartlepool, Cleveland, TS25 2DF. Tel: (0429) 72996.

Ever wanted the time and date to appear on your listings like they do on mini's and mainframes ? All sorts of applications come to mind for the Microscience HL811 real time clock for 3032, 4032 & 8032. The clock stores hours, minutes, seconds, day, month and leap-year information and attaches to the user port and rear cassette port without affecting cassette operation. Contact Microscience Ltd., P. O. Box 14, Bramhall, Stockport, Cheshire, SK7 2QS. Price £ 79.00 incl. V.A.T. & post.

--oOo--

DISCOUNTS

Firms offering discounts were published in the September issue (p130) and all offers therein still stand. We have, however, a new addition in Clearsons Ltd., who specialise in word-processor and computer supplies. A blanket discount to IPUG members of 15% is offered on stock items which include listing paper, printer ribbons (incl. NEC & Qume), disk mailing packs, etc. The exception is floppy disks where the 100+ price for Verbatim diskettes applies, e.g £ 1.40 each (VAT extra) in boxes of 10. See advert back page for further information.

Before actually purchasing any large item of hardware such as new computers, disk drives, etc, it is in your own interests to contact David Annal, IPUG's discount organiser. Certain firms have agreed substantial discounts but, owing to publishing lead times and varying stock levels, we cannot list them fully. As an example, VICs will be available with at least 15 - 17.5% off.

David can be reached on 01-764 4043.

--oOo--

TWO ROUTINES

By Geoff Lawrence

COMPUTED 'GOSUB'

GOSUB is a routine to perform a computed GOSUB. Enter with SYS(826)<expression>.

BASIC 4 version - hex dump

```
033A  A9 03 20 93 B3 A5 78 48
0342  A5 77 48 A5 37 48 A5 36
034A  48 A9 8D 48 20 76 00 20
0352  98 BD 20 EA C2 A5 62 85
035A  11 A5 61 85 12 20 33 B8
0362  4C 4A B7 60
```

BASIC 4 version - disassembly

```
ADDRESS      DEC HEX  OP

826 033A     169 A9 :LDA  #$03
828 033C      32 20 :JSR   $B393
831 033F     165 A5 :LDA   $78
833 0341      72 48 :PHA
834 0342     165 A5 :LDA   $77
836 0344      72 48 :PHA
837 0345     165 A5 :LDA   $37
839 0347      72 48 :PHA
840 0348     165 A5 :LDA   $36
842 034A      72 48 :PHA
843 034B     169 A9 :LDA  #$8D
845 034D      72 48 :PHA
846 034E      32 20 :JSR   $0076
849 0351      32 20 :JSR   $BD98
852 0354      32 20 :JSR   $C2EA
855 0357     165 A5 :LDA   $62
857 0359     133 85 :STA   $11
859 035B     165 A5 :LDA   $61
861 035D     133 85 :STA   $12
863 035F      32 20 :JSR   $B833
866 0362      76 4C :JMP   $B74A
869 0365      96 60 :RTS
```

Old ROM - hex dump

```
0338A  A9 03 20 1D C3 A5 CA 48
0342   A5 C9 48 A5 89 48 A5 88
034A   48 A9 8D 48 20 C8 00 20
0352   B8 CC 20 A7 D0 A5 B4 85
035A   08 A5 B3 85 09 20 A0 C7
0362   4C B5 6C 60
```

Old ROM version - disassembly

```
ADDRESS       DEC HEX  OP

826 033A      169 A9 :LDA  #$03
828 033C       32 20 :JSR   $C31D
831 033F      165 A5 :LDA   $CA
833 0341       72 48 :PHA
834 0342      165 A5 :LDA   $C9
836 0344       72 48 :PHA
837 0345      165 A5 :LDA   $89
839 0347       72 48 :PHA
840 0348      165 A5 :LDA   $88
842 034A       72 48 :PHA
843 034B      169 A9 :LDA  #$8D
845 034D       72 48 :PHA
846 034E       32 20 :JSR   $00C8
849 0351       32 20 :JSR   $CCB8
852 0354       32 20 :JSR   $D0A7
855 0357      165 A5 :LDA   $B4
857 0359      133 85 :STA   $08
859 035B      165 A5 :LDA   $B3
861 035D      133 85 :STA   $09
863 035F       32 20 :JSR   $C7A0
866 0362       76 4C :JMP   $C6B5
869 0365       96 60 :RTS
```

LINE HEADER

LINE HEADER is a routine to identify the address of the line header for a line whose number is at $033B/033F (LO/HI). Old-ROM and BASIC 4 versions are given.

