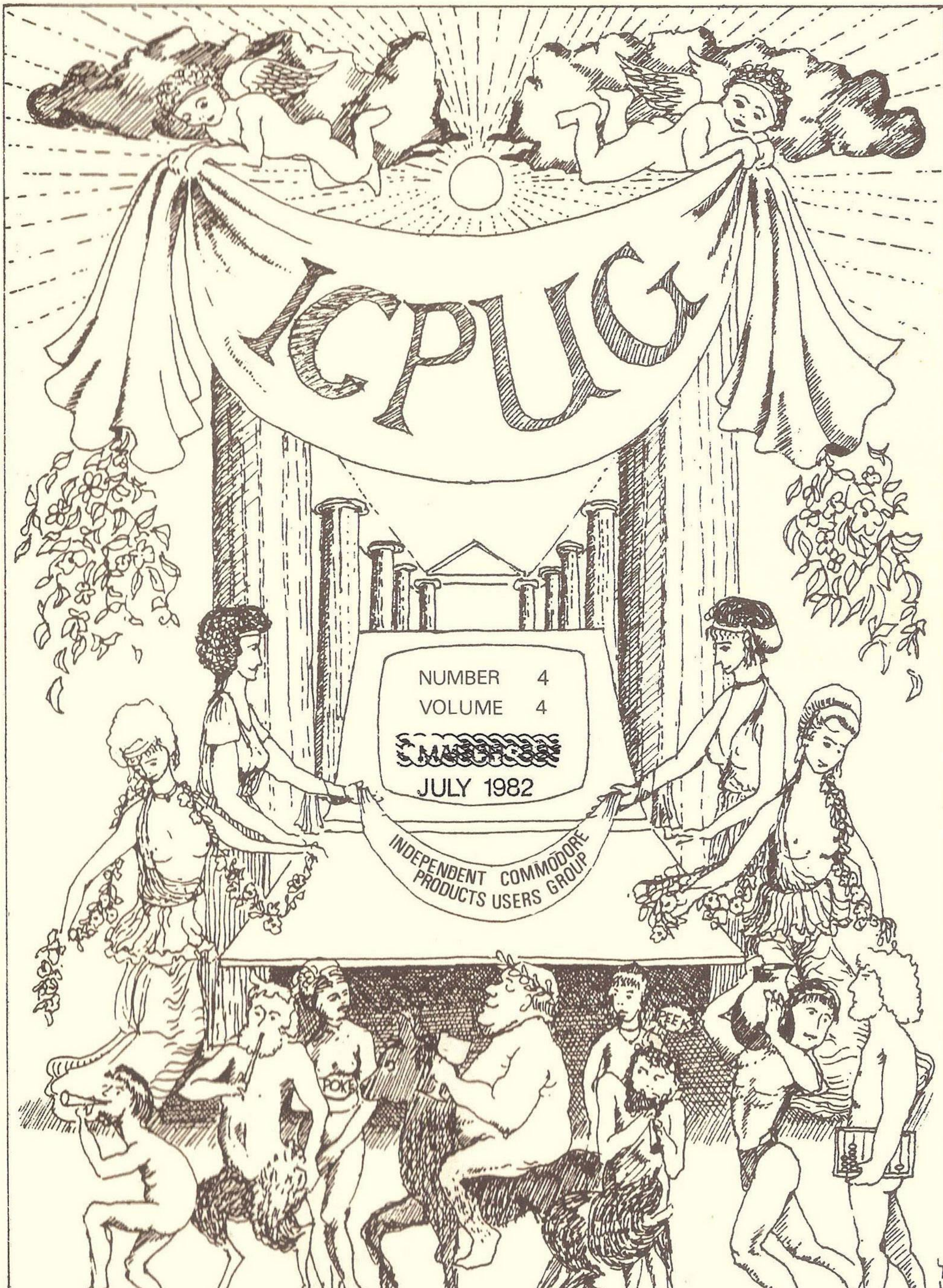

ICPUG
NUMBER 4
VOLUME 4
JULY 1982
INDEPENDENT COMMODORE PRODUCTS USERS GROUP
POKE
PEEK

HONORARY NATIONAL OFFICIALS

Chairman:	Wing Cdr. Mick Ryan 164 Chesterfield Drive, Riverhead, Sevenoaks, Kent TN13 2EH Telephone: Sevenoaks (0732) 453530
General Secretary:	Jim Tierney 11 Collison Place, Tenterden, Kent TN30 7BU Telephone: 058-06 2711
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: (0246) 811585 (work) (0226) 85084 (home)
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
Discounts Officer:	Dr. David Annal 142 Windermere Road, Norbury, London SW16 5HE Telephone: 01-764 4043

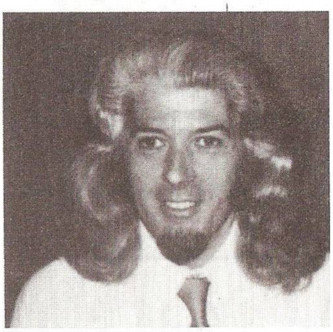
INDEPENDENT COMMODORE PRODUCTS USERS GROUP

Vol. 4 No. 4 **Newsletter** July 1982

Europe's first independent magazine for PET users

Page	Contents
170	Editor's Notebook
171	4022 Printer Modifications
172	Round the Regions
173	Words of Warning — Disk ID Changing
174	Printer Presence Check
176	4022 Secondary Addressing
178	Comal Corner
181	VIC Matters
188	VIC Memory Maps
199	Matters Arising
200	Shop Window
203	Microchess on Ice
204	Disk to Tape Save
211	Commodore Column
212	4032/8032 STOP Key Disable
214	Line 350 800?
216	Review — The PET Index

EDITOR'S NOTEBOOK



Newcomers to computing expect for some strange reason their computer to be accurate. Now within reason this is so, but it does depend upon a number of contraints. Calculations are normally performed to a specified number of significant figures, not in decimal, but binary. In the process of conversion from decimal to floating-point binary format their may well be a rounding error. Furthermore, trigonometric functions are usually evaluated by summing a polynomial series whose constants and number of terms are a compromise between accuracy in keeping with the rest of the machine, and speed of summation. Having obtained a result, the PRINT function usually outputs less significant figures than were used in the computation. Finally, the software itself may have bugs in it which produce assymetric results (e.g don't expect an expression like 4 x 2 to give the same result as 2 x 4). Such a bug occurs in the INT function. When programming one must allow for these restrictions, so don't write something like: FOR I = 1 TO 20 STEP 1.2: IF I = 4.8 THEN PRINT"I EXPECT 4.8" because chances are 'I' will not compute to 4.8 but 4.800000001 for example.

I thought it was a little cheeky, but during the Commodore Show in June, Apple had their first exhibition, not in the heart of London, but of all places at Slough.

There is a tremendous amount of information available in the User Group, more so than perhaps anywhere in the world, on Commodore equipment. One fundamental problem is the collation and dissemination of this vast resource. Most of the work seems to be by the same overloaded few, and as a result, the upgrade to the 'Compendium' with new material which was to form the 'Gazetteer', now seems unlikely to appear.

R.D.G.

--oOo--

4022 PRINTER MODIFICATIONS

By Colin Spencer

I have a late model 4022 printer with the red error LED in the form-feed switch and these mods refer specifically to this type of 4022 printer. I do not know if they will work on the earlier model without the LED. I am very pleased with the printer but I wanted other facilities, particularly the ability to have the printer work as a friction-feed type. I saw that Epson do a friction-feed conversion for their MX-70T and MX-80T printers and as the 4022 is based on these printers, I thought that this would work on the 4022. Chromasonics were kind enough to get this conversion on a sale or return basis for me. I fitted this to my 4022 without too many problems. The biggest problem was replacing the springs on the ruler with the much stronger ones supplied in the kit. This is a very fiddley job and is best done with two people. The ruler is the front mounting for the friction feed. The only disadvantage with this modification is that the only tractor-fed paper that now fits the machine is that with the tear-off tractor holes. This may create some problems for people wishing to print labels. With this set-up, friction and tractor are available at all times.

The second modification is to update the printer to the 4022P. The 'P' suffix means that the printer is bi-directional and this is accomplished by a new ROM. It is an 8K ROM, and so is not readily copied. In the Commodore spares manual, the ROM is listed as a part no. 901631-02 or 325301-01. This replaces the existing ROM on the board and is just a simple matter of just pushing it into the socket. In the spares manual it lists this ROM as being suitable only with printers fitted with an 8MHz crystal. My printer was so fitted and again Chromasonics loaned me a ROM out of a defective printer which I plugged in and sure enough, it worked. They therefore ordered one for me from Commodore, but at the time of writing, I am still waiting for it to turn up from Slough.

I hope these modifications are of interest and if you try them, good luck ! If you want to know more try me on 01-349-2345, or Chromasonics on 01-263-9493.

--oOo--

ROUND THE REGIONS

Watford

The Watford regional group have been very active recently in putting PRESTEL on the PET and VIC. Telèsoftware downloaders have been written and for the price of £ 150 for the hardware to interface your computer to the telephone line, your are in business, delivery permitting. Full details may by obtained from Tony Sweet at Prestel HQ on 01-983 5811.

Recent talks have been on the new hardware from Commodore by John Collins and one on business accounting, including Visicalc, by business consultant Barry Miles.

Slough

It's early days yet but the newest ICPUG group at the East end of Berkshire seems to be getting itself well established. It does by the way encompass both PET and VIC users, at present PET owners outnumber the VIC owners by 3 to 1, but new VIC members are appearing rapidly.

The survey for the initial meeting showed that almost everyone was interested in 'games' and either education, business applications or control systems (in that order of popularity) as their other interest(s).

At the first proper meeting there was much to see and do. Jack Cohen the national membership secretary was there and as well as signing up new members showed us Superscript and some Comal.

Brian Jones, the group's instigator, gave a talk on 'The value of truth' including the use of Boolean expressions in arithmetic operations, and Geoff Luxford, a co-organiser brought along his firm's copy of Silicon Office to show people. At the following meeting John Collins from Commodore gave a review of the new hardware and Derek Gerrish the group's software librarian, showed his Key Chip.

At the June meeting, the group was shown the video of the Commodore new releases prepared for the PET show and discussed various members' impressions of the show and the new hardware. In addition, a variety of 8032 games were on display, and 40-column PET owners as well as VIC owners were heard to mutter about the limitations of their screen size.

The July meeting is sheduled to have two of Slough College's students along, both currently working for Commodore, to demonstrate VIC programs (Simplicalc and VIC Writer) for which they have been preparing manuals. This is in addition to be a PETSPEED demo.

The group has still yet to find a regular meeting place or night, but the first meeting after the summer is likely to be Monday 20th September, contact Brian Jones, either at Slough College, Slough 34585 x81, or at home Reading 661494 for more details.

--oOo--

WORDS OF WARNING - Disk ID Changing

Those of you considering using Brian Grainger's program to change the Disk ID, or for that matter Jim Butterfield's earlier version, should heed the warning that goes with so doing. DOS2 versions automatically initialise a disk by checking and detecting a change in ID. It follows that two disks with the same ID will cause confusion if they are not identical in all other respects, datawise. This is because the drive's Block Allocate Map (BAM) will not be correctly updated with the result that data could be written to occupied blocks. The 'correct' ID is not the one that appears on the header (which is the one readily changed by software), but the one written to every sector on the disk when it is NEWed.

Haphazard changing of the ID is therefore not to be recommended and a program to restore the true ID appeared in Commodore Club News, August '81, by JB himself.

R.D.G.

