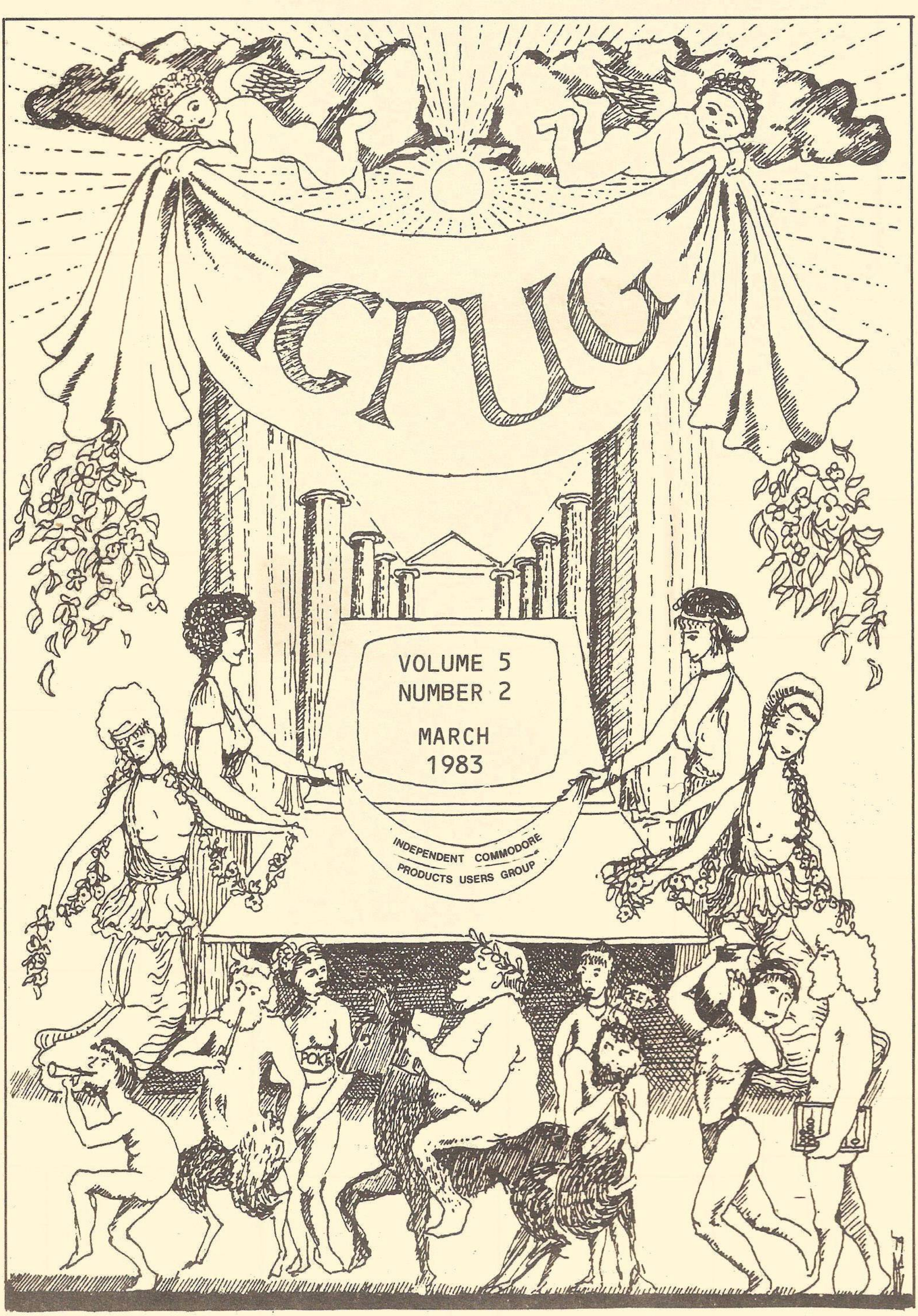

ICPUG
VOLUME 5
NUMBER 2
MARCH
1983
INDEPENDENT COMMODORE
PRODUCTS USERS GROUP
PEEK
POKE

HONORARY NATIONAL OFFICIALS

INDEPENDENT COMMODORE PRODUCTS USERS GROUP

VOL 5
No 2

Newsletter

March
1983

Europe's first independent magazine for PET users

EDITOR'S NOTEBOOK

Occasionally I hear, indirectly of murmurings that the degree of technicality in the Newsletter is way over people's heads. This may be so for some, yet for others it is this very expertise that is sought. This is specialist knowledge that one cannot expect to find in the 'popular' press, but only in a journal dedicated to one manufacturer and its product range (the 'unpopular' press?) My advice to those who are lost is to ask for more explanation if one needs it (don't be greedy and ask for a treatise on the floppy disk drives if you only have cassettes). Alternatively, save it for later when you may then understand it and find it useful.

The 'popular' press try to cover everything and anything, with knowledge spread very thinly, and not very accurately. They seem obsessed with a craze for keeping up with the latest models which one cannot buy and for which little or no software exists. In a classic example, an article on 'PET Books' omitted virtually all the good books on the PET, including material from the States and Ray West's reference work. The editor admitted that the person credited with the article was not responsible for selecting the books for review, neither did they bother - "we just bundled up the ones that publishers sent in". This example of comprehensive, well-researched and in-depth journalism might amuse some readers and perhaps instruct others.

This publication, as the name implies, is aimed at the user, never mind if you have the original Mk1 or the latest and greatest, we support it if it is Commodore. We even support it if it is not, but used in conjunction with Commodore equipment, e.g. Epson, CompuThink, Seikosha, etc. The editor, and other selected members, receive Commodore dealer information and relevant information is passed on by way of the Newsletter. Some information is with-held from publication for various reasons. This includes copyright, e.g. circuit diagrams, what I would consider to be 'Company Confidential', such as trade discounts and prices, and certain marketing information, and occasionally 'limited interest' matters such as some obscure aspect of Fortran on the 9000 Super-PET. What we do reproduce are all the bugs, quirks, anomolies and technical peculiarities that make pitfalls for the unwary.

Several regional groups warrant their own local newsletters. The more established regions already send me a copy. Newer groups should send me a copy for information and since only a few regional secretaries send me details of their group's activities, passed or proposed, I have to glean snippets from these newsletters. This helps to give newer groups ideas for subjects for meetings from the more established ones. Articles with general appeal should, of course, be sent to me for all to read.

The editor regrets that he is no longer able to accept articles on 8050 formatted disks (or 8250 for that matter).

This issue has two independent reviews of the same product, Masterchip, and I make no apology for including two members' opinions of the same device.

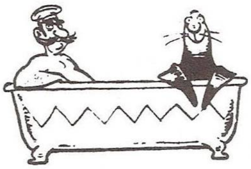
R.D.G.

--oOo--

CLEARSON LTD. EXPAND

As well as their Farnborough (Hants) shop at 30, Camp Road, Clearsons are opening a new shop at 9, Ye Market, Selsdon Road, South Croydon, CR2 6PW. Tel: 01-686 0046. Staff at this new shop will include Roy Wolland and Paul Sargeant. A further shop will open soon in Reading, details next issue. Members will continue to be allowed appropriate discounts when they identify themselves with a valid membership card.

J.B.



--oOo--

THE COMMODORE 64 - FIRST IMPRESSIONS

By Brian Grainger

I was fortunate (?) in getting one of the first batch of PAL versions of the Commodore 64 in this country. I have therefore had a good play, as is evident by the number of articles for the machine in this Newsletter. The purpose of this article is to give my first impressions of the machine.

On opening the box the first thing to go to is the manual. Well, we know what Commodore manuals are like (for ICPUG newcomers, they are awful). This is no exception. I had an American manual but I believe Commodore have arranged a European equivalent to be with the latest batches. This also corrects some of the more obvious errors in the original US manual. It would appear from the US manual that in the States machines are supplied with an aerial switch box which allows both aerial and computer to be connected to the TV at once. Not so in the UK which is a pity, it would be a nice touch. Connecting up was easy and one notices that the RF lead to the TV is incredibly long. Commodore do rectify some mistakes of the Vic it would seem. The separate power supply is not very aesthetic and a nuisance when having to carry your computer to the local ICPUG meeting!

Having connected up you switch on and it works. Having been impressed by this one is immediately driven to despair by the choice of start-up colours. Background is dark blue, foreground light blue. Readability is terrible. Would Commodore please fire the man who devised this! I can only presume the design was carried out on a monochrome TV. It looks OK on that!

The 64 is very similar to a VIC and has some faults the same. The only trouble is that a lot of PET people will upgrade(?) to a 64 and complain more readily. I play around with machine code. No machine language monitor in ROM on the Vic/64. A version of Extramon exists but whether it is freely available is unclear and anyway it is sometimes convenient to have MLM in ROM. The BASIC is essentially

BASIC2+ with good old garbage collection problems. Not nice for an experienced PETter. Because it is BASIC2 it suffers an amusing problem in that typing ?FRE(0) results in a negative number! The reason is that the result is stored in 2 bytes and the maximum number stored in 2 bytes is 32767. This was fine when no more memory was available. Times have changed but the BASIC hasn't. The correction factor is +65535.

The next irritating feature is noticed when saving a program to cassette. When reading or writing to tape the 64 screen goes blank! Don't worry, the messages get displayed but it leaves me feeling very uneasy when using the cassette. Using a 1540 disk drive suffers in the same way as one has to blank the screen manually before accessing the disk. The only solution here is to upgrade to a 1541.

In using different colours I sometimes found that there was interference between adjoining colours. This was particularly noticable in Hi-res mode.

On the software side apart from the FRE(0) anomaly I have only noticed one other peculiarity. By interrupting a LIST (press the STOP key) at the bottom of the screen the cursor disappears!

My last grumble is that from the manual one is told not to insert or remove cartridges with the 64 switched on. I believe this is also true of the Vic-20 and I do not like it. Machines should be idiot proof and an idiot will take a cartidge out at any time.

Well. I've had plenty to say against the 64. Is there any reason to buy one. The answer is yes, many reasons. The facilities of the machine are incredible, albeit a little difficult to use sometimes. For an old PET man the auto repeat keys and restore keys are nice. The sound facilities (SID chip) and visual facilities (VIC chip) are far superior to anything else except perhaps the BBC. Then there are 2 features unique to Commodore. The screen editor (why do other machines have such daft editors?) and on the 64, the 64K of RAM some of which co-resides with the ROM.