BASIC 4 version - hex dump

```
033A   A9 0A 85 11 A9 00 85 12
0342   20 A3 B5 90 09 A5 5D A6
034A   5C A0 00 20 83 CF 60
```

BASIC 4 version - disassembly

```
ADDRESS     DEC HEX  OP
826 033A    169 A9 :LDA  #$0A
828 033C    133 85 :STA   $11
830 033E    169 A9 :LDA  #$00
832 0340    133 85 :STA   $12
834 0342     32 20 :JSR   $B5A3
837 0345    144 90 :BCC   $09 (0350)
839 0347    165 A5 :LDA   $5D
841 0349    166 A6 :LDX   $5C
843 034B    160 A0 :LDY  #$00
845 034D     32 20 :JSR   $CF83
848 0350     96 60 :RTS
```

Old ROM version - hex dump

```
033A  A9 0A 85 08 A9 00 85 09
0342  20 22 C5 90 FE A5 AF A6
034A  AE A0 00 20 9F DC 60
```

Old ROM version - disassembly

```
ADDRESS     DEC HEX  OP
826 033A    169 A9 :LDA  #$0A
828 033C    133 85 :STA   $08
830 033E    169 A9 :LDA  #$00
832 0340    133 85 :STA   $09
834 0342     32 20 :JSR   $C522
837 0345    144 90 :BCC   $FE (0345)
839 0347    165 A5 :LDA   $AF
841 0349    166 A6 :LDX   $AE
843 034B    160 A0 :LDY  #$00
845 034D     32 20 :JSR   $DC9F
848 0350     96 60 :RTS
```

--oOo--

DEBUG

Mike Todd apologises for a small error in his article on screen output routines (p156). References were made to the old-ROM routine entry point $E3EA instead of the BASIC2 entry of $E3D8.

--oOo--

SOFTWARE PROTECTION & ITS IMPLICATIONS

By Mike Todd

Software protection is probably one of the most contentious issues in personal computing at the present time. and there is now a wide range of techniques to protect software against software thieves and pirates.

When we talk about protection, it usually means anti-copying devices to prevent the customer making copies of the software. It usually applies to disk software but techniques exist to prevent copying of cassettes. Unfortunately (for whom ?) the Commodore range of computers and disk drives do not lend themselves to reliable forms of protection.

Before continuing it is worth pointing out that there is no foolproof method to prevent people taking copies of disks - it can be made very difficult but, as long as a disk is readable (and of course it has to be to get the software into the computer !) then it can, and will, be copied.

One of the early forms of protection was incorporated into Microchess. This had a small chunk of machine code in the 2nd cassette buffer and was saved using the machine language monitor. The SAVE command does not save this section of RAM, and so copies made in this way would not run. Of course, most machine code enthusiasts soon caught on and copies of programs thus protected were only safe from the ignorant.

With the arrival of Commodore disk units, protection became much more ingenious. To prevent copying using the inbuilt DUPLICATE command is easy - unused sectors on the disk are corrupted (at its most crude, this can be done with a small magnet !) so that the DUPLICATE command will run into READ errors and abort. This is can be circumvented by using the direct access commands and copying sector by sector (using a technique similar to that described in the September 1981 Newsletter - page 121). This also allows copying of disks protected by special disk software -

provided that they don't rely on information written outside the normal disk format, or rely on the fact that some sectors are missing.