--oOo--

PRINTER PRESENCE CHECK

By Ray Davidson

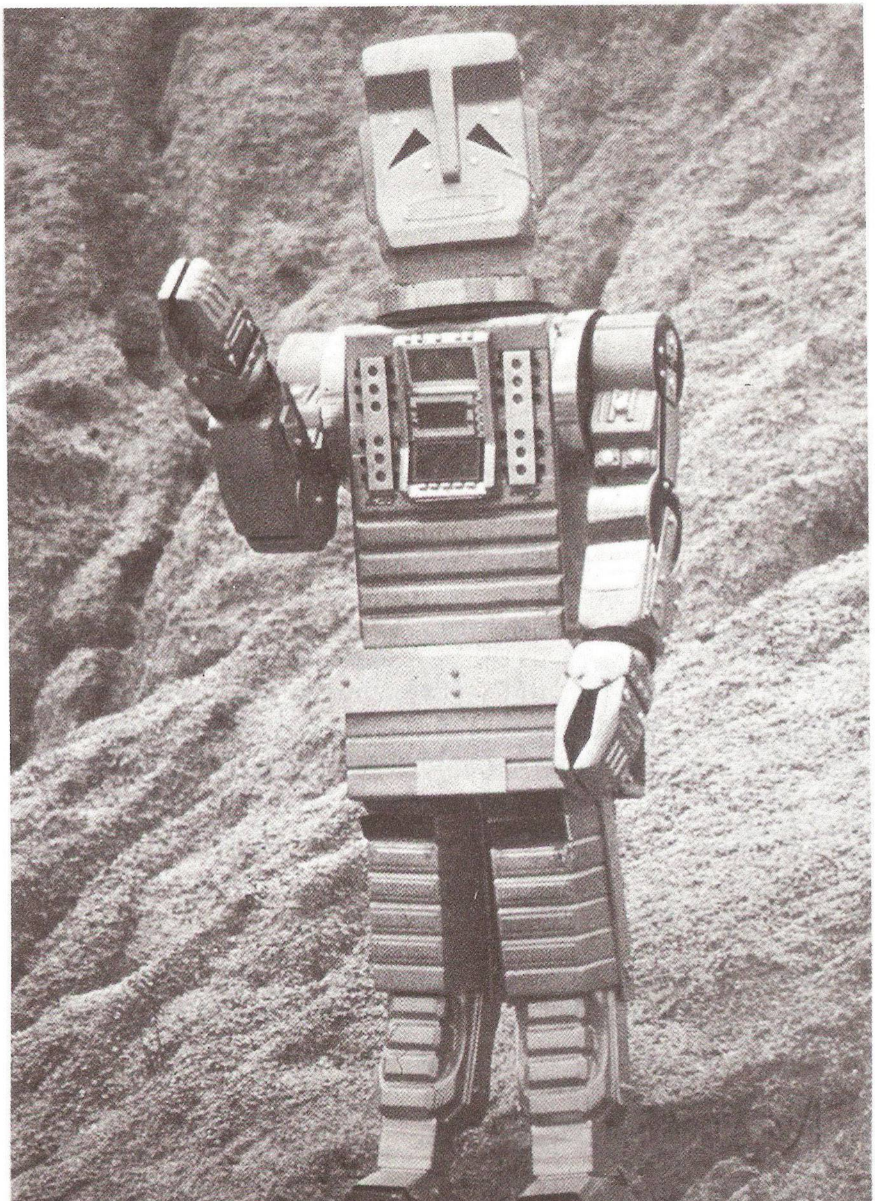
At the programmers' Seminar of the Commodore Show a question was asked about checking from BASIC to see if the printer is ready for use. In the following program line 9000 checks for the presence of the printer and line 9020 outputs the message if it is not ready. Line 9010 is a cop-out in case a fuse is blown or you've lost the connecting lead or something similar. Line 9030 programs a 'pound' sign for a Commodore printer. 9100 checks that the printer is on before making sure that the buffer is empty and then closing the file at 9120.

```
9000 CLOSE4:PR=0:POKE59456,PEEK(59456):P=PEEK(59456)AND1
     AND1
9010 GETG$:IFG$<>""THEN200
9020 IFPTHENPRINT"<rt><rvs> CONNECT THE  PRINTER AND
     SWITCH IT ON <2up>":GOTO9000
9030 A$=CHR$(13)+"?MA3"+CHR$(0):OPEN5,4,5:PRINT#5,A$:A$=""
     :CLOSE5
9040 OPEN4,4:PRINT#4:PR=1
9060 GOTO200
9100 IF(PEEK(59456)AND1)THEN9120
9110 IFPR=1THENPRINT#4
9120 PR=0:CLOSE4:GOTO200
```

This routine will only work with Computhink disks or when there is only one device on the IEEE-488 bus. A Commodore disk drive can produce unpredictable results if it is connected, even though it is switched off. If the check for the printer is made just after the printer is switched off, it will seem OK as it takes a few seconds for the smoothing capacitors in the printer to discharge, during which time it will respond to IEEE signals.

--oOo--

'ROBOTS-I HATE THEM'


Marvin, the manic-depressive robot, from the BBC's 'Hitch-hiker's guide to the galaxy'

4022 SECONDARY ADDRESSING

By H. Cohen.

In the article by Robin Harvey on the Commodore 4022 printer in the March issue (p85), he noted he was having difficulty with the number of lines per page facility. I agree with him ! The manual is not very helpful, but I have tested secondary address 3 on a 4022P printer and found how it works (with the help, I must admit, from David Pocock of CBM, Slough).

Secondary Address 3 - Set number of lines per page:

This is shown in program 1, where line 10 opens secondary address 3, directs 10 lines per page, and closes file #3.

The number of lines must be given as a CHR$ figure for this printer only (the instruction is different for 2022 and 2023 printers).

Line 12 opens the printer for paging and sends the paging flag CHR$(147), which must be followed by a semicolon to avoid a new line. Line 40 sends the paging off flag CHR$(19), again with a semi-colon. With paging in effect, the paging off flag performs a 'top-of-form' setting. The setting of SA #3 must occur before the page flag is set according to David Pocock (I'm not sure why!).

Set Paging

This is shown in program 2, which is similar to program 1 but with line 10 removed, variable J reset to 80 in line 15, and the print statement in line 20.

Note that the printer remembers SA #3 until a new line setting or the paging-off flag is sent, or the printer is switched off.

```
10 OPEN3,4,3:PRINT#3,CHR$(10):CLOSE3
12 OPEN1,4:PRINT#1,CHR$(147);
15 J=40
20 FORI=1TOJ:PRINT#1,"NUMBER OF LINES PER PAGE TEST FOR 10
                   LINES - LINE NO.",I
30 NEXTI
```

```
 40 PRINT#1,CHR$(19);
 50 CLOSE1:END
Program 2
 12 OPEN1,4:PRINT#1,CHR$(147);
 15 J=80
 20 FORI=1TOJ:PRINT#1,"NUMBER OF LINES PER PAGE TEST FOR 60
                                  LINES - LINE NO.",I
 30 NEXTI
 40 PRINT#1,CHR$(19);
 50 CLOSE1:END
```

--o0o--

ICPUG

Would officials of regional groups referring to our User Group as a whole, please use the term ICPUG National, since the term ICPUG Central could be confused with a regional group in the Midlands, or thereabouts.

--o0o--

WANTED

Cassette unit for VIC-20 - will pay £ 30 (+ £ 5 p&p if necessary) OR floppy disk drive - will pay £ 150 + p&p. Contact William McKnight, 65, Castle Road, Skelmorlie, Ayrshire, Strathclyde. Tel: (0475) 520164.

--o0o--

SPECIAL OFFER

Arrangements have been made for members to obtain a new publication from Holland giving a complete annotated disassembly for both old and upgrade ROM sets. The publication comprises 70 A4 pages in loose-leaf form. Should you wish to order one, send an international money order or cheque for US $20, payable to 'Copytronics', enclosing your name and address to Jack Cohen (see inside cover for address).

--o0o--

COMAL CORNER

By Brian Grainger

The purpose of this article is to pass on the latest knowledge I have on the versions of COMAL distributed by ICPUG (Version 0.11). I was very fortunate in manning a stand with Instrutek at the Commodore show. Instrutek wrote the original COMAL interpreter for BASIC4 PETs so I took the opportunity of clearing up some of my queries. The results are presented here.

1) In January, I said that programs SAVEd from BASIC4 could not be LOADed into BASIC2 COMAL. I have since found that this was incorrect and that why I could not do it was due to a hardware problem. COMAL does not SAVE the start and end program addresses like BASIC. It stores offset addresses and so SAVEd COMAL programs will LOAD into any version. An excellent idea. Pity Instrutek have spoilt it by making the latest versions of COMAL different in the command usage, but more of that later.

2) Please note that a command such as RUN 120 is NOT implemented in COMAL.

3) In January, I mentioned things that were automatically added by COMAL and did not need to be typed. I have since found that one need only type CHR and the '$' will be added automatically to make the CHR$ function.

4) I was not totally aware until recently of the flexibility of the string functions of COMAL. Values can be assigned to substrings in COMAL so making it very much easier than BASIC when we have a database situation where we have one string holding different items of information. An example should clarify. Suppose we have set up a database where each record is held in a string with 3 fields of information, Girlfriends name, Date last taken out and telephone number. e.g.

```
A$="MARY--5/6--458923"
```

If you subsequently took Mary out on the 10th July you would need to update your record. In BASIC this would be done by:-

```
            A$=LEFT$(A$,6)+"10/7-"+RIGHT$(A$,6)
```

In COMAL it can be done simply by:-

```
                      A$(7:5)="10/7-"
```

5) In January I gave the COMAL equivalent of PRINT# as WRITE FILE. This is not quite true. When WRITE FILE is used numbers are stored in floating point representation, not the ASCII representation. This is very useful for database applications because no matter what the number is, it will always use the same number of bytes for storage (5). The drawback is that one cannot address other non storage devices with WRITE FILE. The solution is the command PRINT FILE, which is identical in all respects to PRINT# of BASIC. There is also a command INPUT FILE to read the data that was written using PRINT FILE. With these two commands the full input/output capabilities of CBM BASIC exist in COMAL.

6) There are a number of keyword abbreviations available:-

';'	is equivalent to PRINT
'*'	is equivalent to OF in string DIM statements
'#'	is equivalent to FILE

N.B. To abbreviate PRINT FILE one can use PRINT # (note the space) or ';#'.

7) It is possible to LIST (or ENTER) a file as a sequential rather than a program file. To list the file FRED as a sequential file it is necessary to type LIST"FRED,S". As newer versions of COMAL expect the LISTed file to be sequential this is rather important in transferring programs from one version to another.

8) Now for the best news of all. Instrutek told me that PRINT USING, while not implemented in the full COMAL, WAS implemented in the split version. I've tried it and they are right.

In the above I have made reference to newer versions of COMAL. The current status is as follows. The versions for BASIC2, BASIC4 and 8032 machines distributed by ICPUG have until now been at version 0.11. The version for 8096 distributed by Commodore Educational workshops and myself (on 8050 disks) is at Version 1.01. This version allows over 30K of storage space for programs and data. The latest version for BASIC4 machines is version 0.12 but is not available as a 'split' version COMAL. The next formal copy for 8096 will be version 1.02 and I currently have version 1.02A (on 4040 disks!) which is almost, but not quite, version 1.02. The board version is currently equivalent to 1.02 and is available from INSTRUTEK, Christiansholmsgrad, DK-8700 Horsens, Denmark. This board fits any PET and leaves virtually all the RAM space for COMAL programs and data. The cost is about 200 pounds.

The 8096 COMAL versions are in general as the BASIC4 versions with additional commands. Unfortunately, the newer versions, e.g. 0.12, have some different command structures from the originals, e.g. 0.11, and programs are therefore not totally compatible. I hope in the next Newsletter to identify these differences. I might add that the new versions also have new useful commands while leaving more space available for programs so there are some positive benefits. Let us hope that future enhancements will be upward compatible.

--oOo--

THROWING SOME LIGHT ?

During the Commodore Show on the Friday afternoon, the top brass became conspicuous by their absence. Now I happen to know what was going on. Was it game time on the latest VIC, or perhaps a demonstration of 'Complicalc' ? No, someone (who shall be nameless) took along a Sinclair ZX Spectrum - nothing like weighing up the opposition !