The big problem with the BBC and its fantastic graphics is that to use the graphics eats memory and leaves little for programs. On the 64 it eats memory as well but by using the RAM under the ROM, no space is lost to BASIC. The graphics are not particularly easy to use, lots of POKEs, but I am in no doubt that software will appear to make it simple. The ability to soft load packages is a big advance on early micros and as soon as an IEEE slot has been sorted out I'm sure BASIC4 will appear to appease all us PET people.

I have made no mention of the sprite facility. That is because I am not devoted to games which is where I believe this facility is useful. It works and for once is relatively easy to use.

Summing up the machine has its faults but nevertheless is an excellent machine with advanced features. For people used to the PET, they are not going to be completely happy until the IEEE cartridge and BASIC4 are available. The manual is attrocious but the Programmers Reference should solve that problem. The price is acceptable, though not budget levels. It is interesting to note however that at current levels of the pound the 64 is cheaper in the UK than the US !

I finish with a note to those who might have got their 64 around the same time as I did (December 1982). I do not know how you managed it because after sending the odd 1 or two from the initial batch to dealers etc., presumably for demo purposes. Commodore UK refused to accept any more machines from Germany until certain anomolies were resolved. It is nice to know Commodore have the consumers interests at heart sometimes. to the extent of holding up the launch of what should be an extremely successful machine.

--oOo--

MASTERCHIP - TWO REVIEWS

Masterchip is yet another 4K chip which provides an extension to Commodore BASIC on a PET. 8 new editing commands, 43 BASIC statements, 6 direct commands and single key BASIC is included in its repertoire. The ubiquitous auto repeat key is also there.

Extended screen editor.

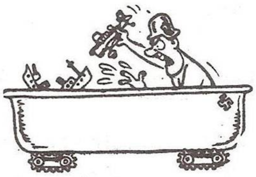
By the use of the 'RUN/STOP' key as a control key the following are provided:

Key	Description
'INST/DEL'	delete screen line
'SHIFT' & 'INST/DEL'	insert screen line
'CURSOR UP'	scroll screen up
'CURSOR DOWN'	scroll screen down
'SHIFT' & 'B'	remove program line up to cursor
'SHIFT' & 'E'	remove program line after cursor
'(QUOTES)'	escape from quote mode
'SHIFT' & 'G'	change character set

Single Key BASIC.

The same control key is used to obtain single key BASIC statements. For example holding down 'RUN/STOP' and 'G' gives GOTO (obvious) or 'K' to give RIGHT$(.. (not so obvious). Altogether 15 keys are memorable because they match by first letter - though 11 do not! New BASIC statements. Area Manipulation. The following can be used direct or in a program.

#AF	Area fill (character in C)
#AR	Area field reverse
#AN	Area field normal
#AI	Area field invert
#AS	Area case shift
#AU	Area case unshift
#AC	Area case invert.



The area manipulation commands affect an area of screen defined by the variables X,Y,A,B. X and Y are the co-ordinates of the bottom left and A and B are its horizontal and vertical dimensions.

Screen Manipulation.

These are the same as above but use the syntax #SF, #SR, etc., rather than #AF. In addition the following movements can be accomplished.

#US	Shift screen up
#DS	Shift screen down
#LS	Shift screen left
#RS	Shift screen right

Similarly, UR, DR, LR, RR can be used to roll the screen.

Double Density Plotting.

#WP	Write point
#EP	Erase point
#FP	Flip point
#TP	Test point

The above points are predefined by the current values of varibles X and Y. Where X lies between 0 & 79 and Y between 0 & 49.

Double Density Bar Plotting.

#WX	Write bar 0,Y to X,Y
#EX	Erase bar 0,Y to X,Y
#FX	Flip bar 0,Y to X,Y
#WY	Write bar X,0 to X,Y
#EY	Erase bar X,0 to X,Y
#FY	Flip bar X,0 to X,Y

Fine Resolution Plotting.

#WF	Write fine 'Y'
#EF	Erase fine 'Y'
#FW	Write fine 'X'
#FE	Erase fine 'X'

These fine resolution bars give the user a range of 320 x 200 in the horizontal OR vertical directions.

Move Screen Routines.

#MS	Move memory to screen (location in X)
#SM	Move screen to memory (location in X)
#SW	Swap memory and screen (location in X)
#CS	Copy screen to printer

Graphics Toggle: #GT will change toggle upper/lower case. Poke Character. #PC puts the characters with the peek/poke value of the variable C at a point on thre screen defined by the variables X and Y. Print At: #PA places the cursor at a position Y lines from the top of the screen and X places across.

New Direct Commands.

SHRINK
Removes all wasted REMs and spaces. It tells you how many bytes it has saved.
DUMP - lists all variables though not arrays.
DISPLAY - displays on the screen the list six lines executed.
HEX n - converts decimal to hex
DEC n - converts hex to decimal
KILL - not strictly a command but switches MASTERCHIP off

Conclusions:

At £ 30.00 MASTERCHIP provides good value for money for those requiring enhanced graphics for the PET and don't want to go for the the real high resolution plotting as offered on the SUPERSOFT and MTU boards.

The manual is fairly well written with several example programs which show off the various graphics commands as well as describing the other features such as the direct BASIC commands (quite useful) and one One- key BASIC (slightly less useful - see text).

The chip can be ordered from: Hodkin Software, Greenfields, Hammerwood, Sussex. Please specify which socket the chip is required for, as well as machine type - e.g. BASIC 2 or 4.

By K.Viney.

* * * * *

MASTERCHIP is a 4K graphics EPROM, costing £ 30 plus V.A.T., from Hodkin Software, at Greenfields, Hammerwood, Sussex, RH19 3QE. It is supplied in versions that fit into UD3, UD4 or UD5 sockets on 3000-, 4000- and 8000-series PETS. Hodkin claims that it adds 43 new BASIC statements to Microsoft BASIC.

The fitting instructions are adequate but not as comprehensive as those which came with my Toolkit and I would find it difficult to obey their admonition not to touch the pins AND to comply with their request to bend the pins to fit if they are splayed. The operating manual is a neat, 4" by 6" eight-paged booklet printed in blue, thus making it difficult to photocopy a working sheet. A loose example program sheet was extremely difficult to read but when finally deciphered did produce impressive screen effects.

My version is for UD3 ($9000) and is activated by SYS 39074, which does not point to the start or end of the EPROM, nor is it a particularly easy value to remember. Unfortunately, it is incompatible with Toolkit and Arrow. It fights with DOS and after the use of disks or cassette it writes SYS 36904 onto the screen. The instruction booklet does not mention this. The REPEAT KEY function is turned on by holding down any key for a few seconds and in my version, the period required to turn it on seems to vary. Once the REPEAT is turned on, I find it rather too fierce, although the REPEAT CURSOR is most useful. With REPEAT turned on, the control-key-plus-single-key BASIC entry is almost uncontrollable and it requires a lot of practice to produce one command rather than many! There is a need for a STOP-REPEAT to be put into this facility. The keys for the commands are not as easily remembered as, for example, in EDEX, where the functions are readily associated with the keys after very short usage.

Having exposed the cons we can now move to the better aspects! The SCREEN TO PRINTER works very well with the MX80 and sends CR and LF but also dumps the screen print command onto the printout. Some utilities I have used avoid this by using a two key press that is not echoed on the screen in command mode. As it stands, it is that much less useful for designing and printing logos, etc., directly from the screen. The screen handling functions set is very easy to use and quick acting. Areas of the screen, as well as whole screens, can be designated and filled with characters and the area reversed and shifted. There is a very useful and easy to use set of double-density plotting functions, which will plot 'quarter' blocks and bars and a fine-resolution facility, which uses the PET character set to its full but, understandably, does not achieve the fine screen dots that are possible on the Apple or the BBC [or the Vic-20, etc - Ed]. If you are interested in games and super, dazzling, fastmoving graphics then this is certainly a worthwhile chip to get, although your software will have limited portability.

DISPLAY enables a program to be traced at controlled speeds. DUMP shows the values of variables at the time of program termination and SHRINK removes all REMs and spaces and will tell you how many bytes you have thus saved. HEX converts a decimal number into its hex equivalent and DEC does the reverse. KILL switches the chip off.

Before passing over negotiable, crisp, crackling currency for your copy of this chip, I recommend that you get a copy of SUPERBASIC from the ICPUG Software Library to see whether or not this will gratify your craving for all-singing-all-dancing screen-writing acrobatics. Although I use my chip for drawing graphs and charts onto the screen I am not at all sure that it is such a 'dramatic transformation of my PET's capabilities'.

Hodkin have a wordprocessor chip, called 'WORDHELF][' . I would like to hear from anyone who has used this chip. Maybe it will save me from my mounting anxiety as my 4040 drives chatter alarmingly and almost fail on loading that excellent, but perhaps 'over copy-protected', 'SUPERSCRIPT', upon which this was written. Or maybe the writer of 'SUPERSCRIPT' will put it into ROM?

W.G.C.Austin.

--oOo--

THE VIC COLUMN

By Mike Todd

I start this time with a word of warning to anyone who is attempting to build their own modules for plugging in to the Vic's expansion slot.

Provided that you remember that the diagram and connections shown in both the Vic Revealed and the Programmer's Reference Guide are looking "in" to the expansion socket, and you ignore the normal numbering convention on the edge connector, then there'll be no problems.

What appears to have happened is that Commodore have numbered their socket as if it were an edge connector so that the numbering is reversed left to right. In other words, Commodore's pin 1 is normally referred to as pin 22, and pin A is normally pin Z. In fact, if you look inside the Vic, the numbering on the socket does not agree with the printing on the circuit board!

I repeat that, provided you stick to Commodore's convention and ignore the normal industry convention then no harm will come to your board. But if you wire up a board with its edge connector as industry standard, then you could be heading for trouble.

My thanks to Bernard Halliwell of Rochdale for pointing this one out, and also for providing some other bits and pieces for this Vic column.

GRAPHIC 4

I mentioned last time the existence of an additional GRAPHIC command within the Super-expander.

I've had a chance to investigate this and the Super-expander a bit further over the past few weeks, and this is what I found.

The Super-expander always works with $1000-$1DFF (4096-7679) holding the character generator used to provide the high-resolution graphics facility and $1E00-$1FFF (7680-8191) holding the actual screen RAM.

With only 3K RAM expansion, this configuration causes no problems, since the screen is where it was anyway and the character RAM is at the upper limit of BASIC RAM.