No matter how complex the protection, the disk must be capable of being read at some time and if it is readable, then it is copyable ! Many enthusiasts (who have the knowledge, patience and the time) see cracking of protection as a challenge in the same way as a crossword enthusiast sees the Times crossword. It could be argued that these people are unlikely to buy the software in the first place, and are thus not depriving publishers of any significant revenue - while businesses (for whom such software is generally written) would find it cheaper to buy another copy rather than spend time and manpower in copying the software.

Resigned to the fact that the disks can be copied, some publishers resort to password protection. Unfortunately. it is impossible to stop the password being handed on by word of mouth - unless of course the password is secretly written on the disk, built into a security ROM or encoded into hardware (the so-called "Dongle"). All three techniques have been used, although the Dongle appears to be becoming the most popular. It is not impossible to copy a ROM or a Dongle - just a lot more difficult. Putting the password on disk, such that it is hidden (and cannot therefore be copied) is difficult but not impossible - the disk format can be amended to allow this password to be fitted in where the normal operating system cannot get at it. Even so, once software is loaded can't it then be saved using the monitor ? Well, if the program runs itself as soon as it is loaded, disables the STOP key BEFORE it loads the main program and corrupts interrupt vectors, pointers etc. then there is a fair chance that the user can't actually get at the program in RAM - unless he uses the crash recovery technique of resetting with the diagnostic pin grounded. This puts him into the monitor from where all vectors etc. can be restored and the software "got at". Clever publishers will, of course. make sure that the software is incomplete, and that the entire program never resides in RAM at once.

Even these techniques are not foolproof since it is possible (eventually!) to look at the software and see what the programmer is trying to do, and if necessary patch out the protection routines. If they are written obscurely (making extensive use of self modifying code, irrelevant blocks of code etc.) then life could be difficult, but not impossible.

The most successful protection uses a combination of all techniques so that, when one level of protection is cracked the next is revealed. This makes copying a bit like a safe-cracker who succedes in cracking a safe only to find another (different) safe inside. If they are nested several deep then he will probably give up before getting to the end.

Unfortunately. there is one occasion when copying software is a legitimate exercise. All good businesses keep back-up copies of their disks; this is as important for the software as it is for data disks. There are a few software publishers who provide a back-up copy free of charge and who will replace damaged disks for a minimal cost. There are those. however, who do not (and will not) provide back-up copies, only offering a replacement service. For most business users this is useless - no business can afford to wait in the middle of using a program for a replacement disk to be provided. My own opinion is that software houses who do not provide at least one back-up copy (and I would argue that there should be two) should be avoided at all costs. They may be offering reasonable software. but their customer understanding is rock-bottom ! There is even one software publisher who provides a replacement copy service only at a cost of 25% of the original purchase price.

Software protection can be very expensive but is it worth it considering the futility of it. A basic level of protection may be useful provided that adequate back-ups are provide as an intrinsic part of the package, but will not prevent unauthorised copying. Of course. companies are not really worried about the odd copy gained in this way, they are more concerned about the hacker who procedes to give copies away to all and sundry. and even sell copies illegally !

Probably the most effective protection is the same that has been used by the main-frame software houses for years - issue software on contract. Into the contract is written permission to copy for use on one machine only and that any unauthorised copying will be considered a breach of contract - a more reliable and easier way to gain legal redress. To do this requires a means to identify copies (a fairly easy thing to do) allowing them to be traced back to source.

--oOo--

A 4000/8000-SERIES INVESTIGATION

By Mike Todd

Recently I've been digging around the circuit diagram for the 8032 and the ROMs of the new 40-column PET (the FAT-40). The following summarises what I've found - but I stress that I own neither machine so can't double check anything.

The 8032 has some rather intruiging design features, some associated with the CRT controller chip (the 6845) and some with the rest of the hardware. 1) There is a NO ROM line to enable the BASIC ROMs (via their chip select pin) and it is also connected to pin 20 of the memory expansion port. One odd thing is that it also goes to pin 5 of the 6502 which in all my references is shown as not connected ! Is this to disable the ROM set if the 8032 is expanded ?

2) Pin 11 of the user port used to be grounded - it is now connected to the graphic select line (CA2 of the 6522).