--oOo--

VIC MATTERS

By Mike Todd

Firstly, I have to admit an error on p23 of the January Newsletter. The colour memory locations quoted are in fact back to front - with video RAM starting at $1000, colour starts at $9400 and if it is at $1E00, colour starts at $9600. Thanks to Pete Rowan of Whitehaven in Cumbria for pointing it out and my apologies for the error.

Arising out of this, it is often important to determine where the video and colour RAM actually starts. When the VIC is initialised, it makes a note of the address of the video RAM in $0288 (648 in decimal) and the actual start address can be determined by PEEK(648)*256. The start of colour RAM can be determined by ((PEEK(648) AND 3) OR 148) * 256.

This last calculation is based on the way the VIC itself converts a video RAM address into a colour RAM address. If you find that you would prefer to calculate the video RAM address and then convert it straight into an equivalent colour address then use DEFFNC(X)=(XAND1023)+37888 at the start of the program and use something like V=video RAM location, POKE V,character and POKE FNC(V),colour. Don't forget that the "X" used in the function definition is a "local" variable and does not change any current value of "X" already calculated in the program.

BUYER BEWARE

There are currently two major books on sale covering the inside workings of the VIC. There is VIC REVEALED by Nick Hampshire, and Commodore's own PROGRAMMERS' REFERENCE GUIDE.

The VIC REVEALED, as many who've bought it will know, is riddled with errors of fact, typing errors and just plain awful presentation. I cannot recommed the book to anyone and to those who have succumbed to the advertising I can only offer my sympathies. Fortunately, the book is to be republished in the next couple of months hopefully with the errors corrected. If the new edition is accurate and better presented then it will be a very useful book indeed, covering aspects of the VIC which are not publicised elsewhere - not even in my columns !

Of course this is no consolation to those who have already bought the book. So in the next couple of months I hope to produce an ICPUG errata for the first edition. This will be available to ICPUG members free of charge - it will also be made available to non-members at a price that has not yet been determined. If you would like a copy, please send a stamped (15.5p), addressed envelope (the same size as the Newsletter envelope) to me and I will send the errata as soon as it is finished. If you have any comments on the VIC REVEALED please feel free to enclose them - you may have found errors that I have missed. I would guess that the errata would be sent out some time at the end of July.

The other publication is the PROGRAMMERS' REFERENCE GUIDE published by Commodore. This too has errors (but not as many as VIC REVEALED) but is very much better presented and contains a great deal more material. This book is actually from Commodore in the USA and when Commodore UK saw it, they didn't like it. Demand for the book was quite strong and so Commodore did eventually release it - not for the original price of £14.95 but for £9.95, reflecting their concern over the accuracy of the contents.

Unfortunately, I hear that some retailers are selling the book at its original price of £14.95. I contacted Commodore who confirmed the price of £9.95 was correct and that the book has always been sent out for sale at this price. Therefore it follows that any dealer selling the book at the higher price is taking advantage of the situation and I would recommend that anyone who has bought the book at the higher price should take the book and the receipt back to the dealer and ask for the difference to be refunded. I must stress that there is no legal obligation to do so.

I would like to see some pressure brought to bear on this sort of unscrupulous dealing, especially as retail price maintenance, although generally illegal, is allowable on books and so Commodore could insist on a specific price to be charged for this book whereas they commit an offence if they try to do it on anything else.

DOWNGRADING YOUR VIC

I have had several letters from people trying to use programs that they've bought but which won't run on their VIC. Generally the problem is that some software is not written to take into account the fact that the screen memory may move, that the program could end up at the wrong address or that there may be cartridges such as the SUPER-EXPANDER installed.

The problems can be overcome by removing extra memory or taking out the SUPER-EXPANDER cartridge. They can also be overcome by downgrading the VIC using a couple of POKEs and SYS commands. These should always be done before loading the program since they effectively reset the VIC from scratch.

Firstly, disconnecting the SUPER-EXPANDER is often important, especially if you want to disable the STOP key. There are two stages involved. Firstly, the kernal side of the VIC needs to be disconnected and this is done by SYS64850. This could be done at the start of a program since it doesn't affect the BASIC program in any way, however it does disable the programmability of the function keys. The rest of the SUPER-EXPANDER can be removed by SYS58232. This SYS does upset the BASIC pointers and will have the same effect as NEW. This latter SYS is not required if any of the following downgrade options are used.

The following POKE/SYS sequence is required to downgrade the VIC:

```
POKE641,0:   POKE642,X:   POKE643,0:   POKE644,Y:   POKE648,Z:
SYS64824
```

X,Y and Z are determined from the following table:

DOWNGRADE TO MEMORY SETUP	642 X	644 Y	648 Z
unexpanded VIC	16	30	30
+3K only	4	30	30
+0K*	18	32	16
+8K	18	64	16
+16K	18	96	16
+24K	18	128	16

* NOTE: the +0K option is a condition that the VIC would never set up itself. It is effectively the same memory map as for a +8K machine, but with NO RAM expansion - all it does is move the screen to $1000 and start of BASIC to $1200.

Inevitably, these POKEs require a lot of typing - there's very little that can be done about that, although it should (at least in theory) be possible to prepare the POKEs on cassettes or disk and load them as required. Apart from allowing you to run programs that would otherwise crash, they also allow simulation of other machines just to confirm that programs developed on one set up will work on another. You can even do a "pseudo upgrade" by using the larger machine POKEs just to see what 24K bytes free looks like.

FINALLY (ALMOST)

A couple of other small points. Firstly, I still do not have details of the VIC software library that I'd hoped to publish this time round. Hopefully I'll have the details next time.

You will know that line numbers greater than 63999 produce a SYNTAX ERROR - so try entering a line number 350720 - the results are interesting and covered elsewhere in this Newsletter.

I had a recent opportunity to try out the VIC disk drive and was pleased with the compatability with my existing PET set up. Unfortunately, I was very surprised how slow the drive was compared with my 3040 disk drive. It would appear that the main reason is that everything in the VIC disk drive is handled by a single microprocessor so that the "look-ahead" features of the PET drives cannot be used. The result is that the drives operate about half the speed.

GARBAGE COLLECTION

I mentioned last time that I would explain the problem referred to as "garbage collection" which can cause the VIC to stop everything for anything up to a couple of hours in the extreme cases.

String variables are stored in a table, just like numeric variables, except that the actual string itself is not stored. Instead, the VIC stores the number of characters in the string and then a pointer to where in memory the string is actually stored. Storing strings in this way means that string manipulation is fairly easy to perform and strings do not have to have their length predetermined as on some small computers.

The combination of the length and pointer to the string is known as the string descriptor since it is describing the string. This of course means that the strings themselves need to be stored somewhere.

When you set a string variable in a program using something like A$="STRING", the characters "STRING" are already stored as part of the BASIC text and the descriptor actually points to those characters in the text. This means that no bytes are wasted storing a string twice. Even if you then use B$=A$, the descriptor for B$ points into the BASIC text.

However, as soon as you start to manipulate strings, the descriptor can no longer point into BASIC text. For instance, after A$="TEST", if you use B$=A$+A$, the resulting string "TESTTEST" must be stored somewhere else.

The BASIC program text starts from a given location and works upwards through memory. The variable tables start immediately following. It is not possible to store these strings immediately after the variable tables since the end of the tables varies during program execution as new variables are used. If the strings were stored here, they would need to be constantly shifted around as new variables are created and then the string descriptors would also have to be updated as their strings are moved.

Instead, these strings a built from the top of memory downwards and every time a new string needs to be stored it is added to the bottom of the existing strings. Eventually, these strings will move down so far that they are in danger of overwriting the variable tables which are moving upwards. As soon as the VIC detects that this will happen, it has two options. It can either give an OUT OF MEMORY error or it can see if any space can be recovered from the strings.

I have said that new strings are added to the bottom of existing strings - well, what happens in the situation of a simple program like:

```
10 A$="ABCDE"
20 FORI=1TO1000: B$=A$+A$: NEXT
```

In this case, only one string is actually evaluated (B$) but in practice, the VIC adds the resulting string to the bottom of the existing strings so that after only a couple of times round the FOR..NEXT loop, there will be several copies of B$ stored in memory and eventually this will result in the bottom of the strings "hitting" the top of variables.

Clearly, most of what is in the string space is "garbage" - that is unwanted strings that need to be got rid of. That is taken care of by the "garbage collection" routine.

This routine has a pointer which is initially set to the very top of the strings. It searches ALL the string descriptors for a descriptor nearest to its pointer. This must be the first used string in the string space and this string is then transferred up to the garbage pointer which is then set to the bottom of this string and the string descriptor updated to point to the string in its new position.

The procedure is then repeated all the way down the string space, used strings being shifted upwards to overwrite unused garbage. In our example, a garbage collection would ignore nearly all the strings in the string space except the last, which would be shifted up to the top. We now have a lot of space left for new strings.

If there are a lot of strings in use, the situation could arise that a garbage collection would recover insufficient memory to store another string, in which case a genuine OUT OF MEMORY will occur.

Looking at the program on p139 of the May Newsletter, we are assigining the string "ABCD" to random locations in an array. As the array is filled, the string space will contain many copies of "ABCD", some of which are required, some are not. When the space is full, garbage collection has to scan string space of up to 500 strings and for each it must scan the 500 descriptors in the array. 500*500=250000, which is a lot of work! Hence the pauses in the program.

MEMORY MAPS

Detailed memory maps for the VIC are few and far between. The only one that I have seen was produced (in a hurry I would guess) by Jim Butterfield. In the next few issues, I want to try to give details of RAM/ROM usage along the lines of the details I produced in the IPUG COMPENDIUM a couple of years back. I have started with the RAM usage maps for the VIC and these include labels which, as far as possible, are the ones used by Commodore and other software authors. Where no definitive label exists, I've invented my own. There are also the BASIC 2 references to assist in the conversion of PET programs to VIC (or even vice-versa).

VIC	BASIC 2	LABEL	DESCRIPTION
000-002	000-002	USRPOK	The USR function jump vector
003-004	-	FACINT	Vector to D1AA - convert FAC to integer in A/Y (-32767 to +32767)
005-006	-	INTFAC	Vector to D391 - convert integer in A/Y to floating point in FAC
007	003	CHARAC	Delimiter character used when scanning and setting up strings
007	003	INTEGR	Integer (0-255) after INT has been called
007-008	003-004	INTEGR	Intermediate integer during OR/AND
008	004	ENDCHR	Delimiter character used when scanning strings and in CRUNCH routine
009	-	TABWRK	Temporary workspace used during TAB function
00A	09D	VERCK	Flag to identify LOAD (=0) or VERIFY (=1) in BASIC ROM (see also 093 - VERCKK)
00B	005	COUNT	Number of characters in CRUNCHed line; number of dimensions in array
00C	006	DIMFLG	Array creation flag - =00 if array to be created if not found in table
00D	007	VALTYP	Variable/value type =00 if numeric =FF if string
00E	008	INTFLG	Flag to indicate integer =00 if floating point =80 if integer
00F	009	DORES	Flag to indicate how to handle reserved words during LIST, DATA and CRUNCH
00F	009	GARBFL	Garbage collect flag used to force garbage collect only once during GETSPA routine
010	00A	SUBFLG	Flag to disallow integer/array in FOR variable, array subscript or FN expression
011	00B	INPFLG	Data read routine flag - 00 if INPUT .. 40 if GET .. 98 if READ
012	00C	DOMASK	Comparison flag (> sets B0 .. = sets B1 .. < sets B2)
012	00C	TANSGN	Sign byte used during TAN and SIN
013	00E	CHANNL	File number of current I/O device - suppress prompts etc when not zero
014-015	011-012	LINNUM	Line number integer (0-63999) evaluated by LINGET routine - used by GOTO and GOSUB
014-015	011-012	POKER	Integer value (0-65535) evaluated by GETADR (float-fixed conversion) for POKE PEEK WAIT & SYS
016	013	TEMPPT	Index to next available space on descriptor stack
017-018	014-015	LASTPT	Pointer to last entry on descriptor stack
019-021	016-01E	TEMPST	Temporary stack to hold three string descriptors
022	01F	INDEX	One byte general workspace
022-023	01F-020	INDEX1	Two byte general pointer for strings, memory and so on
024-025	021-022	INDEX2	Two byte general pointer for numbers, memory and so on
026-02A	023-027	RES	Floating point register used as workspace for MULT and DIV routines
028-029	025-026	ADDEND	Holds addend during UMULT routine (when calculating size of arrays)
02B-02C	028-029	TXTTAB	Pointer to start of BASIC TEXT - points to first byte of program
02D-02E	02A-02B	VARTAB	Pointer to start of VARIABLE table - points to one byte after program
02F-030	02C-02D	ARYTAB	Pointer to start of ARRAY table
031-032	02E-02F	STREND	Pointer to end of STRING space - points to lower limit of free space available for strings
033-034	030-031	FRETOP	Pointer to start of STRING space - points to topmost byte of strings storage area
035-036	032-033	FRESPC	Pointer to most recent string in string space; general string pointer
037-038	034-035	MEMSIZ	End of memory available for BASIC (used to set FRETOP after CLR or FRE)