However, with 8K or more of expansion, the screen RAM would be at $1000-$11FF (4096-4607) and the BASIC program would start at $1200 (4608), which is all very well until you want to use high-res graphics.

If this is the set-up, which it would be on switching on, the minute you first decide to access GRAPHIC 1, 2 or 3, the Super-expander must move your BASIC program up, out of the way and move the screen and character memory around.

If it has to do this, it moves the BASIC program up to start at $2000 (8192) and performs a CLR. This gives the Super-expander all the room it needs.

If, however, the program already starts at $2000 (8192) then the Super-expander leaves the program where it is, and doesn't perform a CLR.

This all sounds a bit complicated - so, if you've got a Super-expander and more than 8K of RAM expansion, reset your Vic (switch off and on) and then type in the following program:

```
10 A=5
20 GRAPHIC 1
30 GRAPHIC 0
40 PRINT A
```

TODD'S LORE IS BEYOND THE FRINGE

Daily Mirror Caption

The program will be occupying memory locations from $1200 (4608) onwards and line 10 will set A=5 in the normal way; line 20 then sets up the graphics facility, but will also move the BASIC program out of the way as already mentioned. This will have the effect of CLR, and A will be reset to zero.

GRAPHIC 0 simply restores the screen to text mode and the value of A is then printed out. Its value should be zero when you first RUN the program.

Now, if you RUN it a second time, the program won't be moved because it's already there, CLR is not invoked, the value of A is not affected and so line 40 will print out the correct value.

This has serious implications if you want to run a graphic-based program and switch to the graphics screen only after doing some computations. The first time this program is LOADed and RUN, the variables will be wiped out at the first GRAPHIC command.

The answer is to have the very first line in the program

```
1 GRAPHIC 1 : GRAPHIC 0
```

This will force the move and a CLR and get it out the way. From now on, any GRAPHIC command can be called with impunity.

OK - so what has this got to do with GRAPHIC 4?

GRAPHIC 4 reverses the process and puts the program back to $1200 (4608) and restores the screen to $1000 (4096) and it performs a CLR. There is also a bug which produces a "SYNTAX ERROR" when it is called - I'm not sure why. I also have doubts as to the usefulness of the command, but it exists and could be useful to someone, somewhere.

"FIND" AND THE PROGRAMMERS AID

There are a couple of problems (not too serious mind you) with the Programmers Aid cartridge.

First, there's a bug which crashes the Programmers Aid when asked to FIND or CHANGE Super-expander keywords (such as GRAPHIC, RJOY etc). This can't be recovered by pressing STOP+RESTORE, only by switching off and on again.

I'm not sure what the bug is, but all seems to be well if you don't use the FIND or CHANGE facility.

STEALING BYTES

The second problem occurs when you initialise the Programmers Aid (SYS 28681). It grabs the top 120 bytes of memory to store its function key definitions.

If you then re-initialise (maybe you used STOP+RESTORE to get out of trouble, thereby disconnecting the Aid), it grabs another 120 bytes.

If you're working with an unexpanded Vic and repeat this, it won't be long before the Aid grabs all the available memory and gives an "OUT OF MEMORY ERROR".

LOADING VIC PROGRAMS ON THE PET

There are some members using a PET to edit Vic programs because of the larger screen; there are those who want to convert Vic programs for use on the PET and many are having problems LOADing programs SAVEd on the Vic into the PET.

When you SAVE a program on the Vic, the memory address at which the program starts is written onto the cassette or disk before the program itself.

Because of the different configurations of the Vic, there are four possible start addresses - $0401 (1025) for a Vic with only 3K expansion; $1001 (4097) for an unexpanded Vic; $1201 (4609) for a Vic with 8K or more expansion; or $2001 (8193) for a Vic with more than 8K expansion and which has recently used a GRAPHIC mode with the Super-expander. They can LOAD the program OK but can't get at it!

Normally, these different locations don't matter on a Vic since the program is normally loaded back where the Vic is expecting to see it, which is not necessarily where it originally came from!

However, a PET always loads a program back at exactly the same address that it came from; but it still expects to see the program at a specific address - $0401 (1025). Therefore, only Vic programs SAVEd from this address (ie SAVEd on a Vic with only 3K expansion) will be accessible.

There are two solutions to the problem. One is to load the program and then move it back to $0401 and proceed from there; the other is to change the PET's pointers to look at the program, wherever it may be.

Both methods have their devotees, and, apart from one technique which involves the use of the Toolkit APPEND command, both require a knowledge of where the program has ended up after the LOAD.

If you're using cassettes, life is very easy since PEEK(252)*256+PEEK(251) will indicate the start address, although in practice, only the most significant byte is required and this is given by PEEK(252).

If you're using disks on the PET, you'll actually have to know the "high byte" of this address. In the four conditions mentioned earlier, this would be 4, 16, 18 or 32 and in the examples that follow these values should be used to replace PEEK(252) when using disks.

The first technique is used when you aren't bothered about the program not occupying the same addresses as it did on the Vic, which in practice is most of the time.

First, clear out any programs already in the PET (NEW will do) and enter a single line:

```
0 REM
```

LOAD your program in the normal way and then LIST it. If you get a complete listing of the program there's no problem, the program has loaded to the correct position straight off; but if you just get your line 0 again, type the following, remembering to substitute the appropriate number in place of PEEK(252) if you're using disks.

```
POKE 1025,1: POKE 1026,PEEK(252)
```

LIST the program again and you should now have line 0 followed by the Vic program. Delete line 0 in the normal way, and the complete Vic program is now ready for you.

The second method is a little more complicated, and would only be used if you really did want to keep the Vic program sitting at the same address.

As before, clear any program out of the PET and LOAD the program as normal, then type

```
POKE 41,PEEK(252): POKE 42,PEEK(44): POKE 43,PEEK(45)
POKE PEEK(252)*256,0 : CLR
```

The final POKE is necessary because the very first byte of the program must be a zero, and this is not saved as part of the program. The POKEs to 42/43 are also necessary to make sure variables and arrays don't overwrite the BASIC program.

I hope this solves the problems that many members have been having - it doesn't, however, solve the other major compatibility problems which occur when trying to get PET programs to run on the Vic and that's the problem of converting some of the rather obscure PEEKs and POKEs used on the PET.

Anything under 1024 (the BASIC RAM workspace) and between 32768-33767 (the PET screen) are fairly easy, but accesses to the PET's interface chips can cause severe problems. I'll try to look at the problem and come up with some possible solutions.

THE MACHINE CODE MONITOR

This has a couple of oddities to it. I've already mentioned the need to POKE 43,1 when returning to BASIC in order to get your BASIC program back, but this doesn't seem to solve all the problems as the monitor corrupts several locations in zero page (as it at least admits to doing!).

Bernard points out that if virtual page zero is enabled using the E command, and then attempt to change the registers using the R command and overtyping new values on the screen, the new values of A, X and Y are ignored by the program.

He also points out an error in the accompanying booklet in section 2.3.4 on the "E" command. The kernal is said to use zero page and "some of the $0200-$0800 pages". In fact, the kernal uses most of $0200-$0400.

THE SUPER EXPANDER AGAIN

Whilst on the subject of memory usage, the Super-expander uses a variety of locations between $0000-$0400. It uses from $62-$6D when they're not being used by BASIC and $9B-$9F, $C3-$C4 and $FB-$FC as well.

It changes the CHKSHF vector at $28F-$290 as well the BASIC interpreter vectors except for $302-$303 (WNDMN). It also changes the IRQ and BRK vectors at $314-$317. The only other vectors changed are the WNDCHI and WNDCHO vectors at $324-$327.

There are several unused locations in this area, and the Super-expander uses those from $2A1 to $2D9 as well as much of the cassette buffer.

From this last fact it would appear that cassette files can't be used within a program using the Super-expander's graphics commands without extreme caution and a really good understanding of both the Super-expander's quirks and the way the cassette routines operate.

THE VIC-20 TOP-20

With 110,000 Vics sold into the home computing market last year in the UK alone, Commodore have released details of their top twenty best selling Vic-20 programs.

1 - INTRODUCTION TO BASIC I - £14.95 (cass+book)
2 - INTRODUCTION TO BASIC II - £14.95 (cass+book)

These have been the best selling items in December and January and are both programming courses written by Professor Andrew Collin of Strathclyde University. Part I alone has sold over 100,000 worldwide.

3 - HOPPIT - £4.99 (cass)
4 - BLITZ - £4.99 (cass)

Next come two very popular cassette games which I'm sure many of you will have seen, if you don't already have them.

HOPPIT is a game of reactions in which you have to guide a frog across the road and is a fair imitation (considering the price) of the arcade game.

Written by 18 year old Darryl Mattocks during Summer holidays, it just shows you how writing such a simple but effective program can earn someone in excess of 200 a month!

You don't need to be "commissioned" to write a popular program. Although not an orignal program by any means, Simon Taylor (16) wrote the Vic version of BLITZ and sent it to Commodore in the hope that they could use it!

5 - SARGON II CHESS - £24.95 (cart)
6 - ADVENTURELAND - £24.95 (cart)
7 - GORF - £24.95 (cart)

All three of these are ROM cartridges, and they must be "extra" popular to attract buyers at the (at least in my opinion) rather high price of £24.95

SARGON II CHESS is supposed to be one of the best chess games around for micro-computers. The original algorithm was written by Dan & Kathe Spracklen and published by Hayden in 1979 and this cartridge is the Vic-20 version.

It certainly gives a good run for the money, although BOSS, from Audiogenic is reputed to beat SARGON II and plays a much better game. It's cheaper but it does need 8K expansion because it is loaded from cassette.

ADVENTURELAND is the first of the series of the Scott-Adams (often referred to as "Mr Adventure" in the USA) adventure games. They are all very addictive, but once solved lose a lot of their attraction.

That's a good point to remind you of the ICPUG Adventure-Swap scheme run by Brian Roberts mentioned in the January Newsletter.

GORF is several arcade games in one (just like the real arcade version of GORF, but I gather slightly trimmed). I've not tried it so can't comment, but judging by its position it can't be too bad.

8 - ENGINE SHED - £4.99 (cass)

This is a junior maths teaching program aimed at 7-11 year olds and comes on cassette.

9 - OMEGA RACE - £24.95 (cart)

Commodore's version of the arcade game of the same name is reported to be one of the best Vic implementations of an arcade game around.

10 - MISSION IMPOSSIBLE - £24.95 (cart)

Another of the Adventure series, more difficult than some of the others, but another excellent game.