3) The CB2 line of the 6522 (which goes to the user port and is used to generate sound effects for SPACE INVADERS, etc) is connected to an internal loudspeaker - through which you hear the internal "chime" facility. This line is gated by PA7 (diagnostic sense line) from one of the 6520 chips. Grounding the diagnostic line disables the loudspeaker. This could also be done by reconfiguring the port at $E810 with bit 7 as output in which case a "0" in bit 7 would disable and "1" enable the loudspeaker.

4) The BASIC4 ROMs in the 8032 are identical in every respect to those in all 4000 series machines except for the $E000 ROM which contains the screen/keyboard handling routines.

When I first examined a disassembly of the 8000 series "E" ROM I was extremely puzzled to see the 6845 CRT controller being set up for 40-columns and not 80!! I shall try to explain why this is so. Each character is made up of 8 rows of 8 dots. These are generated by the character generator ROM which gets its information for which character is to be displayed from the video RAM. On 40-column machines there are 40 x 8 = 320 dots horizontally and these are clocked on the screen at a frequency of 8MHz. The character to be displayed is selected by the CRT chip at a rate of 40 characters per row. Increasing the width to 80 characters is done by increasing dot frequency to 16MHz and selecting two characters for each character space on the screen. This is done by having two banks of RAM and flipping between them to provide alternately even/odd numbered characters within each character space.

The 6845 also has a light pen input available on pin 21 of the memory expansion port. Unfortunately this requires some additional circuitry to synchronise the light pen signal with the clock pulses.

The counter which selects which row of each character is being displayed normally counts from 0-7 (ie 8 rows per character). This would mean that all characters were contiguous on the screen. To provide vertical spacing in text mode, characters are redefined to 10 rows with the last two being blank. This is done by sending the appropriate control sequence to the 6845 chip and is done whenever you use CHR$(14) or CHR$(142). If you use the normal POKE59468,14 or 12 command you switch the character generator without affecting the 6845.

It is interesting to note that the circuit diagram shows the possibility of having a 2316 or a 2332 (ie 2k or 4k) character generator. In addition, there is a CHR OPTION signal derived from the 6845 (in rather an odd way) which should provide the option of selecting an additional character set. I suspect that this is provided for the 9000 series (the micro-main-frame) to allow access to the APL character set.

This signal is derived from one of the 6845 address lines, another of which is designated as an INVERT signal. This appears to be totally different from the normal character invert signal (obtained from bit 7 of the character code in RAM). If set, the entire screen will be inverted.

I don't have access to a FAT-40 or 8000 machine, but would imagine the following POKEs would access these facilities and could be tried by anyone with the appropriate machine:

```
POKE59520,12:POKE59521,X
```

where X=16 to invert screen, X=32 for alternate character set - however there is scope for experimentation.

Turning to the 12" 4000-series machines (nicknamed the FAT-40), these are hybrids, halfway between the 3000 and 8000 series.

The hardware is similar to the 8000, with the screen controlled by a 6845 (but with only half the RAM), and most of the software enhancements of the 8000 are available (eg TAB, BELL. clear to end of line etc). However, you can't set text windows and you don't have separate TAB/ESC/RPT keys or line insert or delete functions or input/output vectoring.

The only difference in ROMs is the "E" ROM as mentioned earlier - this is a KERNEL ROM as on the 8000 and has a table of vectors at the start. At the end of the article is a list of some of these routines.

Use is made of the 2nd cassette buffer (rendering it virtually useless). This is done because extra variables are needed and can't be stored in zero page (where the 8000 stores them) since the space is occupied by the screen line wrap-around table.

Many games programs access locations $97 and $A6 (151 and 160) which hold the key value of the key currently pressed. Old machines had a value of 0-80 which had to be converted to ASCII by looking up in a table. The 8000 and FAT-40 both provide ASCII values here. This can cause problems with some programs which will need minor modifications.

Some software which requires access to screen routines may require changes to addresses and the following is a list of some of the more important routines. Note that those in brackets are KERNEL addresses in a vector table. Note that BASIC4.40 indicates original BASIC4. 4.41 for FAT-40. and 4.80 for 8000-series.