VIC	BASIC 2	LABEL	DESCRIPTION
039-03A	036-037	CURLIN	Current line number being executed (FF in CURLIN+1 indicates direct mode)
03B-03C	038-039	OLDLIN	Holds line number after STOP/END/BREAK for use by CONT
03D-03E	03A-03B	OLDTXT	Holds pointer to first character in current statement for use by CONT
03F-040	03C-03D	DATLIN	Current DATA line number
041-042	03E-03F	DATPTR	DATA pointer used by READ - is left pointing to character after last DATA item read
043-044	040-041	INPTR	Temporary storage of pointer to data in INPUT and READ routines
045-046	042-043	VARNAM	Identity (ie name) of variable to be searched for in variable table
047	044	FDECPT	Pointer into numeric conversion constants (FOUTBL)
047-048	044-045	VARPNT	Pointer to body of variable VARNAM in variable table
049	046	LSTPNT	Index used by LIST to save pointer into text line
049	046	ANDMSK	Holds the first mask parameter in the WAIT command (the AND parameter for unwanted bits)
04A	047	EORMSK	Holds the second mask parameter in the WAIT command (the EOR parameter for testing the byte)
049-04A	046-047	FORPNT	Pointer to current FOR..NEXT variable - also pointer to target variable in LET, INPUT etc.
04B	048	OPPTR	Pointer into operator table OPTAB during FRMEVL
04B-04C	048-049	VARTXT	Temporary storage for TXTPTR during READ, INPUT and GET
04D	04A	OPMASK	Mask used to set up relational operators in FRMEVL
04E-04F	04B-04C	DEFPNT	Pointer to function descriptor in variable space
04E-04F	04B-04C	GRBPNT	Pointer used by garbage collect routine (GARBAG)
04E-052	04B-04F	TEMPF3	Temporary floating point accumulator
050-051	04D-04E	DSCPNT	Pointer to descriptors during string operations
053	050	FOUR6	Length of string variable during garbage collection (ie 7 byte in variables, 3 in arrays)
054-056	051-053	JMPER	JMP xxxx - used to jump into function subroutines
056	053	OLDOV	Holds old overflow value during arithmetic operations
057-05B	054-058	TEMPF1	Temporary floating point accumulator
058-059	055-056	HIGHDS	Pointer used by block transfer routine BLTU (end address of block to be transferred)
058-059	055-056	ARYPNT	Pointer to start of array when initialising it; pointer used by garbage collection
05A-05B	057-058	HIGHTR	Pointer used by block transfer routine BLTU (upper end of block to be transferred)
05C-060	059-05D	TEMPF2	Temporary floating point accumulator
05D	05A	DECCNT	Number of digits after/before decimal point in ASCII-float/float-ASCII conversion (FIN/FOUT)
05E	05B	TENEXP	Exponent evaluated by ASCII-float/float-ASCII (FIN/FOUT)
05F	05C	DPTFLG	Flag decimal point read during ASCII-float conversion (FIN) - B7=1 if it has been
05F-060	05C-05D	LOWTR	Pointer used by block transfer routine BLTU (start address of block to be transferred)
05F-060	05C-05D	LINPTR	Pointer to start of text line found by FNDLIN; pointer to start of line used by LIST
05F-060	05C-05D	VARPTR	Pointer to variable during variable search (PTRGET)
05F-060	05C-05D	GRBTOP	Workspace for garbage collect routine
060	05D	EXPSGN	Sign of exponent during ASCII-floating point conversion (FIN)

VIC	BASIC 2	LABEL	DESCRIPTION
061-063	05E-060	DSCTMP	Temporary string descriptor storage
061-065	05E-062	FAC	Main floating point accumulator 5 bytes are EXP/MO/MOH/MO/LO
066	063	FACSGN	Sign of FAC
067	064	SGNFLG	Sign of number read by ASCII-float routine (FIN)
067	064	DEGREE	Counter used in polynomial evaluation of LOG and TRIG functions
068	065	BITS	Bit overflow for normalisation routines
069-06D	066-06A	ARG	Floating point accumulator used to hold arguments in evaluations
06E	06B	ARGSGN	Sign of argument
06F	06C	ARISGN	Sign of result of arithmetic evaluation
06F-070	06C-06D	STRNG1	General string pointer
070	06D	FACOV	Overflow byte for main floating point accumulator FAC
071	06E	FBUFPT	Pointer into ASCII conversion area (FBUF) used in float-ASCII conversion (FOUT)
071	06E	BUFPTR	Pointer into BASIC input buffer (BUF) during CRUNCH routine
071-072	06E-06F	CURTOL	Workspace used in calculating size of array during array creation
071-072	06E-06F	TEMPTX	Temporary storage of TXTPTR during VAL function
071-072	06E-06F	POLYPT	Pointer to start of constants table during polynomial evaluation
073-08A	070-087	CHRGET	Routine to read next character from BASIC text and identify integer, letter or delimiter
079	076	CHRGOT	As CHRGET but reads current character
07A-07B	077-078	TXTPTR	Pointer into BASIC text used by CHRGET/CHRGOT routine
08B-08F	088-08C	RNDX	Last random number generated - used as seed for RND function

VIC	BASIC 2	LABEL	DESCRIPTION
090	096	STATUS	Serial bus and cassette status byte (ST)
091	09B	STOPFL	Stop key flag - copy of 912F set during TI update - is FE if STOP key pressed
092	09C	TSERVO	TAPE read - timing byte derived from tape and used to adjust main timing constants
093	09D	VERCKK	Flag to indicate LOAD or VERIFY in kernel routine - copy of VERCK (see 00A)
094	0A0	ICHRFL	IEEE/serial deferred output flag - indicates character waiting to be sent
095	0A5	IDATO	IEEE/serial character for output
096	0AB	TEOB	TAPE read - flag to indicate end of data block (ie no data being received from tape)
097	0AD	TEMPX	Temporary storage of X register during BASIN
097	-	TEMPY	Temporary storage of Y register during RS232 character fetch
098	0AE	NFILES	Number of files currently OPEN
099	0AF	DFLTI	Current INPUT device - normally 0 (keyboard)
09A	0B0	DFLTO	Current OUTPUT device - normally 3 (screen)
09B	0B1	TPARIT	TAPE write - parity of bit written to tape
09C	0B2	TBYTFL	TAPE read - flag to indicate byte now completed
09D	-	MSGFLG	Flag to suppress I/O message; bit=0 to suppress (B6 for "I/O ERROR#", B7 for normal messages)
09E	0B4	FNMIDX	Index to filename while checking/transferring cassette filename
09E	0B4	HDRTYP	Header identification for writing to tape
09E	0C0	TERRLG	TAPE read - pass 1 error log - keeps track of error addresses in TERTAB
09F	0B5	CNMIDX	Index to tape buffer while checking/transferring cassette filename
09F	0C1	TCOREC	TAPE read - pass 2 error log - index into TERTAB while errors are corrected during pass 2
0A0-0A2	08D-08F	CTIMR	Jiffy clock - updates 60 times per second - 3 byte integer
0A3	0B7	TSFCNT	TAPE read/write - bit counter for 8 serial bits to send or receive
0A3	-	SEREOI	Serial bus EOI flag - B7=1 if EOI to be sent with character
0A4	0B9	TBTCNT	TAPE read/write - counter for number of pulses read while reading/writing a single bit dipole
0A4	-	SERBYT	Serial bus shift register (byte assembled here during read)
0A5	0BA	CNTDN	TAPE write - countdown to start of data
0A5	-	SHFCNT	Serial bus counter
0A6	0BB	TIDX1	Tape buffer number 1 - counter of number of characters in buffer (buffer written when >= 192)
-	0BC	TIDX2	Tape buffer number 2 - counter of number of characters in buffer (buffer written when >= 192)
0A7	0BD	PASNUM	Count of which pass cassette is currently doing
0A7	0BD	LDRCNT	TAPE write - leader/spacer count - counts number of short pulses
0A7	-	INBIT	RS232 temporary storage for received bit

VIC	BASIC 2	LABEL	DESCRIPTION
0A8	0BE	TERROR	TAPE read - flag error in byte during cassette read
0A8	0BE	TWMI	TAPE write - flag for first half of word marker written
0A8	-	BITCI	RS232 received bits count
0A9	0BF	TBITER	TAPE read - difference between number of short & medium pulses read (used as error indicator)
0A9	0BF	TWM2	TAPE write - flag for second half of word marker written
0A9	-	RINONE	RS232 receive flag - start bit check
0AA	0C2	TMODE	TAPE read - current mode (B7=end of read, B6=data read in progress, B0-3=countdown to data)
0AA	-	RIDATA	RS232 received byte buffer - the byte is assembled here
0AB	0C3	TCKS	TAPE read - block checksum calculated after read is complete
0AB	0C3	TSPCLN	TAPE write - length of leader/spacer required
0AB	-	RIPTY	RS232 received byte parity
0AC-0AD	0C7-0C8	STADD	Start address for LOAD and SAVE; pointer used as destination for tape LOAD
0AC-0AD	0C7-0C8	LPTRS	Pointer used for scroll and INSert - used as pointer to line to be moved
0AE-0AF	0C9-0CA	EA	End address for SAVE; pointer to destination for serial LOAD; end address set by LOAD
0AE-0AF	-	COLPTR	Pointer to colour memory used during INSert
0B0	0CB	TCON1	TAPE write - timing constant
0B1	0CC	TCON2	TAPE write - timing constant
0B2-0B3	0D6-0D7	TBUF	Start address of cassette tape buffer
0B4	0CE	TTIX	TAPE read timing flag
0B4	-	BITTS	RS232 transmit bit count
0B5	0CF	TEOT	TAPE read - end of tape (more precisely end of read operation - all data read)
0B5	-	NXTBIT	RS232 transmit - next bit to send
0B6	0D0	TERRR	TAPE read - error flag (combined from TERROR & TERRR)
0B6	-	RODATA	RS232 transmit - byte to be sent
0B7	0D1	FNLEN	Number of characters in filename
0B8	0D2	LA	Logical address (file number) of current file
0B9	0D3	SA	Secondary address of current file
0BA	0D4	FA	Primary (first) address of current file (ie the device number)
0BB-0BC	0DA-0DB	FNADR	Starting address of filename
0BD	0DD	TCHRI	TAPE read - byte read from tape
0BD	0DD	TCHRO	TAPE write - byte to be written to tape
0BE	0DE	TBLKN	TAPE read/write - number of blocks remaining
0BF	0DF	TBUILD	TAPE read - serial word buffer where byte is assembled