Well, that's the top-ten and, in true Radio One style, here's a quick run-down on numbers eleven to twenty.

11	STRATEGIC ADVANCE	£4.99 - (cass)	board game of strategy needs +16K
12	APPLETREE & BIRDS	£4.99 - (cass)	Junior Maths (7-11)
13	AVENGER	£19.95 - (cart)	fair version of the infamous Invaders
14	STAR BATTLE	£19.95 - (cart)	another arcade game almost like real thing
15	VIC MONEY MANAGER	£9.99 - (cass)	personal finance & cash flow - needs +8K
16	JELLY MONSTERS	£19.99 - (cart)	same as PACMAN but they daren't admit it!
17	VIC ROAD USER	£9.99 - (cass)	tests knowledge of the highway code - needs +8K
18	GORTEK & THE MICROCHIPS	£12.99 - (cass etc)	teach programming to young people (see below)
19	MASTERMIND	£9.99 - (cass)	general knowledge quiz extra questions at £1.99 per cassette - needs +8K
20	ROBERT CARRIER'S MENU PLANNER	£9.99 - (cass)	has recipes & meal plans needs +8K

Well that's the top-twenty. I'd like to mention the "GORTEK AND THE MICROCHIPS" which is more than just a way of teaching computer programming.

It's a unique concept in teaching young people computer programming aimed at 10-13 year olds, but younger children can use it with parental guidance).

The planet Syntax is being invaded by the fearsome Zitrons! Gortek is furiously teaching the microchips to program the great computer for the attack.

The package was written by three British primary school teachers who combined to write a short story to teach computer awareness to children. Their final result is a very clever and innovative approach to computer education and is the first of a series of six being prepared by the team.

It comes complete with a colourfully illustrated, 44-page book containing the story book and the Microchips Training Manual that will teach the child the fundamentals of BASIC programming. There are two cassettes containing 12 programs and even a Gortek badge!

I might add that this package was the first software to be released by Commodore for the 64, and should shortly be available for the 4000 series.

WATCH THOSE WIRES

Well, that's enough free plugs for Commodore - now to a plug problem associated with the use of the Vic modulator. Many dealers are having Vics returned because of faulty modulators, although the problem is rarely the modulator but the plug and cable connecting the modulator to the Vic.

With the fairly heavy flexing that this cable gets, especially as people tend to wrap the cable very tightly around the modulator for storage, the wires in the cable and in the DIN plug frequently break.

It is possible to re-solder the plug, but if wires break inside the cable then things get a bit trickier. So, take care of that cable (and indeed the modulator to TV cable) as repair or replacement could be difficult or expensive.

--oOo--

MX80 LINEFEED SWITCH SOLUTION

By Barry Biddles.

The solution to the MX80 auto-linefeed problem (p76 Jan issue) is simply a matter of bringing out the line feed switch from SW2 to a small toggle switch of your own mounted in the removeable panel at the back of the printer. The two connections to the circuit are not made by soldering directly to the switch, which would invalidate the guarantee, but by using miniature probes clipped to the legs of any components on the board which are already connected to the desired points. In this way, the printer can rapidly be restored to "standard" should it be necessary to invoke the guarantee. The panel may either be left off, or you could borrow an unmodified one from a friend.

I have taken the board out in order to trace the printed wiring underneath, and have discovered the most suitable points for connection. One wire should go to the BACK END of DIODE ZD4, which is situated to the right of the big chip near SW1. The other should go to PIN 14 of the CENTRONICS SOCKET. Access to this is gained by removing the small PCB. PIN 19 may be used as an alternative to the diode connection. Suitable probes can be obtained from RS COMPONENTS under the title "miniature probe". The switch should of course be left in the "open" position.

The switch on SW1 should be set to give line feed on buffer full if you are using PLUSDOS, in order to be fully compatible with the &LIST facility.

Regarding the pounds switch, the same method is likely to be suitable, but I had no reason to discover the connection as I do not need this facility myself.

--oOo--

COMAL CORNER

By Brian Grainger

Well, I'm into my second year of COMAL articles and people are still writing to me about COMAL, so I think the interest is there and the language is here to stay.

News this month centres on Version 2.0. I have now seen Borge Christensen's notes on this version. It is apparently written for the Commodore 64, but as Commodore are advertising the 500 with COMAL as a soft loaded language I wonder whether it is also intended for this machine. Latest news is that COMAL 2.0 for the 64 is not likely to be around for a while and as yet the cost is unknown. I'm not sure I can wait that long so I'm looking intently at existing COMAL versions with a view to conversion.

The good news on version 2.0 is that all programs written in 1.02 or 0.12 COMAL should run in it. Upward compatibility has been achieved. One new command of 2.0 is apparently in 1.02 as well so apologies to all I have confused with my description of ENTER.

The ENTER command in 1.02 and 2.0 will enter program lines from a disk file into the program memory. However unlike 0.12 it will erase the existing program in memory. Thus ENTER cannot be used to merge program lines from a utility library. Its sole use appears to be for transfer of programs between one COMAL version and another. What has been added to 1.02 and 2.0 is a MERGE command which is more powerful than the original ENTER. Its syntax is as follows:

```
MERGE [<line no.>] [,<increment>] <file name>
```

This will merge a LISTed program from the disk into the existing program starting at the given line number and incrementing the line numbers by the value of increment. If line no. is omitted from the MERGE command a value of increment+greatest line number in current program will be used (thus the disk program will be appended to that in memory). The default value of increment is 10. The line numbers of the program on disk are not used and are irrelevant. This new command makes the merging of existing utilities into new programs considerably easier.

The second new facility of version 2.0 COMAL is in the introduction of string FUNCtions in addition to the normal floating point and integer functions.

The third improvement on 2.0 is the ability to store PROCs or FUNCtions on disk which are called up into memory as required. Instead of defining the PROC in the program one stores the PROC as a program on disk (it is not clear from the notes if it should be LISTed or SAVEd!) All one does in the program is to define a PROC as EXTERNAL. An example follows.

```
PROC EXAMPLE(IN,REF OUT) EXTERNAL "0:EX1",9
```

This says that the procedure EXAMPLE with parameters IN and OUT is stored on disk unit 9, drive 0, with a filename of EX1. A call to the procedure is as normal and during execution when a call is made the procedure will be loaded into the workspace from disk and executed. On completion of the procedure the workspace will be freed again. In this manner quite large programs can be executed by storing much of the program on disk and calling up as required. The penalty is increased run time while waiting for disk files to be loaded.

As readers who tried to obtain Len Lindsay's COMAL Handbook, after the review last time. are aware- it did not appear in the UK when I said it would. There were some last minute hitches. I actually have a copy from the States (I write this in Feb) and deliveries in the UK are expected in March/April.

A point that I have not mentioned previously but has been brought to my attention. On version 0.12 or 0.11 LABEL names like PROC names are GLOBAL even when used in CLOSED PROCs. Everything is LOCAL in CLOSED procedures on the full COMAL 1.02 (or 2.0). Moral - take care with label names!

To finish this time I enclose a little procedure that will SAVE COMAL programs on tape under ARROW format. You have to have ARROW fitted of course. First the procedure from Nick Higham:

```
 9020 PROC SAVE' CLOSED
 9030  DIM F$ OF 15
 9040  INPUT "<clr,rvs>FILENAME<off> ": F$
 9050  FOR I:=1 TO 15 DO POKE 671+I,32
 9060  FOR I:=1 TO LEN(F$) DO POKE 671+I,ORD(F$(I))
 9070  PRINT "<dn,rvs>POINTERS<off>:<dn>";PEEK(134);
       PEEK(135);PEEK(21);PEEK(22);
 9080  PRINT PEEK(81);PEEK(82)
 9090  POKE 688,PEEK(23)
 9100  POKE 689,PEEK(24)
 9110  POKE 690,PEEK(81)
 9120  POKE 691,PEEK(82)
 9130  SYS (49134)
 9140 ENDPROC SAVE'
```

<u>(c) 1983 ICPUG</u>

To save a program to tape in Arrow format your program should contain the above procedure and its first two lines should be:

```
          1 EXEC SAVE'
          2 END
```

Now type RUN and the program will be saved and six numbers displayed. Make a note of the numbers.

To reload, type NEW then POKE the following memory locations with the numbers noted IN THE SAME ORDER-134,135,21,22,81,82. The last two values form an address-lo and address-hi byte. Add 5 bytes to the value and store the new lo in 112 and the new hi in 113. The program is loaded by SYS(49140).

It may seem laborious doing 8 POKEs before loading a program but it is much faster than using a standard cassette save. Bear in mind that the SYS values above will be dependent on where your ARROW chip is located.

--oOo--

DRIVING STEPPER MOTORS

By Tom Mead.

A recent query about driving stepper motors in the problems page, prompted several members to offer their suggestions. The most practical advice came from Tom Mead of Sunderland and the following article is based on his notes.

MRT

Most stepper motors are driven from 12v DC and, because of the way they work, they require the generation of a 4-phase drive sequence.

The motor has several coils arranged in four sets (Fig 1) wound on a stator and a permanent magnet rotor which is "pulled" round by the stator coils being driven in sequence.

In general use, connections 2 and 5 are connected to a common +ve supply line and the result is shown in Fig 2. Underneath the stator coils is shown the necessary switching sequence for driving the motor forwards and backwards. Note that, to reverse direction, it is only necessary to invert the pulses supplied to C/D.

The phased sequence can be readily presented to the user port lines in BASIC or (with suitable delays to prevent the pulses going too fast) in machine code.

The simplest interface is a direct drive using a Darlington driver chip with its own input resistors such as ULN 2003 (16 pin - 7 drivers) or ULN 2803A (18 pin - 8 drivers) and the circuit diagram of a single stage in shown in Fig 3. Note that all components are on the chip and Fig 4 shows how the 2803 may be connected, although a heat sink should be used if all eight drivers are used.