B2	B4.40	BASIC4.41	BASIC4.80	
E1DE	E000	E036(E000)	E04B(E000)	Initialise the screen
E285	E0A7	E0A7(E003)	EA07(E003)	Get char from KB buffer
E29D	E0BF	E116(E006)	E116(E006)	Input from screen
E3D8	E202	E202(E009)	E202(E009)	Output char to screen
E61B	E442	E442(E00C)	E442(E00C)	Main interrupt entry
E62E	E455	E455(E00F)	E455(E00F)	Interrupt handler
E6E4	E600	E600(E012)	E600(E012)	Return from interrupt
E229	E04B	E042(E015)	E051(E015)	Clear screen
----	----	E60F(E018)	E07A(E018)	Set TEXT mode
----	----	E617(E01B)	E082(E01B)	Set GRAPHICS mode
----	----	E61D(E01E)	E088(E01E)	Set up 6845 chip
E5BA	E3E2	E6EA(E021)	E3C8(E021)	INS & scroll screen down
E53F	E369	E3D1(E024)	E3E8(E024)	Scroll screen up
E64D	E474-a	E4BF(E027)	E4BE(E027)	Handle keypress
----	----	E657(E02A)	E6A7(E02A)	Chime
----	----	-b- (E02D)	E3E8(E02D)	Set repeat flag
----	----	-b- (E030)	E1E1(E030)	Set top/left window
----	----	-b- (E033)	E1DC(E033)	Set bottom/right window
E257	E079	E06B	E05F	Home cursor
E25D	E07F	E071	E067	Set up line parameters
E748	E65B	E798	E755	Screen line addresses
E6F8	E60B	E73F	E6D1	Keyboard decode table

-a- keypress is not handled as a separate routine in BASIC2 or BASIC4.40
-b- facilities not available in BASIC4.41. performs RTS.

--o0o--

DISK MATTERS

By Mike Todd

Disk users may be interested in the following snippets. They are true for DOS2.1 although may work on DOS2.5 - PRINT#15 in the examples following assumes an OPEN15,8,15.

1) The disk unit performs a series of diagnostic routines when reset and indicates any error by flashing the LEDs on the front. The following is a list of the error indications (the number is the number of times the LEDs flash and FDC is the floppy disk controller processor):

1 -	RAM failure $00-$FF	6 -	RAM failure $1000-$13FF
2 -	ROM failure $F000-$FFFF	7 -	RAM failure $2000-$23FF
3 -	ROM failure $E000-$EFFF	8 -	RAM failure $3000-$33FF
4 -	ROM failure $D000-$DFFF	9 -	RAM failure $4000-$43FF
5 -	FDC not responding	10 -	FDC format unrecognised

It is possible to see the effect of error 5 using the following - the first line forces the FDC into a crash and the second forces a reset.

```
     PRINT#15,"M-W";CHR$(3);CHR$(16);CHR$(1);CHR$(208)
     PRINT#15,"U:"
```

2) The rate which the head steps in and out is determined by the value of $1000. It is initially 15 but can be changed:

```
     PRINT#15,"M-W";CHR$(0);CHR$(16);CHR$(1);CHR$(X)
```

Where X is the new value in the range 8-22. a value of 8 makes the head move very quickly !

3) When the disk unit encounters an error on the disk, it will retry several times (normally 10), and will then re-initialise the drive and have another go. The number of retries is determined by $435C and can be changed:

```
     PRINT#15,"M-W";CHR$(92);CHR$(67);CHR$(1);CHR$(Z)
```

Where Z is the number of retries - if Z > 127 (ie bit7=1) then the auto-re-initialise is disabled.

--oOo--

STRICTLY FOR BEGINNERS

Now that you understand variables and string variables, let us consider another way in which PET stores information. Arrays are one or more columns of data, arranged in one or more rows. PET can handle arrays of up to 11 elements (single pieces of information) without any special programming instructions. For more than 11 items, you must tell PET what to expect, so that it will reserve enough space for the desired function, in memory. This is done with the command DIMENSION, usually shortened to DIM.