VIC	BASIC 2	LABEL	DESCRIPTION
0C0	0F9	CSW1	Cassette switch flag
0C1-0C2	0FB-0FC	STAL	Start address for LOAD and cassette write
0C3-0C4	-	PTRXX	General pointer used for transfer of vectors and a type 3 cassette LOAD
0C5	097	LSTX	Matrix value of key pressed during previous keyboard scan
0C6	09E	NDX	Number of characters in keyboard buffer
0C7	09F	RVSFLG	Flag to indicate reverse field is on (00 if off, 12 if on)
0C8	0A1	INPLEN	Number of characters on line of input - used to suppress trailing spaces
0C9	0A3	LINLOG	Log of screen line number for start of input line
0CA	0A4	COLLOG	Log of screen column for start of input
0CB	0A6	KEYVAL	Matrix value of key pressed during current keyboard scan
0CC	0A7	CURSOR	Flag to indicate cursor required (0= cursor on, 1=cursor off)
0CD	0A8	BLNCT	Countdown for cursor blink
0CE	0A9	CCHRSV	Character on screen under the cursor while cursor is inverted
0CF	0AA	CURSOO	Flag to indicate cursor is in its inverted phase (if not 00 then CCHRSV is invalid)
0D0	0AC	INPSRC	Flag to indicate input from screen (3) or keyboard (0)
0D1-0D2	0C4-0C5	SCRPTR	Start address of current screen line
0D3	0C6	CPOS	Current cursor position on line (includes wrap around lines)
0D4	0CD	CCTL	Flag to indicate quotes - flips every time quotes encountered & controls cursor
0D5	0D5	SLINEL	Number of characters on screen line (21, 43, 65 or 87 characters)
0D6	0D8	SLNUM	Line number on which cursor is situated
0D7	0D9	SCHARI	Screen value of character on screen read during screen input
0D7	0D9	SCHARO	Last character output to screen
0D7	0D9	TDIPX	TAPE read - bit read from tape (evaluated from dipole pulse pair)
0D7	0D9	TWCKS	TAPE write - block checksum work byte
0D8	0DC	NINS	Number of INSerts still outstanding
0D9-0F1	0E0-0F8	SCRPTH	Screen line table - holds screen line addresses (high bytes) - B7=0 if line is wrap around
0F2	-	SCRWLG	Screen row log used during screen scroll routines
0F3-0F4	-	COLPTR	Pointer to colour memory during scroll and DELete
0F5-0F6	-	KEYMTX	Address of start of current keyboard decoding matrix
0F7-0F8	-	RIBUF	RS232 - pointer to start of receive buffer
0F9-0FA	-	ROBUF	RS232 - pointer to start of transmit buffer
0FB-0FE	-	-	UNUSED
0FF-10A	0FF-10A	ASCWRK	Assembly area for float-ASCII conversion
100-13E	100-13E	TERTAB	Error table set on first pass of tape read, used to correct errors on second pass
13E-1FF	13E-1FF	-	BASIC and hardware stack space

VIC	BASIC 2	LABEL	DESCRIPTION
200-258	200-250	BBUFF	BASIC input buffer - all input is placed here first
259-262	251-25A	LATBL	Table of logical addresses (file numbers)
263-26C	25B-264	FATBL	Table of primary (first) addresses (device numbers)
26D-276	265-26E	SATBL	Table of secondary addresses
277-280	26F-278	KEYBUF	Keyboard buffer - all characters from keyboard placed here during keyboard scan
281-282	-	LORAM	Lowest available RAM byte
283-284	-	HIRAM	Highest available RAM byte
285	-	TMOOFF	IEEE/serial timeout defeat flag - settable on VIC but never checked
286	-	COLNOW	Current colour code (0=BLACK, 1=WHITE etc)
287	-	COLCUR	Colour of character under the cursor
288	-	SCRAMH	High byte (ie page number) of screen memory
289	-	MAXKEY	Size of keyboard buffer (maximum number of characters storable)
28A	-	RPTFLG	Flag for which keys to repeat (=00 for cursor keys, =80 for ALL keys, =40 for NO keys)
28B	-	RPTCNT	Repeat speed counter
28C	-	RPTDLY	Repeat delay countdown before key will repeat
28D	098	SHFFLG	Shift key flag (B0=SHIFT, B1=CBM key, B2=CTRL key)
28E	-	SHFLST	Last shift key image for debounce software
28F-290	-	CHKSHF	Vector (EBDC) for shift key + CBM key check
291	-	FLIP	Screen case toggle flag - if B7=1 then toggle is disabled
292	-	PUSFLG	Screen push-down flag used to enable push-down during screen input (non-zero to enable)
293	-	M51CTR	RS232 Pseudo 6551 - control register
294	-	M51CDR	RS232 Pseudo 6551 - command register
295-296	-	M51AJB	RS232 spare control bytes (UNUSED)
297	-	RSSTAT	RS232 status register
298	-	BITNUM	RS232 number of bits to send/receive (eventually transferred to BITCI)
299-29A	-	BAUDOF	RS232 baud rate timing constant for 9114-9115 timer
29B	-	RIDBE	RS232 input buffer pointer - points to latest character input (end of buffer)
29C	-	RIDBS	RS232 input buffer pointer - points to first available character (start of buffer)
29D	-	RODBS	RS232 output buffer pointer - points to last character output to RS232 (start of buffer)
29E	-	RODBE	RS232 output buffer pointer - points to next available buffer space (end of buffer)
29F-2A0	-	CINVTM	Temporary store for interrupt vector during tape operations
2A1-2FF	-	-	UNUSED

VIC	BASIC 2	LABEL	DESCRIPTION
			BASIC INTERPRETER - VECTOR TABLE
300-301	-	WNDERR	Vector (C43A) - error message X = error message number
302-303	-	WNDMN	Vector (C483) - main line input and decode - entered just before line is read from keyboard
304-305	-	WNDCRN	Vector (C57C) - CRUNCH routine - new BASIC line can be intercepted & new keywords identified
306-307	-	WNDLST	Vector (C71A) - LIST intercepts LIST before character in A is printed or expanded to keyword
308-309	-	WNDGNE	Vector (C7E4) - main execution loop - intercepts just before next statement is executed
30A-30B	-	WNDEVL	Vector (CE86) - arithmetic element evaluation - allows new functions to be evaluated
30C	-	SYSA	Accumulator storage for SYS - passed to SYS routine and updated on exit
30D	-	SYSX	X register storage for SYS - passed to SYS routine and updated on exit
30E	-	SYSY	Y register storage for SYS - passed to SYS routine and updated on exit
30F	-	SYSS	Status reg storage for SYS - passed to SYS routine and updated on exit
310-313	-	-	UNUSED
			KERNEL - VECTOR TABLE
314-315	-	CIRQ	Main IRQ vector - initialised to EABF
316-317	-	CBRK	BRK instruction vector - initialised to FED2 (same routine as RESTORE key uses)
318-319	-	CNMI	NMI vector - initialised to FEAA (checks if RESTORE and STOP pressed)
31A-31B	-	WNDOPN	OPEN vector (F40A) normal parameters should be set on entry
31C-31D	-	WNDCLS	CLOSE vector (F34A) normal parameters should be set on entry
31E-31F	-	WNDCKI	Set input device (F2C7) same as CHKIN
320-321	-	WNDCKO	Set output device (F309) same as CHKOUT
322-323	-	WNDCLC	Restore normal I/O (F3F3) same as CLRCHN
324-325	-	WNDCHI	Input a character (F20E) same as CHRIN
326-327	-	WNDCHO	Ouput a character (F27A) same as CHROUT
328-329	-	WNDSTP	Check for STOP key (F770) same as STOP
32A-32B	-	WNDGET	Get a single character (F1F5) same as GETIN
32C-32D	-	WNDCLL	Close all files (F3EF) same as CLALL
32E-32F	-	WNDRST	UNUSED - but initialised to FED2 (BRK vector)
330-331	-	WNDLD	LOAD vector (F549) intercepts LOAD routine with start address in C3-C4 & parameters set
332-333	-	WNDSV	SAVE vector (F685) intercepts SAVE with start in C1-C2, end in AE-AF & parameters set
334-33B	-	-	UNUSED
33C-3FC	33A-3F9	CBUF1	Cassette buffer (192 bytes long)
3FD-3FF	-	-	UNUSED

VIC WITH NO MEMORY EXPANSION:

VIC	BASIC 2	LABEL	DESCRIPTION
0000-03FF	0000-03FF	ZPAGE	General workspace for BASIC and OPERATING SYSTEM (0000-008F is BASIC only)
0400-0FFF	-	EXP3K	EMPTY
1000-1DFF	0400-....	TEXT	BASIC text space - program starts at 1001
1E00-1FF9	8000-83E7	SCREEN	Screen memory
1FFA-1FFF	83E8-83FF		UNUSED but not available (is corrupted by clearing screen etc)
2000-3FFF	-	BLOCK 1	EMPTY
4000-5FFF	-	BLOCK 2	EMPTY
6000-7FFF	-	BLOCK 3	EMPTY
8000-83FF	-	CHRGEN	Character generator for normal upper case + graphics
8400-87FF	-		Character generator for inverted upper case + graphics
8800-8BFF	-		Character generator for normal upper case + lower case
8C00-8FFF	-		Character generator for inverted upper case + lower case
9000-900F	-	VIC	VIC chip registers
9010-910F	-		NOT ACCESSIBLE
9110-911F	-	VIA1	VIA number 1 (interrupt connected to IRQ line)
9120-912F	-	VIA2	VIA number 2 (interrupt connected to NMI line)
9130-93FF	-		NOT ACCESSIBLE
9400-95F9	-		UNUSED colour RAM (each byte is only 4 bits)
95FA-95FF	-		UNUSED
9600-97F9	-	COLRMX	COLOUR RAM (each byte is 4 bits wide)
97FA-97FF	-		UNUSED
9800-9BFF	-	IOBLK2	INPUT/OUTPUT expansion area I/O block 2
9C00-9FFF	-	IOBLK3	INPUT/OUTPUT expansion area I/O block 3
A000-BFFF	-	BLOCK5	ROM cartridge expansion area (used for auto started games etc)
C000-E0F5	-	BASIC	BASIC interpreter ROM number xxxxxxx (note that there are 245 bytes in the KERNEL ROM)
E0F6-FFFF	-	KERNEL	KERNEL ROM number xxxxxxx

VIC WITH 3k RAM EXPANSION ONLY:

VIC	BASIC 2	LABEL	DESCRIPTION
0000-03FF	0000-03FF	ZPAGE	General workspace for BASIC and OPERATING SYSTEM (0000-008F is BASIC only)
0400-1DFF	0400-....	TEXT	BASIC text space - program starts at 0401
1E00-1FF9	8000-83E7	SCREEN	Screen memory
1FFA-1FFF	83E8-83FF		UNUSED but not available (is corrupted by clearing screen etc)
2000-3FFF	-	BLOCK 1	EMPTY
4000-5FFF	-	BLOCK 2	EMPTY
6000-7FFF	-	BLOCK 3	EMPTY
8000-83FF	-	CHRGEN	Character generator for normal upper case + graphics
8400-87FF	-		Character generator for inverted upper case + graphics
8800-8BFF	-		Character generator for normal upper case + lower case
8C00-8FFF	-		Character generator for inverted upper case + lower case
9000-900F	-	VIC	VIC chip registers
9010-910F	-		NOT ACCESSIBLE
9110-911F	-	VIA1	VIA number 1 (interrupt connected to IRQ line)
9120-912F	-	VIA2	VIA number 2 (interrupt connected to NMI line)
9130-93FF	-		NOT ACCESSIBLE
9400-95F9	-		UNUSED colour RAM (each byte is only 4 bits)
95FA-95FF	-		UNUSED
9600-97F9	-	COLRMX	COLOUR RAM (each byte is 4 bits wide)
97FA-97FF	-		UNUSED
9800-9BFF	-	IOBLK2	INPUT/OUTPUT expansion area I/O block 2
9C00-9FFF	-	IOBLK3	INPUT/OUTPUT expansion area I/O block 3
A000-BFFF	-	BLOCK5	ROM cartridge expansion area (used for auto started games etc)
C000-E0F5	-	BASIC	BASIC interpreter ROM number xxxxxxx (note that there are 245 bytes in the KERNEL ROM)
E0F6-FFFF	-	KERNEL	KERNEL ROM number xxxxxxx