To drive the motor, the relevant patterns should be output from the port at a maximum frequency of about 100Hz which will limit the speed to about 2 revs per second

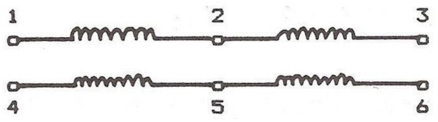


FIG 1
stator coils

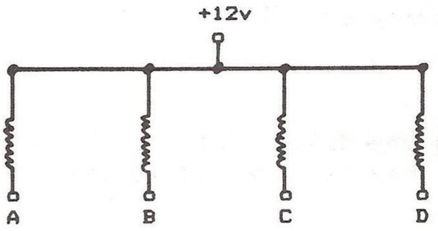


FIG 2
stator coils
normal connections

	A	B	C	D	
phase 1	1	0	1	0	FORWARDS
phase 2	0	1	1	0	
phase 3	0	1	0	1	
phase 4	1	0	0	1	
phase 1	1	0	1	0	
phase 1	1	0	0	1	BACKWARDS
phase 2	0	1	0	1	
phase 3	0	1	1	0	
phase 4	1	0	1	0	
phase 1	1	0	0	1	

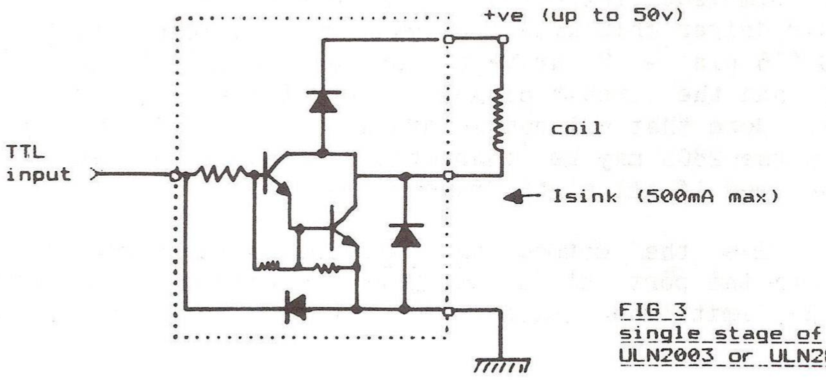


FIG 3
single stage of
ULN2003 or ULN2803

although higher rates can be used if opposite coils are driven in anti-phase (rather than all four at +12v, drive two at +12v and the opposite two at -12v). However, it is usually cheaper and simpler to buy a better quality motor if higher rates are required.

An alternative interface uses a 40109 level shifting buffer with a VQ1000 quad FET (Fig 5). Although there is no obvious advantage in using this method of interfacing, it is a popular configuration.

If you want to drive a single stepper, the software is reasonably simple, but driving more than one (as is usually the case) requires quite complex programming and needs to be done in machine code.

An alternative is to generate the necessary phasing in hardware, using only two lines (direction and clock) from the computer. One simple way is shown in Fig 6 using 74114 J-K bistables or the 74114 can be replaced by a 7474 D type wired as shown in the figure.

The exclusive-OR gates act as programmable inverters, with DIR low they simply buffer, with DIR high they invert and hence will reverse rotation.

The SAA 1027 chip contains all the necessary phase generation AND drivers on one chip and, like the above circuitry, requires only clock and direction inputs.

With phasing taken care of in hardware, two approaches are now possible using an internal, computer generated clock or an external clock and these are shown in Fig 7.

The 60Hz clock rate from the 555 is chosen to be the same as the 60Hz "jiffy" clock in the PET.

Thus, bits 0-3 will switch the appropriate motor ON and bits 4-7 will determine its direction, provided that pulses are being received from the 555.

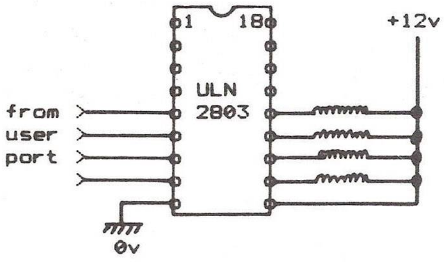
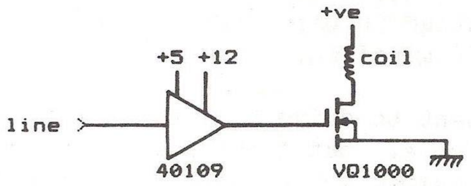


FIG 4
connections to
ULN 2803

FIG 5
interface
using FETs



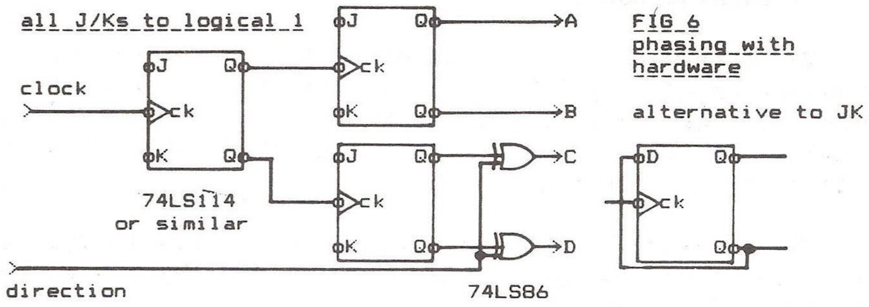


FIG 6
phasing with
hardware

alternative to JK

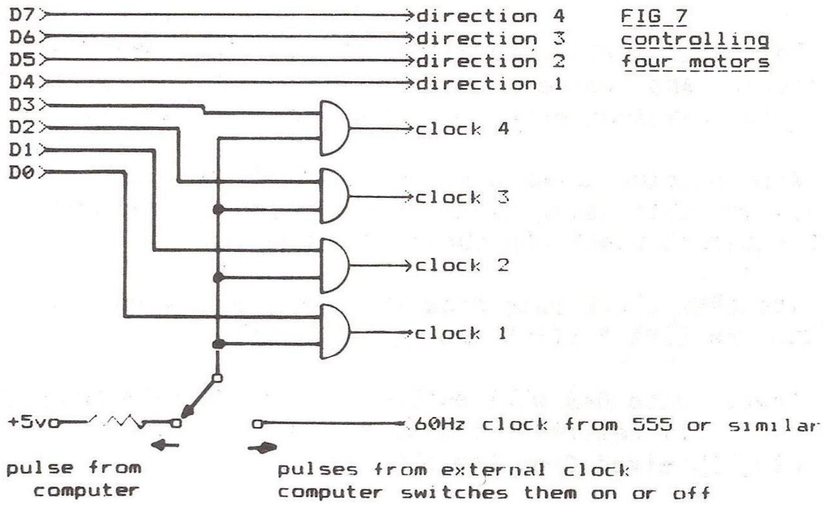


FIG 7
controlling
four motors

Thus, a byte of 11110001 will switch motor 1 on in forward mode, while 01011111 will switch all four on simultaneously but with 1 & 3 in reverse and 2 & 4 forward.

If the external clock is disconnected, the clock pulses must be generated on bits 0-3 of the output port so that, to step motor 1 forward would require the port to have a sequence of bytes of 00010001, 00010000, 00010001 ..... sent 60 times per second.

This would need to be done in machine code, but at least allows a more precise control over the motors.

ADDITIONAL COMMENTS

Normal relay cartridges aren't recommended for such applications because of speed limitations and possible reduction in contact life with the inductive loads.

Most of the chips mentioned are available from the major component suppliers such as Radiospares, Maplin and so on, and Impex Electrical (Market Road, Richmond, Surrey) have literature on stepper motors, including how to drive them. Also try the following firms:

J Bull (electrical) Ltd, - 12v motors by Smiths
34-36 America Lane,
Haywards Heath,
Sussex

Como Drills, - 6/15v variable gearbox drive motors
The Mill,
Mill Lane,
Worth,
Deal, KENT

Chiltmead, - 12/24v steppers (6/12 position - work on 5v)
Norwood Rd,
Reading, Berks

--oOo--

REVIEW - CP/MAKER

By Kevin Viney.

Introduction.

There are two principle ways of having CP/M on the PET. The first is to add an external black box - such as the Small Systems device - and the second is by means of a board like CP/Maker as described in this article. Before I describe the board itself though, perhaps a recap of CP/M is in order.

Confusingly, nobody (apart from maybe the illusive Gary Kildall - inventor of CP/M) is absolutely positive what CP/M actually stands for! Most of us who all thought we did, have been recently bewildered (see Computer Weekly January 6th) by 'experts' all saying something different. The '/', it seems, has made us wonder whether Control Program for Microcomputers (usually accepted) or Control Program/Monitor was correct....

History.

The history of CP/M goes back to 1972 when the then small company Intel asked Gary Kildall to produce a language compiler for their new 8008 - forerunner to the 8080. PL/M - programming Language/Microcomputers was the final result of this work, though along the way several other developments occurred. First, IBM defined a standard format for the 8" floppy disk and then Shugart Associates started producing $500.00 drives that soon replaced the paper tape of the time.

So with these drives, an 8080 CPU, and RAM now available, only an operating system (OS) was needed for Intel to provide each engineer with a complete development system. That OS was written (in PL/M) by Kildall, and called CP/M. In one of those much regretted decisions that often happen in history, Intel turned away CP/M, arguing that their engineers could all use PL/M on a big timesharing mainframe. So finally, after burning the midnight oil in a converted garage (sounds familiar?), CP/M was up and running in 1974. A year later however, a crucial change was made that was to give a huge boost to the popularity of this OS. Glenn Ewing of Imsai and Kildall came up with the idea of taking out all the hardware dependent bits of CP/M and sticking them in a separate section that could be modified by the Licensee to suit his own machine. This act was a godsend. The software writers were overjoyed at the prospect of vast markets, and the micromakers were pleased at the thought of a standard OS requiring modification of just 15 calls to run any CP/M software on their machine. A strong user group also helped to promote CP/M around this time.

In the face of such changes as the introduction of 5-1/4" floppies, cheap memory, the Z80 and much new software, a decision was made to re-write CP/M. Version 2.2 was created, and while it was quite crude in many respects, it was this very simplicity that enabled it to keep up fairly well with future changes in mass storage, main memory and faster CPUs. It is salutary to think that at the last count there were 3,500 different hardware configurations running this OS not including 16-bit machines using CP/M-86 or the PET!

The Operating System.

CP/M then is an environment in which programs can be run and files accessed. It provides an operating environment in which any standard CP/M program can run on any machine that uses this OS. The latter consists of three parts:

1. CCP - console command processor. This interfaces to the user and interprets the CP/M commands given by him.

2. BDOS - basic disk operating system. This handles file management operations and allows application programs to access the wide range of available system functions by using simple subroutine calls like 'read file'.

3. BIOS - basic input/output system. Here we implement the low level I/O functions like 'get' and 'put' character.

Of those mentioned above, only BIOS needs to be rewritten for use on a new computer system.