At this point I think I should explain the word SYNTAX, because I should have covered it earlier, when you first started getting those annoying words on the screen which also stopped your program. Yes, I mean 'SYNTAX ERROR'. Syntax is the way in which you MUST tell PET what to do, via the keyboard. Not only the words, but also the punctuation, must be precisely correct for PET to understand.

PET has been pre-programmed (in Read Only Memory) by Commodore to read what you type. But it can only read COMMODORE BASIC. This is slightly different from other BASIC languages, although all are basically (forgive me) the same. This is why, if you type a comma where PET expects a colon, you will receive a SYNTAX ERROR message on the screen. PET is telling you that you have entered something which PET does not understand.

Back to business. The syntax we are presently interested in is as follows:

```
100 DIM A(20),A$(20):FOR I = 0 TO 19:INPUT A(I):
    INPUT A$(I) : IF A(I) = 0 THEN 120
110 NEXT I
120 FOR I = 0 TO 19:?A(I),A$(I):NEXT I
130 INPUT "WHICH NUMBER DO YOU WANT";N
140 FOR I = 0 TO 19:IF A(I)=N THEN ? "THE NUMBER YOU
    REQUIRE IS ";A(I);"AND THE NAME IS ";A$(I)
150 NEXT I
```

Notice that this program features multi-statement lines, to save space on screen. It creates two arrays. one numerical and one alphanumerical (i.e. accepts words or numbers or both). Then it allows you to select any parts of the data that you wish. One quite important point here - don't forget that PET has a memory better than an elephant. PET NEVER forgets. So until you either switch off, or alter the contents of these arrays, deliberately or accidentally, their content will remain exactly as it is now. This is what the program actually does:
100 DIM etc., causes PET to allocate space in memory. and labels the spaces A(0). A(1). to A(19), also A$(0), to A$(19).

The colon ':' means that is the end of that statement, so that PET knows it has finished that command. and now goes on to the next command.
FORI=0TO19:INPUTA(I);:INPUTA$(I):IFA(I)=0THEN120 means : - starting with I equal to 0, wait for the user to type in a number (INPUT) and place that number in position A(0). The semicolon means stay on the same line of the screen. The colon separates the next command from this one. Now wait for the user to type in a name, and place this name in space A$(0). Another colon to signify the end of that statement. Then look at the number in A(0), and if it is a 0, go straight to line 120, ignoring all lines in between. This is one way to halt the program whenever you want it to. and go on to do something else. We will consider line 120 when we come to it.
110 NEXTI means go back to FOR I and make I equal to I + 1. in this case I will now equal 1 (0+1=1). So the next number you type in will be in space A(1), and the next name will be in A$(1), and so on until I = 19. Then after executing A$(19). PET will go back to FOR and find that it has now reached the total you asked for, (19), so it will then go on to the next line of the program.
120 FORI=0TO19:?A(I),A$(I):NEXT I. This is fairly straightforward. It asks PET to start at A(0) and A$(0) and print them on the screen. The comma between A(I) and A$(I) causes PET to leave a gap between them on the screen of 10 spaces. (Unless the number is longer than 10 digits. in which case the gap will be 20 spaces). This proceeds as

with the INPUTs until A$(19) is reached and printed on screen. (N.B. If you used the '0' option in line 100, then all the A(I)'s and A$(I)'s after that will be set to zero and blank after the point when you keyed in A(I) as 0. To prevent this you could enter another IF A(I) = 0 THEN 130 on line 120 after the A$(I) and the NEXTI must be put at say, line 125, or it would not be executed.
130 INPUT"WHICH NUMBER DO YOU WANT";N. This line asks you to type in a number. You will notice that INPUTs always cause PET to print a question mark on the screen. This is to remind you that PET is waiting for you to type something in. Try typing in a letter and see what happens. Yes, a different error message, which does not stop the program. REDO FROM START. This means you have typed in alphanumeric data when PET wants numeric. You merely have to re-type a number and the program will proceed.
140 FORI=0TO19 etc. By now you should know what a FOR-NEXT loop is. It merely increments a variable by 1 (or more if told to do so), and works on the information given afterwards. In this case, we ask PET to check whether the number we have just typed in is the same as one in the array or not. If it is, then we ask PET to print a message on the screen. If it is not equal to the number in the array, then PET goes on to the NEXT in the array.