VIC WITH 8k, 16k OR 24k MEMORY EXPANSION:

VIC	BASIC 2	LABEL	DESCRIPTION
0000-03FF	0000-03FF	ZPAGE	General workspace for BASIC and OPERATING SYSTEM (0000-008F is BASIC only)
0400-0FFF	-	EXP3K	3k RAM expansion area - available for POKEs and machine code but not used by VIC
1000-11F9	8000-83F7	SCREEN	Screen memory
11FA-11FF	83E8-83FF		UNUSED but not available (is corrupted by clearing screen etc)
1200-1FFF	0400-....	TEXT	BASIC text space - program starts at 1201
2000-3FFF	-	BLOCK 1	First 8k RAM expansion block - total RAM expansion is 8k
4000-5FFF	-	BLOCK 2	Second 8k RAM expansion block - total RAM expansion is 16k
6000-7FFF	-	BLOCK 3	Third 8k RAM expansion block - total RAM expansion is 24k
8000-83FF	-	CHRGEN	Character generator for normal upper case + graphics
8400-87FF	-		Character generator for inverted upper case + graphics
8800-8BFF	-		Character generator for normal upper case + lower case
8C00-8FFF	-		Character generator for inverted upper case + lower case
9000-900F	-	VIC	VIC chip registers
9010-910F	-		NOT ACCESSIBLE
9110-911F	-	VIA1	VIA number 1 (interrupt connected to IRQ line)
9120-912F	-	VIA2	VIA number 2 (interrupt connected to NMI line)
9130-93FF	-		NOT ACCESSIBLE
9400-95F9	-	COLRMX	COLOUR RAM (each byte is 4 bits wide)
95FA-95FF	-		UNUSED
9600-97F9	-		UNUSED colour RAM (each byte is only 4 bits)
97FA-97FF	-		UNUSED
9800-9BFF	-	IOBLK2	INPUT/OUTPUT expansion area I/O block 2
9C00-9FFF	-	IOBLK3	INPUT/OUTPUT expansion area I/O block 3
A000-BFFF	-	BLOCK5	ROM cartridge expansion area (used for auto started games etc)
C000-E0F5	-	BASIC	BASIC interpreter ROM number xxxxxxx (note that there are 245 bytes in the KERNEL ROM)
E0F6-FFFF	-	KERNEL	KERNEL ROM number xxxxxxx

MATTERS ARISING...

Colin Mair informs me that there is a bug in Nick Higham's program on p51, March issue, for POKing into REM statements. His program looks for the value 143 in every location from 1024 to the end of BASIC text and it is possible to find this value in the locations used to hold the link addresses and line numbers. POKing these could cause program corruption, or even a crash when listing !

The solution lies in looking for the end of BASIC line - indicated by a location holding zero. When this is found, the next four locations, which hold the link address and line number, are skipped.

```
1 A=42:REM A=124 FOR OLD ROM
2 M=PEEK(A)+256*PEEK(A+1):INPUT C
3 FOR I=1024 TO M
4 IF PEEK(I)=0 THEN I=I+5
5 IF PEEK(I)=143 THEN POKE I+1,C
6 NEXT I:LIST
```

The cassette merge (p92) was, as you may have realised, for BASIC 2. For BASIC 4, the POKE14,1 should be POKE16,1 and for the VIC-20, POKE19,1. In addition, for other locations:

BASIC 1	2 & 4	VIC-20	
611	175	153	Input device # (CMD)
525	158	198	Keyboard character count
527	623	631	Keyboard buffer.
3	14/16	19	I/O off-normal flag

It has been pointed out that this is an excellent way of loading a VIC with a program 'saved' on a PET/CBM.

--oOo--

SHOP WINDOW

The computer-aided design package, Path-Trace, mentioned on p125, comes from Ergon Design, 75, Gilhams Avenue, Banstead, Surrey, SM7 1QW. Tel: 01-394 0733. The program requires a 'Supersoft' high resolution graphics board (reviewed Nov '81 and listed at £ 149).

Many of you have given the odd sideways glance at 'Uncle Clive's' prices, especially the 100Kbyte micro-floppy. From late September a 3" disk unit will be marketed in the UK by BATS-NCL of Regents Park Road, London (tel: 01-349 4511). One-off price £ 65, extra disks cost about £ 4.50. The system, called the mcd-1, is being configured into three different products by BMB Computers of Woking. The first is for the VIC range using VIC's DOS, the second is an IEEE-488 unit for the larger Commodore machines, and the third is a CP/M version. The drive is made by Metrimpex, a Hungarian company in Budapest. A unit is also expected from Hitachi with a higher performance.

Looking for spare 65xx chips ? Try Datatext (McLelland) Ltd., Unit 3A, Bankfield Industrial Estate, Kitson Road, Leeds, LS10 3YY. Tel: (0532) 455169. Typical prices are R6502P £ 4.54, R6520P £ 2.64, R6522P £ 4.00 and R6551P £ 5.63. The 2MHz versions cost a little more. UK carriage free, first class letter post by return.

Supersoft have extended their range of software to complement their hi-res graphics boards. About ten new utility packages are available on cassette or disk which include hi-res screen save, dump to printer (various), function plotting. MX-LIST and MX-DUMP can be used to generate genuine 8 x 8 PET graphics characters (on a 7 x 6 matrix on the 4022 !). Supersoft also offer the Epson MX-80 F/T Type 2 for only £ 450 + VAT. More details from Supersoft, First Floor, 10-14, Canning Road, Wealdstone, Harrow, Middlesex, HA3 7SJ. Tel: 01-861 1166.

VIC users will no doubt have seen advertised the 40-column capability with 32K RAM expansion provided by the Colour Writer VIC Expansion at £ 220 + VAT. Details from Beelines (Bolton) Ltd., FREEPOST, Bolton, BL3 6YZ. Tel: (0204) 382741.

Compsofts DMS has been around some while now, but the latest release has its own internal mini-word processor. Facilities are limited to cover only those needed to produce personalised letters which can the be addressed to selected clients. Details from Compsoft Ltd., Hallams Court, Shamley Green, Nr. Guildford, Surrey, GU4 8QZ. Tel: (0483) 898545.

The Spriinter (yes, double-i) is an interface unit for printers which incorporates a standard converter, a spooler, a protocol converter and 32K buffer. The unit incorporates a 6502C processor (2.4MHz) and can be used as a stand-alone computer. I/O comprises two serial bi-directional ports, one parallel in and one out which may optionally be made Centronics-compatible and two ports, one in, one out, IEEE-488 which with optional software, may act as a limited-function IEEE-488 bus controller. Further information from Mutek, Quarry Hill, Box, Wilts. Tel: Bath (0225) 743289.

I have always fancied a no-break power supply for my PET to avoid those disasters on the mains during thunderstorms. Backpack is a relatively small unit that will keep PET (and disk drive if so fitted) going for 15 minutes after power failure. Backpack cost £ 140 + VAT from Wego Computers Ltd., 22a, High Street, Caterham, Surrey, CR3 5UA. Tel: 49235.

Now to plotting. MICROPLOT is claimed to be the first plotter designed specifically for the PET. Taking A4 size paper, it is driven by the IEEE-488 interface. Plotter, digitiser, firmware and accessories can all be supplied by Business Electronics, Rownhams House, Rownhams, Southampton, SO1 8AH. Tel: (0703) 738248.

The MW-1000 Mini-Winchester is a compact desk-top unit giving up to 12Mbyte of hard disk storage under CP/M or PET DOS, or both. The 3Mbyte version costs £ 2538. Further details from Small Systems Engineering Ltd., 2-4, Canfield Place, London, NW6 3BT. Tel: 01-328 7145.

Numerous accessories for the VIC-20 are available from Stack. VICKIT is a 'Toolkit'-type EPROM adding 9 commands to BASIC for program development and supporting a LIST command with space-bar pause. Price £ 25 + VAT. VICKIT II is a 4K EPROM incorporating the above, but in addition has twelve statements to greatly simplify programming to produce hi-res VIC graphics. Both VICKITs fit into the Stack Storeboard, or ROM Switchboard. Also available is a light-pen and a low-cost RS232 interface (£ 22.99 + VAT). Further goodies and information are available from Stack Computer Services Ltd., 290-298, Derby Road, Bootle, Liverpool, L20 8LN. Tel: 051-933 5511. Contact Neil Cornes.

Signal processing is accommodated by the Model 4096 package which features the following: Firmware - 4K EPROM giving FFT (1024 point) of input or computer generated signals with automatic scaling for plotting. Correlation, auto- or cross- as required; amplitude statistics, spectral density, double-density graphics and high speed histogram plotting. Hardware - wide ranging programmable sampling clock, 4K RAM per channel, 8-bit A-D and D-A per channel with -5 to +5 volt range. The unit can function as a spectrum analyser, digital correlator or storage oscilloscope. Price £ 399 + VAT. Further details from Micro Enterprises, 88 Park Hill, London SW4 9PB. Tel 01-622 6816.

Designers of add-ons may be interested in the components and accessories service from WASEC. The facility provides a broad range of IEEE-488 and IEC-625 related semiconductors, connectors and literature. Full details in a leaflet from WASEC, tel: 01-647 3199.

Another data acquisition system for the IEEE-488 bus comes from Biodata and is known as Microlink. A wide range of plug-in modules is available and prices are competitive. Contact Biodata Ltd., 6, Lower Ormond St., Manchester M1 5QF for details - tel: 061-236 1283.

Lamar Instruments produce two products of interest, a ROM simulator and a single board computer called Superkim. The latter is an upmarket KIM-1/SYM-1/AIM-65 with Multi-tasking and is KIM-1 compatible in many ways. Details from Lamar Instruments, 2107, Artesia Blvd, Redondo Beach, California 90278.

R.D.G.

--oOo--

MICROCHESS ON ICE

By Walter Green

After many months of investigation, I now have a copy of Microchess which, loaded and run on an old-ROM PET, will behave like any other version but with extra facilities. At any stage of the game, typing '-' (minus) exits to BASIC. From BASIC a SYS call will enter the monitor which forms part of the revised program. M 1900 19B0 will display zero page as it was when '-' was typed. G 17F0 returns to BASIC.

Type 'POKE6189,51:RUN' and the board appears with white's King Pawn on K4, but black will not reply. Make as many moves as you like, black and white, copy a game from a book; make any piece appear on any square, or vanish ! Play a game with a pal - if you wish to adjourn the game, place a blank cassette in your deck and type SYS2062 and make a copy. Reload the copy and RUN. The board appears with the pieces in the adjourned position. You do not wish to continue ? Type 'R' and they return to the initial state for another game of Microchess.

If any Microchess owners are interested in these changes, they may contact me at 151, The Hatherley, Basildon, Essex, SS14 2QH.

--oOo--

DISK TO TAPE SAVE

By Brian Grainger

This is the first in what I hope will become a series of articles. The aim is to introduce members to some of the software that is available either in the regional groups or through the national library. Sometimes the articles will review software that is available and sometimes, like this one, complete programs will be presented.