The real power of CP/M is the number of application programs available. These applications run the complete range of microcomputer software and in many cases a new piece of software is written for CP/M before being coded for other systems. Remember CP/M offers virtually every major language (eg Pascal, COBAL, FORTRAN and BASIC) as well as word-processing (Wordstar) and spreadsheets (SuperCalc) and so on. Contrary to popular belief though, not all CP/M systems are equal. They can vary for instance because of version number; location of CP/M within memory; type of diskette used and logical layout on the diskette itself.

The CP/Maker hardware.

The board itself takes around 15 minutes to fit inside the PET and should not provide any difficulty to those who feel happy wiring up a 13 Amp plug. All connections are made within the PET itself. The board contains both a Z80 and a 6502 (to prevent possible damage to the PET's own processor) and any software written under CP/M 2.2 will run. There is an up/download capability for transmitting or receiving files from other CP/M machines and an RS232 option is available for an extra £ 450.

CP/Maker gives the user 96K RAM (in a similar but not totally compatible way to the 96K PET) but you can run the normal 40-col/80-col PET software when the board is inactive. It's worth bearing in mind though that there may well be difficulties in configuring certain CP/M software packages to the smaller screen machines. Silicon Office and 96K Visicalc have been specially configured to make use of CP/Maker. Note that these are not identical programs to those running on the 96K PET. The package comes complete with the CP/M 2.2 diskette and a fairly reasonable operating manual. Included in the review pack was Digital Research's own manual which I must confess I found incomprehensible, obscure, and likely to have been written only for engineers!

If the reader is interested in a clear guide to CP/M then a book well worth considering is the CP/M User Guide, by Thom Hogan (published by Osborne/McGraw-Hill).

Operation of CP/Maker is fairly straight forward, though it is even more important than usual to read the operating manual. The command set will have to be learnt (there are few similarities with that of the PET) and many file conventions and other terms memorised. At an early stage the utility programs will need to be used (remember, unlike the PET, a COPY command is not built-in) and for the very brave, assembly language utilities like DDT (Dynamic Debugging Tool!) are included.

Operation.

After the board is activated, the familiar CP/M sign-on message is displayed and the following prompt given: A> ('A' and 'B' are the CP/M equivalents of drive numbers)

The operating manual describes how copies are made of CP/M itself onto backup disk and emphasises the importance of backing up program/data files. The manual talks of the subtle distinction between built-in commands (numbering only 7 in all), transient commands (these are brought in from disk) and line editing commands. For PET users the latter will seem alien and primitive but like many of CP/M's commands, they are there to keep CP/M as machine independent as possible. Operation of CP/M on the PET is relatively simple and while one can level criticism, it's mainly directed at CP/M itself rather that the board or any of the CP/M software.

One point which could cause a problem however is the overall speed of the board. Both the 6502 and Z80 are running at 1 MHz and benchmarks even of compiled CBASIC suggest running speeds slower than PET interpreted BASIC! For a package like say Wordstar, this may not matter but it is certainly a point to bear in mind.

Conclusions.

At £ 557.75 inc VAT, CP/Maker may seem like a lot of money. Certainly one should ask oneself a series of searching questions before parting with hard-earned cash. Do you want to solve specific problems that can only be done on CP/M and the PET? Can you afford the cost of the big-name software or will you be writing your own - CAN you write your own? It may be that you are fascinated by CP/M itself or could make use of that extra 64K. Either way, understand what you want your computer to do, read some of the manuals beforehand and don't be afraid to ask for help. Above all, if you are using CP/M for the first time, be prepared for a shock. There is no screen editor, very few built in commands, un-helpful error messages (eg BDOS ERROR), 8-letter maximum for file names, to mention just a few! Application packages such as SuperCalc will aleviate some of these problems, but remember that you may wish to build in your own user-friendly error handling routines yourself when writing programs - could you do it? I'll leave you with that thought.

CP/Maker and further information can be obtained from: Tamsys Ltd (who loaned the board for review) Pilgrim House, 2-6, William St, Berkshire, SL4 1BA. Tel: Windsor 56747.
Prices: CP/Maker £ 557.75 inc VAT.

RS232 interface £ 57.50 inc VAT.

--oOo--

Vic-20's only, they said, and threw me and my '64 out.

COMMODORE 64 - BIT MAP MODE DISPLAYS

By Brian Grainger

The purpose of this article is to explain how the VIC chip can be used to display high resolution graphics. Something which the Commodore instruction manual fails to hint at, let alone explain ! It is assumed that my article on character mode displays has already been read.

DISPLAY BASE

The DISPLAY BASE is accessed by the VIC to define which of the pixels on the screen are to be displayed. The screen resolution is 320 pixels across by 200 pixels down giving a total of 64000 pixels. One byte of memory can give the status for 8 pixels so at least 8000 bytes of memory are required.

The DISPLAY BASE is made up of 8K bytes of memory with each bit representing the status of one screen pixel. The final 192 bytes are unused for high resolution display.

NOTE:- Eight consecutive bytes do NOT form 64 horizontal pixels. They form 8 horizontal pixels on each of 8 vertical rows reading from top to bottom. Each 8 consecutive bytes of the DISPLAY BASE therefore give pixel definitions for a single screen character position. This makes for more difficult access to the pixels than need be. See my Hi-res graphic program elsewhere in the Newsletter for how it can be done.

DISPLAY BASE LOCATION

Each byte of the DISPLAY BASE has an address which can be defined by a 14-bit address code, WHEN VIEWED BY THE VIC. The corresponding microprocessor address is given by this value PLUS the value of the VIC start address (given by the offset in $DD00).

Bit 13 of the DISPLAY BASE VIC address is given by bit 3 of VIC register $18. Bits 12-0 are changed by the VIC itself to access the individual bytes in the DISPLAY BASE.

For the technically minded bits 12-3 vary at the rate of 40 locations per 8 rasters (stepping through the 40 locations on each raster). Bits 2-0 vary at the raster rate. This explains why the bit map is oddly configured.

The DISPLAY BASE start address is thus given by bit 3 of VIC register $18 followed by 13 '0' bits. On switching the 64 on. bit 3 of VIC register $18 is 0. The default DISPLAY BASE start is thus:

00 0000 0000 0000 = $0000 (VIC address)

By setting VIC register $18 the programmer can set the start of the DISPLAY BASE to one of two options within the VIC address space.

Lowest DISPLAY BASE start is $0000
Highest DISPLAY BASE start is $2000
Interval between consecutive start points is $2000.

By combining the two options within the VIC address space with the four bank options of the VIC, the DISPLAY BASE could be located anywhere in the 64K. I have not checked, but I suspect that a DISPLAY BASE area covering microprocessor addresses $1000-$1FFF or $9000-$9FFF would not work correctly due to conflict with the character generator ROM.

COLOUR.

In Hires (Bit Map) mode the VIDEO MATRIX is used in addition to the COLOUR MATRIX to define displayed colours. Colours are defined for the 8 x 8 pixel character positions NOT FOR INDIVIDUAL PIXELS. This saves on memory utilisation but does mean that one cannot make colour pictures freely since one is bounded by the character position areas.

STANDARD BIT MAP MODE.

To set up Standard Bit Map mode bit 5 of VIC register $11 should be 1 and bit 4 of VIC register $16 should be 0.

Where a bit 0 is addressed in the DISPLAY BASE the colour is given by bits 3-0 (lower nybble) of the VIDEO MATRIX address corresponding to the character position in which the pixel occurs. Where a bit 1 is addressed in the DISPLAY BASE the colour is given by bits 7-4 (higher nybble) of the same VIDEO matrix location.

MULTICOLOUR BIT MAP MODE.

Multicolour Bit Map mode is obtained by setting bit 5 of VIC register $11 and bit 4 of VIC register $16 both to 1.

As for Multicolour character mode dots are interpreted in pairs, rather than individually. The screen resolution is thus reduced to 160 horizontal by 200 vertical. The colour displayed (over the two dot positions) by the various dot pair combinations can be found from the following table:

Dot Pair	Colour Displayed
00	Colour defined by bits 3-0 of VIC register $21
01	Colour defined by bits 7-4 of VIDEO MATRIX address corresponding to the character position in which the pixel occurs.
10	Colour defined by bits 3-0 of VIDEO MATRIX address corresponding to the character position in which the pixel occurs.
11	Colour defined by bits 3-0 of COLOUR MATRIX address corresponding to the character position in which the pixel occurs.

Well that covers the aspects of Bit Map modes you need to know to carry out high resolution graphics. If required, Sprites can be displayed in conjunction with Hi-res graphic displays making for some quite interesting effects.

--oOo--

REVIEW - SUPERSPELL

Let me say at once that I think this program is one of the best I have ever had the pleasure of reviewing. It's name really belies it's versatility. Not only is it a spelling checker but it will perform other miracles as well, such as solving your crossword for you!

The program is written by Simon Tranmer who also wrote 'Superscript'. The latter is now recognised as the wordprocessor for the PET and has now been adopted by Commodore as their official wordprocessor package. Versions will be available for the '64' called 'Easyspell' to match corresponding wordprocessor 'Easyscript'. So much for the pedigree, now back to the subject!

'Superspell' comes on a protected disk and is supplied with a comprehensive but easily understood manual. The disk contains the program itself together with a dictionary of approx. 30,000 words. The dictionary exists in two different versions which give the user the choice of British or American spelling! It is possible to use either of these or a combination of both. These dictionaries are permanent and cannot be changed but, as we shall see later, they can be added to.

The first time one uses 'Superspell' it is necessary to create your own copy of the dictionary on a blank disk. This is then used by the system as the current dictionary. A supplementary dictionary can be built up during the course of time by adding words of your own that you use frequently. This is called the user dictionary. It can be merged into the main dictionary later if required. Having copied the dictionary, this disk is used from now on and the dictionary on the main disk is no longer required, although, of course, is still there if anything happens to the copy.

Having loaded the main program, the 'Superspell' disk is removed and the created dictionary disk placed in drive 0. The disk containing the file to be examined is placed in drive 1. A menu appears. The versatility of the program can be seen if we run briefly through the options which it allows:-

1. Check a Document.

This is the main purpose of 'Superspell' and the program defaults to it in the absence of any command to the contrary. (This principle applies throughout the program and speeds things up considerably). Having selected this option, the disk directory is read. If the file name is known, it is possible to enter it straight away, or else the directory can be stepped through, one entry at a time, and when the desired file is reached, it is selected by pressing return.