When PET reaches I = 20 in the last line of the program, it goes on to the next program line. As there is no next line, the program automatically ends, and READY appears on screen, indicating that PET is ready and willing for the next job. But again, don't forget that all the names and numbers you typed in are still present in PET's memory. To prove it, first of all, type LIST and RETURN. You will see the program listing appear on screen. Then type RUN130 and RETURN and the program will ask you for which number you require. Then after typing in a number, provided it was one of the numbers you typed in first of all, you will get the appropriate message on screen. If it is not a number that you have already typed in, the program will just end, as soon as PET has checked all 20 (0 to 19) numbers in its array in memory.

Note that if you now wish to type in another program, you MUST first of all type NEW to erase the present program from PET's memory. But if you wish to keep the previous program, you merely SAVE it on your cassette tape.

So. now we have covered multi-statement lines in programs, and created arrays. Also we have selected a particular part of an array at will. and printed it neatly on the screen.

POINTS TO REMEMBER:
PET counts from 0, unless you tell it otherwise.
';' means stay on same line of the screen
':' means end of this statement, go to next statement
FOR must be followed by NEXT somewhere in the program, and it is good practice for beginners to always specify the variable, e.g. NEXT I
IF must always be followed by THEN somewhere in the program

Finally, please do not hesitate to contact me if you are having the slightest difficulty, or to let me know if there is any aspect of PET BASIC you would like to see in this column.

Ray Davies. 105, Normanton Road, Derby, DE1 2GG.

--oOo--

REVIEW

Faster Basic Supersoft £30.00 + V.A.T.

This is another chip from the prolific Supersoft. It was written by D.J.Mundy and is available for any 2K ROM area of a BASIC 2 or 4 PET. Faster BASIC can be paired with most of the other Supersoft 2K chips to form a 4K chip, thus utilising ROM space to the full (I have it paired with Arrow - a pretty powerful combination !).

As its name suggests, Faster BASIC attempts to make your programs run more quickly. It does this in three ways, by speeding up:
(1) Interpretation of integer constants,
(2) Interpretation and location in memory of variables,
(3) GOTOs GOSUBs and THEN followed by a line number.
The increase in speed is obtained by modifying each program statement the first time it is executed. These modifications are totally transparent to the user though,

because they are 'undone' when a LIST or SAVE command is given. There is a noticeable delay of about a second when listing a long program immediately after a run, while F.B. reinstates the changes.

What sort of speed increases can be achieved ? Well this obviously varies considerably but I would say the average increase is 25%, e.g. 45 secs. as against 60 secs. If a program has been written using all the well known speed increasing techniques such as defining the most commonly used variables first, using variables instead of constants and placing frequently used subroutines at the head of the program then Faster BASIC is not likely to have a pronounced effect.

In my opinion, it's main advantage is that programs can be written without regard to the above principles with impunity. With Faster BASIC it is possible to have programs that run fast AND are well structured and easy to understand.