This program started life in the Watford regional group before coming into the hands of the North Herts Group. Its purpose is to take a copy of all programs from a disk and put them onto a cassette. As most ICPUG software now comes on disk this could be a useful way for regional group organisers of getting cassette copies of programs to copy to those ICPUG members without disks. This first program uses the standard CBM 'Save' ROM routines so can be quite slow for a full disk. It is written in BASIC4 but REM statements indicate changes required for BASIC2.

```
 10 REM IDEA & ORIGINAL PROGRAM
 20 REM
 30 REM BY GEOFF PYKE & MARTIN SIMPSON
 40 REM
 45 REM EXTENSIVELY MODIFIED FOR RABBIT
 46 REM
 47 REM BY PHIL MORTIBOY & B. GRAINGER
 48 REM
 50 REM MAY 1982
 60 REM
 65 REM ******************************
 70 REM *PLEASE COPY BUT DO NOT SELL*
 75 REM ******************************
 80 REM
 85 REM WRITTEN FOR USE ON A BASIC 4
 90 REM PET WITH DOS 1 OR DOS 2A DISK
 95 :
 96 DIMD$(75),A(255)
100 PRINT"<clr><rvs>**************************************
    ***";
110 PRINT"*********DISK TO TAPE BACK-UP **********";
```

```
120 PRINT"****************************************"
130 PRINT"<dn>THIS PROGRAM WILL COPY ALL THE PROGRAMS "
140 PRINT"      FROM ANY DOS 1 OR DOS 2A DISK "
150 PRINT"<dn>AND SAVE THEM ONTO A TAPE CASSETTE"
155 PRINT"          IN RABBIT FORMAT."
160 PRINT"<dn>        ***********************"
170 PRINT"        **<rvs>PLEASE ENTER CHOICE<off>**"
180 PRINT"        ***********************"
190 PRINT"<dn>     1: COPY ALL PROGRAMS TO TAPE
200 PRINT"<dn>     2: SELECTIVE COPY OF PROGRAMS
210 GETC$:IFC$<>"1"ANDC$<>"2"GOTO210
235 PRINT"<dn><rvs>**************************************
    **";
240 PRINT"*********PLACE DISK IN DRIVE 0**********";
250 GOSUB2200
260 OPEN15,8,15,"IO":OPEN4,8,4,"#":Z$=CHR$(O)
265 GOSUB2300
270 PRINT"<clr><rvs>*************************************
    ***";
280 PRINT"**NAME:"DN$"  ID="ID$"  DOS "DO$"**";
290 PRINT"****************************************"
300 T=18:S=1:D=0:GOSUB2500
305 IFC$="2"THENGOSUB2600
310 PRINT"<clr>":POKE212,1:SYS63628:REM TURN CASSETTE ON
    (BAS2 SYS63559)
320 POKE249,52:POKE59411,61:REM TURN CASSETTE MOTOR OFF
330 PRINT"<clr>":X=32768:POKE59468,14:I=0
340 READA:IFA>=0THENI=I+1:GOTO340
350 GOSUB2700
360 REM SET UP M/C ON SCREEN AND EXECUTE
470 RESTORE:FORA=33768-ITO33767:READB:POKEA,B:NEXT:POKE
    33767,256-I
480 POKE209,PEEK(32768)
490 POKE218,1
500 POKE219,128
510 SYS33753
1990 :
1991 REM ****************************
1992 REM * CHECK DISK ERROR CHANNEL *
1993 REM ****************************
1994 :
2000 INPUT#15,EN,ED$,T1,S1
```

```
2010 IFEN>19THENPRINT"<clr><rvs>DISK ERROR<dn>":PRINTED$:
     CLOSE4:CLOSE2:CLOSE15:END
2020 RETURN
2090 :
2091 REM ************************
2092 REM * LOAD DISK BUFFER WITH *
2093 REM * RQD SECTOR DATA BLOCK *
2094 REM ************************
2095 :
2100 PRINT#15,"U1"4;0;T;S:GOSUB2000:RETURN
2190 :
2191 REM ********************
2192 REM * WAIT FOR 'RETURN' *
2193 REM ********************
2194 :
2200 PRINT"<rvs>****PRESS RETURN WHEN YOU ARE READY*****";
2210 PRINT"****************************************"
2220 GETA$:IFA$<>CHR$(13)GOTO2220
2230 RETURN
2290 :
2291 REM ************************************
2292 REM * READ DISK NAME & ID & DOS VERSION *
2293 REM ************************************
2294 :
2300 T=18:S=0:GOSUB2100:GOSUB2400
2310 DN$="":FORI=1TO16:DN$=DN$+CHR$(A(I+143)):NEXTI
2320 ID$=CHR$(A(162))+CHR$(A(163))
2330 DO$=CHR$(A(165))+CHR$(A(166)):RETURN
2390 :
2391 REM ****************************
2392 REM * GET DATA FROM DISK BUFFER *
2393 REM * AND PLACE INTO ARRAY A( ) *
2394 REM ****************************
2395 :
2400 FORB=0TO255:GET#4,A$:GOSUB2000
2410 A$=A$+Z$:A(B)=ASC(A$):NEXTB:RETURN
2490 :
2491 REM **********************
2492 REM * READ DISK DIRECTORY *
2493 REM **********************
2494 :
2500 GOSUB2100:GOSUB2400:T=A(0):S=A(1):REM NEXT TRACK/
     SECTOR
```

```
2520 FORB=2TO226STEP32
2535 IFA(B)<>130GOTO2560
2550 D=D+1:D$(D)="":FORI=1TO16:D$(D)=D$(D)+CHR$(A(B+I+2)):
     IFA(B+I+3)=160THENI=16
2555 NEXTI
2560 NEXTB:IFS<>255GOTO2500
2570 RETURN
2590 :
2591 REM ***************************
2592 REM * SELECT PROGRAMS TO COPY *
2593 REM ***************************
2594 :
2600 PRINT"<clr>":FORA=1TOD
2610 PRINTD$(A);"   < COPY?  (Y/N) ";
2620 GETA$:IFA$="N"THEND$(A)="":GOTO2630
2625 IFA$<>"Y"GOTO2620
2630 PRINTA$:PRINT
2640 NEXTA:RETURN
2690 :
2691 REM ****************************************
2692 REM * POKE NAME LENGTH, NAME, PROGRAM START  *
2693 REM *   TO SCREEN FOR EACH PROGRAM TO COPY   *
2694 REM ****************************************
2695 :
2700 FORA=1TOD
2710 IFD$(A)=""THEN2760
2715 L=LEN(D$(A)):POKEX,L
2720 FORB=1TOL:POKEX+B,ASC(MID$(D$(A),B)):NEXT
2725 X=X+B:OPEN2,8,2,D$(A)+",P":GOSUB2000
2730 GET#2,A$:LS=ASC(A$÷Z$)
2735 GET#2,A$:HS=ASC(A$+Z$)
2740 IFLS=1ANDHS=4THENLS=0: REM NEEDED FOR RABBIT
2745 POKEX,LS:X=X+1:POKEX,HS:X=X+1
2750 CLOSE2
2755 IFX>=33768-I-20THENA=D
2760 NEXTA:POKEX,0
2770 RETURN
19980 :
19981 REM **********************
19982 REM *   PET SPECIFIC M/C  *
19983 REM **********************
```

```
19990 :
19991 REM **************************************************
      **************
19992 REM * FOR USE WITH STANDARD PET 'SAVE' USING CASSETTE
       NUMBER ONE  *
19993 REM **************************************************
      **************
19994 :
20000 DATA 169,1        :REM LDA #$01      (START)
20010 DATA 133,212      :REM STA $D4
20020 DATA 169,0        :REM LDA #$00
20030 DATA 133,150      :REM STA $96
20040 DATA 164,209      :REM LDY $D1
20050 DATA 177,218      :REM LDA ($DA),Y
20060 DATA 133,251      :REM STA $FB
20070 DATA 200          :REM INY
20080 DATA 177,218      :REM LDA ($DA),Y
20090 DATA 133,252      :REM STA $FC
20100 DATA 32,227,246   :REM JSR $F6E3     (BAS2 DATA 32,164,
                                            246 JSR $F6A4)
29990 :
29991 REM **************
29992 REM * COMMON M/C *
29993 REM **************
29994 :
30000 DATA 160,0        :REM LDY #$00
30010 DATA 177,218      :REM LDA ($DA),Y
30020 DATA 73,128       :REM EOR #$80
30030 DATA 145,218      :REM STA ($DA),Y
30040 DATA 164,209      :REM LDY $D1
30050 DATA 152          :REM TYA
30060 DATA 170          :REM TAX
30070 DATA 200          :REM INY
30080 DATA 200          :REM INY
30090 DATA 177,218      :REM LDA ($DA),Y
30100 DATA 133,209      :REM STA $D1
30110 DATA 201,0        :REM CMP #$00
30120 DATA 208,3        :REM BNE LAB2
30130 DATA 76,22,253    :REM JMP $FD16     (BAS2 DATA 76,209,
                                            252  JMP $FCD1)
30140 DATA 24           :REM CLC           (LAB2)
```

```
30150 DATA 138          :REM TXA
30160 DATA 105,3        :REM ADC #$03
30170 DATA 101,218      :REM ADC $DA
30180 DATA 133,218      :REM STA $DA
30190 DATA 144,2        :REM BCC LAB3
30200 DATA 230,219      :REM INC $DB
30210 DATA 169,0        :REM LDA #$00      (LAB3)
30220 DATA 133,150      :REM STA $96
30230 DATA 133,157      :REM STA $9D
30240 DATA 169,8        :REM LDA #$08
30250 DATA 133,212      :REM STA $D4
30260 DATA 32,93,243    :REM JSR $F35D     (BAS2 DATA 32,41,2
                                            43   JSR $F329)
30270 DATA 208,0        :REM BNE START
30280 DATA -1           :REM END OF DATA
```

As a means of speeding up the program a version was created that used the 'Rabbit' save routines. It will need a 'Rabbit' chip (or relocated 'Rabbit' machine code) to work. The routine below should replace lines 19980-20100 in the original and then a whole disk can be saved in about 17 minutes ! I used relocated 'Rabbit' code in the routine below but instructions are given on how to change it for the various chip versions. Note it does NOT work with the original RAM 'Rabbit' that used the first cassette buffer for storage. Apart from the speed this type of program is extremely useful for making a cassette backup of disks so that when the day comes to have your disk away for repairs, or your disk is inadvertently erased, you may get another copy of your programs easily.

In both versions of the program it is up to the user to ensure the disk does not contain programs that are too long for the available RAM space.

Next Newsletter I hope to provide the routine to do the 'SAVE' in 'Arrow' format.

```
19981 REM ***********************
19982 REM * RABBIT SPECIFIC M/C *
19983 REM ***********************
19990 :
19991 REM **************************************************
      ******************
19992 REM *   FOR USE WITH 'RABBIT' USING 2ND CASSETTE BUFF
      ER FOR STORAGE   *
19993 REM **************************************************
      ******************
19994 REM * HUNT YOUR 'RABBIT' FOR THE BYTES C9 53. DISASSE
      MBLE RABBIT FROM *
19995 REM * THE 2ND VALUE AND YOU WILL SEE CMP #$53, BNE
                                  $03, JMP $ABCD.    *
19996 REM * USE JSR $ABCD IN LINE 20220
                          *
19997 REM **************************************************
      ******************
19998 :
20000 DATA 169,1        :REM LDA #$01      (START)
20010 DATA 141,93,3     :REM STA $035D
20020 DATA 169,5        :REM LDA #$05
20030 DATA 141,99,3     :REM STA $0363
20040 DATA 164,209      :REM LDY $D1
20050 DATA 169,0        :REM LDA #$00
20060 DATA 153,76,3     :REM STA $034C,Y
20070 DATA 136          :REM DEY           (LAB1)
20080 DATA 177,218      :REM LDA ($DA),Y
20090 DATA 153,76,3     :REM STA $034C,Y
20100 DATA 152          :REM TYA
20110 DATA 208,247      :REM BNE LAB1
20120 DATA 164,209      :REM LDY $D1
20130 DATA 177,218      :REM LDA ($DA),Y
20140 DATA 141,95,3     :REM STA $035F
20150 DATA 200          :REM INY
20160 DATA 177,218      :REM LDA ($DA),Y
20170 DATA 141,96,3     :REM STA $0360
20180 DATA 165,201      :REM LDA $C9
20190 DATA 141,97,3     :REM STA $0361
20200 DATA 165,202      :REM LDA $CA
20210 DATA 141,98,3     :REM STA $0362
20220 DATA 32,148,121   :REM JSR $7994
```

COMMODORE COLUMN

The Commodore Computer Show this year was spread out over a much greater floor space this year, so much so that the attendance looked poor since there was ample room to move. However, the show was very successful with attendance in excess of 8,000 over the three days. The show had its moments, especially when one software dealer found that someone else was apparently selling his wares ! There were over 150 exhibits, displaying over 500 innovations. VIC-20 sales were announced as over 40,000 since December.