The program now scans the file (or set of files, as global links can be used if present). For each word encountered that has not previously been used in the same individual file, it prints a dot on the screen. This lets you know that the search is active. At the end of what seems to be an incredibly short time, statistics are shown on the screen together with any words that the program cannot match in either the main or the user dictionaries. These may be wrongly spelt or simply unique words that are not present in the dictionaries. The words can be listed on a printer if desired. A printout obtained from scanning this article can be seen at the end.

It is now possible to edit the file. Each page containing a word flagged as not being present in the dictionary is presented on the screen highlighted in reverse. It is now possible to accept it as it stands, correct it, or add it to the user dictionary. The next word will then be highlighted and so on. At the end of correcting the edited file will be written back to the disk!

2. Check Document with Frequency Report.

This option will print on the screen, or the printer, all the words used in the document, together with the number of times they have been used! A menu is provided which allows the words to be printed in alphabetical order or in order of most frequently used to least frequently used or vice versa. The search proceeds as above and the same statistics are given.

3. Print User Dictionary.

This lists out your own dictionary which you have created whilst using 'Superspell'. This can also be output to the printer if required.

4. Delete from User Dictionary.

Allows the removal of unwanted words before merging with main dictionary.

5. Merge User Dictionary.

Merges user dictionary into main dictionary and allows a copy to be made. Words are stored in the main dictionary in a compressed form so less space is taken up than when in user dictionary and searching is that bit faster. The maximum number of words that the main dictionary will accommodate depends on the system and is approx. 60,000 on a 4040 and 200,000 on a 8050.

6. Search Dictionary.

It is possible to search the dictionary and see if a word is in it. Full CBM pattern matching is implemented so that, for example, one could search for all words beginning with ann by using ann*, or a crossword clue where only certain letters are known by using ?u?l??a??. A list will be given of all possible matches (duplicate, guildhall, sublimate).

Additional Facilities.

Provision is made for printers with, or without, automatic line feed and the use of continuous or separate sheets of printing paper.

Any Criticisms ?

With any program, however good, one can always say, if only... Often the programmer will also have said the same thing to himself and rejected it because of some impracticability unknown to me, the user. If only the search through the dictionary started at the correct first letter instead of always at A. If only the list of new words just put into the user dictionary stayed on the screen long enough to read. If only the program did not alter the order of files on the original disk. These points are all minor ones. Go and buy this program!

Distribution.

'Superscript' and 'Superspell' can now be obtained separately or together on one disk. Details from Precision Software Ltd. 4, Park Terrace, Worcester Park, Surrey KT4 7JZ. Tel: 01-330 7166. (ICPUG members are reminded that substantial discounts are available - contact John Bickerstaff our discounts officer).

Summary.

Almost an essential add-on for any owner of 'Superscript'. Superb for those who canot spel. Indispensable for authors who require simple word counts or who wish to improve their writing by avoiding overuse of their pet words.

Example:
We put the above article through 'Superspell' and this is what it told us -

```
 unrecognized word list for filename ... Superspell
 total  number of words = 1079
 number of unique words = 407
 number of sentences    = 72
 number of paragraphs   = 27
 average word length    = 4.59
 approx                   belies
 bickerstaff              canot
 easyscript               easyspell
 icpug                            impracticability
 jz                               kt
 spel
 no of unrecognized words = 11
```

I also used 91 'the's, 32 'is's, 29 'of's, etc. Care to count them? [Superspell did not find the use of both 'disc' and 'disk' - Ed]

By David Annal.

--oOo--

PRESTEL - THE STORY SO FAR

By Stephen Rabagliatti

By the time that you read this article, you will probably already have bought your adaptor and understand all about Prestel, Telesoftware, Ceefax, Private Viewdata, Electronic Mail and all the other buzz-words currently being hurled about by the popular press .... such is the speed at which things are moving on this front at the moment !

Suffice it to say that one of the most exciting developments in micro-computing is at last about to become an economic proposition for the vast bulk of micro enthusiasts as opposed to the limited few who have been able to afford it up to now. I refer to the imminent launch of MICRONET-800 the major database on Prestel specifically designed for (and partly run by) the Micro-computer user and enthusiast.

Through Micronet, you will be able to obtain all of the necessary hardware and software to allow your PET to communicate with a Nationwide network of computers that contains news, advice, technical information, software (including business, games, utilities), an Electronic Mail service, all for around £ 50.00.

For those of us that have already been using Prestel, the whole idea is very attractive as it means that there will be more information of use available and also a broader range of people able to use the impressive Mailbox service.

The main advantage to all of us, however, will be Telesoftware. It is possible to store PET, CBM64, Apple, etc. software on the Prestel system in a form that can be decoded (with transmission error checking) by the micro, giving one access to a potentially unlimited bank of software to run. There are already well over 100 PET programs from the ICPUG library stored on the system and by the time you read this there will be at least 250 free programs for each of the micros supported by Micronet. 10% of these programs will change each month and there will also be commercial programs that you may have to pay for stored in other sections. These are likely to be programs of the 'Space Invader' variety and also applications files for programs like Visicalc, Silicon Office etc.

Not only will you be able to 'download' the software for your PET, but also you will be able to download the programs put up for Apple, BBC, Sinclair, etc., and with little modification these programs will run in your PET. This can only be good for all micro users in the long run as more and more software will become available to the public domain.

ICPUG already has a number of its own pages on Prestel and these contain telesoftware for ICPUG members only and information about local group meetings, ... in fact it is possible for members of ICPUG to actually edit their own local group's pages through the ACC which organises the User Group's activities on Micronet.

If you are interested then contact your local ICPUG organiser who should be able to help.

The costs.

Subscription to Micronet (including Prestel sub) is £ 52 per year and there is also a joining fee of about £ 50 which covers the cost of the subsidised equipment. Therefore budget for £ 100 in the first year plus telephone charges. As Prestel is available at local call rate for about 90% of the UK and access time is free after 6pm, you can keep these costs very low with a little caution.

--oOo--

THE 64 COLUMN

By Mike Todd

Apart from a quick foray inside the Commodore 64 in the November '82 Newsletter, there's been very little coverage from within ICPUG.

This issue puts that right with a vengeance. As well as 64 material elsewhere in this Newsletter, I'm starting a regular 64 column along similar lines to the Vic column which I've been putting together for the last year or so.

The BASIC on the 64 is identical to that on the Vic-20 and the use of RAM up to $0400 (1024) is so similar that I would recommend anyone wanting details of this RAM usage to refer to my memory maps in the July '82 Newsletter.

There are a couple of differences due to the fact that the 64 has a 6510 processor which uses locations 00 and 01 as an extra input/output port - 00 is used as a data direction register, and 01 as the port (bits 0-2 are used to control the memory map and bits 3-5 are part of the cassette interface). Therefore the USR jump vector (USRPOK) has moved to $310-$312 which are unused on the Vic.

The new CIA chips ("Complex Interface Adapters") are used in a slightly different way to the timers on the interface chips in the Vic. As a result, there are a couple of locations used as timer flags as follows:

2A1	CIA #2 NMI control flag - used by the RS232 interrupt (NMI) routine to determine which section of the RS232 routine to execute.
2A2	CIA #1 Timer A control flag - used during tape read to start/stop timer A
2A3	CIA #1 Interrupt log - used during tape read to identify if timer A caused the interrupt
2A4	CIA #1 Copy of 02A2 used to identify timer enabled during tape read

2A5 Used during screen operations as a screen row marker - Vic uses location F2 (SCRWLG) which is part of the screen line table on the 64.

2A6 Flag set up during initialise routines to identify NTSC or PAL version of the 64. If 2A6 (678 decimal) is 1 then the 64 is a PAL version; it is 0 for NTSC. The flag is used to select the appropriate set of timing constants which are different for the different clock frequencies of the two models. Further discussion on these differences later.

MEMORY CONTROL

The memory maps on pages 313/314 of the Nov '82 Newsletter show the versatility of the 64's memory map.

The three ROMs on the 64 (BASIC, KERNAL and character generator) are switched in and out by setting or clearing bits 0-2 of location 01 as follows:

$\overline{\text{LORAM}}$: bit 0: if set to 1, the BASIC ROM ($A000-$BFFF) is available - if set to 0, this ROM is replaced by RAM.

$\overline{\text{HIRAM}}$: bit 1: if set to 1, the KERNAL ROM ($E000-$FFFF) is available - if set to 1, this ROM is replaced by RAM.

$\overline{\text{CHAREN}}$: bit 2: if set to 1, the character generator ROM is NOT available to the programmer. Instead, the 64 sees the I/O chips at $D000-$DFFF. If set to 0, the character generator becomes available and the I/O chips are no longer accessible.

There are several other lines which affect the memory map.

The $\overline{\text{EXROM}}$ line on the expansion socket controls the area $8000-$9FFF and if pulled low will switch the RAM away, leaving space for an expansion cartridge.

The $\overline{\text{GAME}}$ line, also on the expansion socket, is pulled low to force the memory map of the 64 to look like the map of the MAX. MAX cartridges have this line held low and the 64 will appear exactly the same as the MAX when they are used.

The logic used to control the memory map with these (and a few other) control lines is quite complicated and Commodore have opted to use a programmable logic array in the 64 to implement this.

This PLA receives all the necessary control and address lines and converts them into ROM or RAM enable lines. As a result, the effect of different combinations of control lines is difficult to predict and there are some odd interactions which the diagram on page 314 of the November Newsletter goes someway to illustrating.

GOING FURTHER INSIDE THE 64

Before going on to look at some practical uses of these lines and some other interesting 64 bits and pieces, I thought I'd have a quick rummage around the innards of the 64 and offer some comments.

STEALING TIME

First of all, the new VIC chip used on the 64 "shares" some of the memory space used by the main processor. In fact this chip is the "heart" of the 64's timing, and every so often it "steals" time from the processor and accesses memory itself to get information on characters, sprites and so on - it's a bit like an interrupt sequence, except totally in hardware.

This can take quite a long time, and so the processor's timing is upset. This has dire consequences during serial bus or cassette access and, by turning the VIC chip off just before accessing cassettes, the 64 manages to get round the problem. It is this which causes the blank screen during cassette operations.

The 64 doesn't do this when using a device on the serial bus and so problems can occur. For instance, if this pause occurs while the 1540 disk drive is waiting for the 64 to respond, the delay can be too great and the disk drive hangs up. It can also cause the 1515 printer to hang up, usually when printing long lines.

A change has been made to the software in the 1540 drive to allow it to cope with this delay and this is why you need to change a ROM in the 1540 or use a 1541 (which has the new ROM fitted) when using the 64.