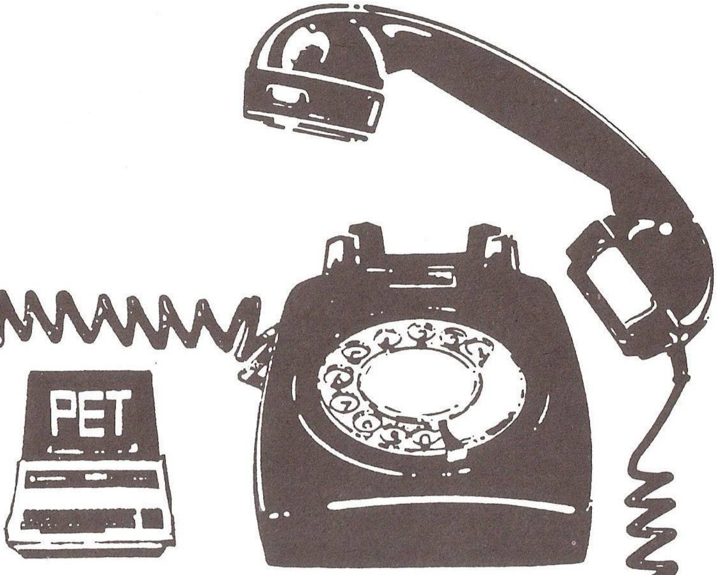
Here are some timings (in jiffies) for single statements, each executed 2000 times: POKE32768,4 (349 with F.B., 1092 without), A=12345678 (127,698), A(1)=1 (496,665). F.B. is particularly effective on long programs with many inter-twined GOTOs and GOSUBs. My timings for the PCW benchmarks are, with F.B. (1.7, 4.8, 12.8, 12.5, 12.8, 20.9, 37.5, 11.3) and without (1.9, 9.9, 18.1, 20.0, 21.5, 32.4, 50.0, 11.7). With F.B. the PET out performs the Apple, TRS80 and Sharp MZ80-K on these benchmarks !

Faster BASIC uses a few locations in page zero and 64-bytes in the string storage area, but it doesn't touch either cassette buffer. It is turned on and off with one SYS command and it overwrites part of CHARGET. It is compatible with ARROW but incompatible with PIC-CHIP and the TOOLKIT.

In summary I highly recommend this chip. It could be used by the businessman to speed up his BASIC Stock Control program (obviously it won't speed up a pure machine code program), by the hobbyist to produce faster running graphics games or by the mathematician who wishes to have a rapid, but well structured numerical integration routine.

Nick Higham, 90 Half Edge Lane,
Eccles, Manchester, M30 9BA

--oOo--



CLEARSONS

Computer Supplies

It had to come, Cash and Carry Centre

All your computer and word processing Supplies from one centre

Listing paper - all sizes available, Floppy Discs, Disc Packs,
Magnetic Tape, Continuous Labels, Printer Ribbons,
VDU Tables and Trolleys
Full range of Acco Computer Stationery filing Systems

High speed copying and collating service - able to reduce and enlarge up to A3

Clearsons Speedprint
all other types of printing

Call in or ring
Farnborough (0252) 518022 - 518717 - 41313

CLEARSONS LTD.
30, Camp Road, Farnborough, Hants.

Large Car Park at Rear

THIS SPACE

is available for advertising

To:

Membership Secretary,
IPUG,
30 Brancaster Road,
Newbury Park,
Ilford, Essex.

ANNUAL SUBSCRIPTION 1982

I enclose £7.50 (U.K.)/ £11.00 (Overseas) (delete as applicable) Annual Subscription to IPUG.

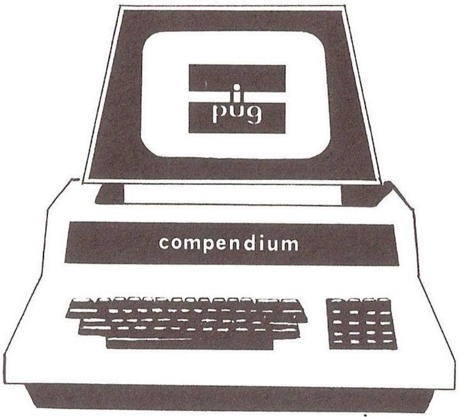
NAME (Block capitals) ....

ADDRESSS (Block capitals) ....

....

....

IPUG



COMPENDIUM

of the Independent **PET** *Users Group*

The Compendium is now available at the reduced price of £2.50 from the Membership Secretary. Please send your cheque or postal order, made payable to IPUG, to:

Jack Cohen, 30 Brancaster Road,
Newbury Park, Ilford, Essex IG2 7EP.

Printed, published and distributed by Carlton Press, 10 Bernard Road, Gorleston, Gt. Yarmouth, Norfolk NR31 6EG, England, UK. Tel: Gt. Yarmouth (STD 0493) 61331