Talking to a member the other day, it appears now that Commodore admit to a bug in DOS2.5 (8050) where if one tries to copy all files on a disk to another, it hangs up around the 8th file. This has been cured by the upgrade, but no fix for 2.5 users has been proffered.

Commodore's new 16-bit machine, the DX256, is reported as being priced from $2,995 in the US.

Dealers having difficulty recruiting sales and technical staff are being aided by Commodore in a £ 30,000 nationwide recruitment project to find up to 60 more salesmen. Dealers are also sceptical about the appointment of Mills Associates, well known for their ICL maintenance service, as the authorised nationwide maintenance contractor for PET systems. The service is designed for users with more than one machine on site, who do not require the 24-hour response of a full service contract.

R.D.G.

--oOo--

ODD BIT

In nature, nothing is ever right. Therefore, if everything is going right.... something is wrong.

--oOo--

4032/8032 STOP KEY DISABLE

The usual way of disabling the STOP key whilst keeping the TI/TI$ 'jiffy' clock running is by shifting the IRQ vector to a routine in RAM that updates the clock but resets the zero-page key image before joining the interrupt sequence. This is fine for machines with 1/60th second interrupts, i.e. BASICs 1 and 2, but unfortunately BASIC 4 8032s and 12" (Fat-40) 4032s use a 1/50th second interrupt with a software patch to speed up the clock which the above routine by-passes with the result that the clock runs at only 5/6ths of normal speed - highly embarassing if your program includes a large digital time display !

As far as I know the 'speed up' patch is not listed in any BASIC 4 ROM tables, but since for the 4032 the relevent counter is at $03ED I was able to locate it by searching ROM for the consecutive bytes ED 03 and found it at $E42E-$E441. I do not have an 8032 so have not been able to check the corresponding location [Try $E431-$E441 - Ed].

Just pointing the IRQ vector at the patch doesn't work as it exits directly to $E458 so the STOP key disable cannot be inserted. However, by reconstructing the patch in RAM, pointing the IRQ vector at it and following it with the usual 'stop key disable' routine, the following solution results:

```
          4032            8032

CLK1 JSR $FFEA CLK1 JSR $FFEA ;update clock, test stop key
     INC $03ED      INC $F8   ; increment counter
     LDA $03ED      LDA $F8
     CMP #6         CMP #6    ;6th interrupt ?
     BNE CLK2       BNE CLK2  ; no - cont stop key disable
     LDA #0         LDA #0    ;yes, so
     STA $03ED      STA $F8   ;reset counter and
     BEQ CLK2       BEQ CLK2  ;update clock an extra time
CLK2 LDA #$FF CLK2  LDA #$FF
     STA $9B        STA $9B   ;disable stop key
     JMP $E458      JMP $E458
```

A hex dump and BASIC loader (using strings, not DATA statements to avoid interfering with DATA in the main program) for the 4032 are presented and include the usual disable (SYS900) and enable (SYS938) routines. They are located at $0384 - $03B4 in the second cassette buffer since $033A - $0380 is now used by DOS, but obviously can be relocated elsewhere.

```
10 REM ** STOP KEY DISABLE **
11 REM    W.LYONS 7.4.1982
12 :
13 A$="120169143133169003133145088096"
14 A$=A$+"032234255238237003173237003201006208007"
15 A$=A$+"169000141237003240236169255133155076088228"
16 A$=A$+"120169085133144169228133145088096"
17 FOR I=0TO48:POKEI+900,VAL(MID$(A$,3*I,3)):NEXT
18 SYS900:REM DISABLE STOP KEY (SYS938 ENABLE)
```

```
0384 78 A9 8F 85 90 A9 03 85
038C 91 58 60 20 EA FF EE ED
0394 03 AD ED 03 C9 06 D0 07
039C A9 00 8D ED 03 F0 EC A9
03A4 FF 85 9B 4C 58 E4 78 A9
03AC 55 85 90 A9 E4 85 91 58
03B4 60 00 00 00 00 00 00 00
```

William Lyons

--oOo--

MEMBERS PRIVATE SALES & WANTS

Two chips for sale - rock bottom offer ! COMMAND-O and POWER - both for 8032, socket UD12 ($9000) and complete with manuals. £ 35.00 each. Phone Jim MacBrayne on 041-639 6626 after 6pm.

--oOo--

LINE 350800 ???

By Mike Todd

When Jim Butterfield (cries of "who's he?" from newcommers to the Commodore scene) arrived in the UK on a recent trip he kept muttering something about line number 350800. Try using this line number and see what you get. The results may not mean much to you, but it does demonstrate a bug in the BASIC interpreter.

It works equally on all PETs and VICs; even the APPLE (ugh !) has the problem but with a different line number (432640) If you haven't tried it by now I'll tell you what happens. Both PETs and the APPLE crash to the monitor although sometimes the whole machine hangs. On the VIC, the effect is the same as using the RESTORE key.

Why it happens is actually fairly simple and due to a bug I've had marked on my BASIC 2 disassembly for over 18 months, I've just never investigated the consquences.

In simple terms, the interpreter checks that the line number is less than 64000 and should enter the SYNTAX ERROR routine if not. Unfortunately it does this via a routine in which line numbers in the range 350720-353279 (432640-435199 on the APPLE) will force the interpreter into the middle of the ON..GOTO/GOSUB routine with the effect that memory gets corrupted and the processor crashes.

If you understand machine code, here's the explanation (based on BASIC 2, but the coding is identical on other BASICs - only the addresses differ).

At $C883 the line number is checked for an intermediate value greater than or equal to $1900 (6400 in decimal). If the loop were to continue this would become an illegal line number when multiplied by 10. The ERROR branch to $C7F0 is done via $C85B which branches if the accumulator is not $89. This is actually part of the ON..GOTO/GOSUB routine where, having already confirmed that ON.. is not followed by GOTO it goes on to confirm that it is followed by a GOSUB token ($89) - if not it generates a SYNTAX ERROR.

If the integer accumulated so far is in the range \$8900-\$89FF the accumulator will be \$89 and the branch is not taken, SYNTAX ERROR is not printed and the ON..GOTO/GOSUB routine continues in the middle. This results in the stack being corrupted (there is a PLA which corrupts the RTS address) and the machine will eventually crash.

Line numbers of the form 35072x to 35327x result in an intermediate value of \$8900-\$89FF and thus a line number in the range 350720-353279 will cause the crash. On the APPLE the range is \$A900-\$A9FF (since the APPLE's GOSUB token is \$A9) and the resulting line number range is 432640-435199.

The coding is responsible is as follows:

```
     $C7F0  JMP $CE03  ;SYNTAX ERROR
------------------------------------------
     $C853             ;ON..GOTO/GOSUB ROUTINE
     :
     $C85B  CMP #$89   ;check for GOSUB token
     $C85D  BNE $C7F0  ;if not
     :                 ;rest of ON..GOTO/GOSUB
     $C863  PLA
     $C864  JMP $C702
------------------------------------------
     $C873             ;READ LINE NUMBER ROUTINE
     :                 ;clear $11/12
     :
     $C879             ;RTS if current character not digit
     :
     $C87F  LDA $12    ;get high byte of integer so far
     :
     $C883  CMP #$19   ;is integer so far >= $1900 ?
     $C885  BCS $C85B  ;if it is
     :                 ;multiply by 10
     :                 ;add in current digit
     :                 ;get next character
     $C8AA  JMP $C879  ;and handle it
```

--oOo--

REVIEW

The PET Index
£ 12.50

Gower Publishing Co. Ltd.,
Gower House, Croft Rd.,
Aldershot, Hants.

If you are looking for something interesting to read, then forget it. But then a telephone directory doesn't make good reading (the plot is poor and there are too many characters). Over the years and from a limited set of magazines I have made notes of where to find certain items that may be of future interest. My list is limited to the articles that I have, and to whether at the time I thought the item worth noting. In retrospect it's a mess, since my interests and my computer equipment have changed in this time and the list never seems to contain references to the current requirement, but PET Index to the rescue. From a wide range of magazines, mostly UK, the author has systematically recorded each relevent item in a format which initially looks a little odd, until one realises that the data is also available on disk. The format is designed to enable fast retrieval by computer, based on the user's search criteria.

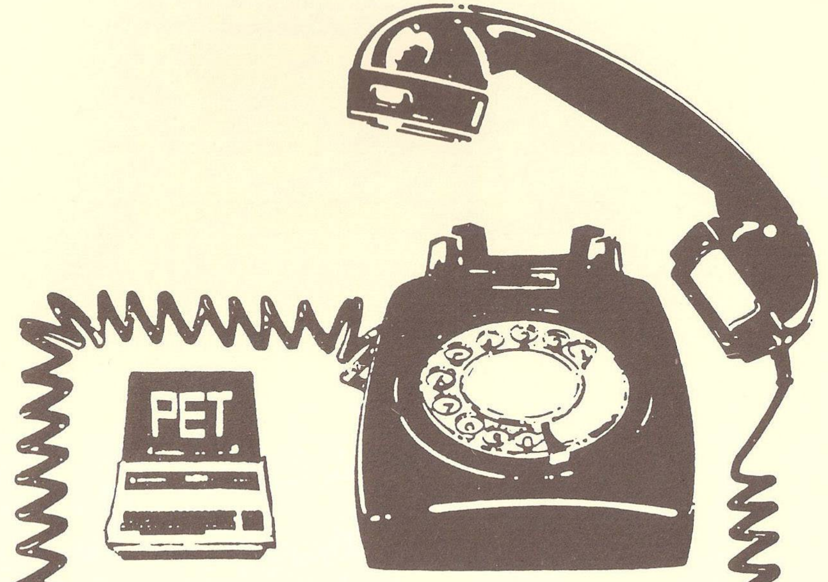
Since the PET has not caught on in the US to the extent that it has in Europe, the coverage in US & Canadian magazines is somewhat limited. Mike has wisely omitted references to their user group magazines, since these are somewhat difficult to obtain in the UK, and has restricted the reference material to Byte, Compute and Kilobaud Computing. I regard the omission of '6502/6809 Micro' as serious for over the years this magazine has contained some original material on subjects not covered in any other source.

One virtue of the Index is that subsequent references and corrections to that reference are also included, so that one does not have to suffer any inaccuracies that were later corrected on the original text.

For completeness I would have liked to have seen included a list of PET-related books and an indication of their coverage. Never-the-less for the information seeker this is a worthy addition to one's resources. Don't forget to get the disk of the book and let the computer do your searching, it will find so much more.

R.D.G.

--oOo--



CLEARSONS LTD.

Cash & Carry Computer
and
Word Processing Supplies

All types of computer stationery

Listing paper Ex-stock

Free quotation for your letterheads, invoices, statements etc.

Appointed dealer for

COMMODORE
micro computers

JUST LIFT THE PHONE

Farnborough, Hants

518022 & 518717

30 Camp Road, Farnborough, Hants.

Printed and distributed by Richardson Printing Ltd., Unit 23, Colville Road Works, Colville Road, Oulton Broad, Lowestoft, Suffolk NR33 9QS. *Telephone: (0502) 67029*