For those who don't have the 1541 ROM, or are using the printer, it is possible to turn the VIC chip off using: POKE 53265,11. When you're finished, use POKE 53265,27 to turn it back on again - or you could just press STOP+RESTORE which would do it for you.

A BIT FURTHER INSIDE

I've already mentioned the PLA used to control the memory addressing, but there's a control line I've not mentioned which affects the PLA and the 6510 processor.

This is the DMA line on the expansion connector. When pulled low, this allows an external device to have control of the address, data and read/write lines within the computer, although the VIC chip still has access to memory. An additional line on the connector allows the external device to predict when the VIC chip will be taking over the timing.

This would be used by a second processor card (Z80? - sorry, wash my mouth out with soap and water!!!) which, in theory at least, should expand the range of software available.

The two CIA interface chips have a useful facility. They both have time-of-day clocks built in. These are timers which consist of 24 hour clocks organised as 4 registers - hours; minutes; seconds; tenths of seconds.

These clocks are synchronised to the mains frequency and operate totally independently to the system clock. On USA machines the mains frequency is 60Hz. Fortunately, these timers can be reprogrammed to work on our 50Hz. They even allow an "alarm" interrupt to be generated whenever a preset time is reached.

The programming of these timers can be a bit tricky, and I'll be covering it in a later column. Also, there is no battery backup facility, so both clocks stop and are reset when the 64 is turned on.

With the external power supply, the 64 runs much cooler than the Vic, and with a built in modulator, the number of trailing leads is reduced by one.

The fact that the modulator is built in to the 64 doesn't mean that video and audio aren't available. These are still available on the DIN connector although these are slightly different to the Vic's connections. For instance, pin 1 no longer has +5 volts for the modulator, instead it has the luminance signal from the VIC chip, and pin 5 now is an input to the 64, which feeds the auxilliary input of the SID chip through a DC blocking capacitor.

As a final comment on the innards of the 64, while the two joystick controls are available independently through bits 0-4 of registers at $DC00 and $DC01, the X and Y inputs of the VIC chip used by the "paddles" need to be multiplexed. This is done with a 4066 analogue switch chip and the necessary selection is done using bits 6 & 7 of $DC01. There is only one light-pen input and that is on the push button input of control port 1.

BUGS IN THE 64

Those who are familiar with the development of Commodore computers will not be surprised that there are a few bugs in the 64.

Apart from problems reported of a failure of some 64's to load tapes recorded on Vic-20s, there are a couple of bugs carried over from the Vic-20 such as the inability to use a TAB or SPC at the start of a PRINT# statement - see the January '83 Newsletter, page 11.

There is also a serious bug reported from the USA (and I've not yet been able to check it on a UK 64) which occurs when typing on the 23rd, 24th or 25th line of the screen with a line which is over 80 characters long, if you try to delete the 80th character, the 64 does some odd things.

You will also notice that, typing FRE(0) produces a negative number of bytes free! The reason for this is the fact that FRE(0) returns an integer in the range -32767 to +32767. Since the value of FRE(0) on a 64 is greater than 32767, the integer routine assumes that the most significant bit, which is now a 1, is actually indicating a negative number. The cure is quite simply to add any negative FRE(0) value to 65535 to find the correct number of bytes free.

There are some Vic bugs which have been put right in the 64. Firstly, it looks as though the RS232 driving software will work correctly in both three line and multi line modes.

On the Vic, reading ST would not return the RS232 status register. This meant that the RS232 status always had to be checked using a PEEK, but on the 64, reading ST immediately after an RS232 operation will return the correct value and will clear the status register at the same time.

The CTRL key on the Vic was only used to access the colour controls and the RVS facility. It seemed a bit of a waste considering that normal ASCII keyboards use the CTRL key to access the ASCII control characters.

Well, on the 64 it appears that this is implemented anc pressing CTRL-A will generate CHR$(1), CTRL-Z generates CHR$(26) and so on! It may seem a trivial improvement, but it has far reaching implications in the use of the 64.

THE RAM BENEATH THE ROM

I've already mentioned the ability to switch out the ROMs and replace it with RAM and hinted that there's a use for it. Well, there is!

At long last, there's the possibility of modifying the BASIC interpreter and the KERNAL to cure bugs and so on. Simply by copying the ROMs into the RAM which occupies the same address and switching over to this RAM, the ROM contents can be got at.

To do this, first of all, transfer the BASIC ROM into the underlying RAM by:

```
FOR I = 40960 TO 49151 : POKE I,PEEK(I) : NEXT
```

If you POKE to a ROM location, the data goes into the RAM underneath so you now have the BASIC in RAM, and to get at it you need to clear bit 0 of $0001. Its normal value is 55, so POKE 1,54 will turn the RAM BASIC on - POKE 1,55 puts it back.

Transferring the KERNAL ROM is simply a matter of:

```
FOR I = 57344 TO 65535 : POKE I,PEEK(I) : NEXT
```

This RAM is then selected by POKE 1,53 (52 if you want both BASIC and the KERNAL), and restored by POKE 1,55.

Since the interrupt routines may need to access the KERNAL ROM just as the switch is taking place, it's a good idea to disable the keyboard scanning interrupt. This is done by POKE 56333,127 and restored with POKE 56333,129.

AND THE CHARACTER GENERATOR

You'll also have to do this if you want to get at the character generator which occupies the same address space. as the I/O chips.

To get at it, disable the interrupts to prevent the KERNAL trying to access the I/O chips, then set CHAREN low (POKE 1,51) - you can now PEEK at the character generator (53248 to 57343). Before you attempt to output anything, you MUST restore the 64 to normal by resetting CHAREN (POKE 1,55) and restoring the interrupt.

This facility means that the character generator can be copied anywhere in memory giving the ability to redefine the character set. An article from Jim Butterfield on how to do this is reprinted elsewhere in the Newsletter.

MODIFYING BASIC

If you want to modify BASIC, I would suggest that you don't tamper with the main entry points of the routines but rely on patches to the code. There is of course no specific reason why you shouldn't muck about with the code locations, but life is likely to get rather complicated and the consequences could only be predicted with a VERY good knowledge of the principles of the BASIC and KERNAL codes.

OK - so we don't want major re-writes of the ROMs. But there is ample space for user patches to the ROM sets and this is in the spare RAM $C000-$CFFF which is always available, unless it's being used for video tables.

Then there's always the RAM from $8000-$9FFF which can either be used by ROM cartridges (by pulling the $\overline{\text{EXROM}}$ line low) or by BASIC or by anyone....

Speaking of cartridges, there are three areas that can be occupied by cartridges and these can be seen by referring to the memory maps in the November Newsletter.

The first cartridge area is $8000-$9FFF and this is where games cartridges are likely to fit as they car autostart easily and still have full access to BASIC and KERNAL ROMs.

It is also possible to have a cartridge sitting "on top" of the BASIC ROM or even one "on top" of the KERNAL ROM. In this way, new languages or operating systems could be implemented, although I suspect that the best way to implement such things would be to "soft-load" into the RAM space from disk (or even cassette?).

With the total capability to copy ROM packs into RAM, and then SAVE them to disk or cassette, software companies are inevitably going to try to prevent this because of the likelihood of the software being ripped off.

On the Vic, this isn't so easy. Apart from needing extra RAM at the cartridge location, SAVEing and LOADing requires a good inside knowledge of the Vic.

I've no doubt that software manufacturers are going to get very clever in this area. The principle way of identifying the fact that the software is in RAM and not ROM is to write to the location and then try to read the byte back. I can think of at least half a dozen ways that this could be done very easily - all it needs is a way of hiding the necessary code in the software so that the computer "Sherlock Holmes'" can't find it.

It may be easier for the ROM pack to actually use the RAM space either for important subroutines or to hold data. This way, the "ripper-offer" will need to do a major re-write of the ROM code to cope with the fact that he has only one bank of memory to play with, whilst the ROM has two!

INSIDE THE BASIC ROM

Elsewhere in the Newsletter you'll find a list of the BASIC ROM routines, together with their BASIC 2 and Vic equivalents.

In actual fact, the 64s BASIC ROM is almost byte for byte identical to that on the Vic, except that it is relocated to $A000-$BFFF.

PET EMULATOR

There is a program selling at $30 in the USA, and hopefully to be available soon in the UK, which will allow the 64 to look just like a Commodore PET.

The implications are enormous as there is a huge range of PET software already available and this could fill the software gap which currently exists with the 64.

It is reported to cope easily with screen and zero page PEEKs and POKEs, and CB2 sound generation and many programs work without any modification at all.

Some features are missing due to the major differences in hardware between the PET and the 64. For instance, the 64 doesn't have the numeric keyboard like the PET and complicated programs using SYS calls into the ROMs are not likely to work unmodified.

The emulator contains the so called "DOS WEDGE" which allows easy access to disk commands.

There is also rumoured to be a Vic emulator in the pipeline which will allow the 64 to run Vic programs and cartridges. This is being developed in the USA and is not (and may not be) available in the near future.

SPRITES

I don't intend to go into the theory of generating sprites at this stage; however, assuming that you've managed to generate them, PEEK(53278) will detect a sprite-sprite collision and PEEK(53279) will detect a sprite-background collision. It is suggested that these locations are checked TWICE, and if either shows a collision, then one did occur.

I have had a couple of queries asking how to have more than eight sprites on the screen at any one time. Early publicity was hinting at this as a possibility.

Yes, it can be done, but the software must be written in machine code and is likely to be extremely sophisticated.

There are two ways of doing this. The first involves changing the sprite pointers every sixtieth of a second so that, on the next screen scan there is a different set of sprites.

This would effectively allow a couple of separate "screens" of sprites to be multiplexed onto the screen. The disadvantage would be that, as the number of extra pages increased, so the sprites would start to flicker as they were on the screen for less and less time.

The other way is to consider the sprites made up in bands across the screen and to use the raster count to flip to different sprite pointers for each band of sprites on the screen.

This latter technique may not be very easy to use, and may not even work! I'd be interested to see if anyone can put together a multiple sprite utility which is easy to use and which could be put into the, at present, non-existent 64 software library.

Before leaving sprites, there is a problem in compatibility between NTSC and PAL machines as a result of the screen timings.

First of all, the NTSC screen has 262 raster lines per screen with a frame rate of 60 per second, and the PAL screen has 312 lines per screen at a frame rate of 50 per second.

This shouldn't affect the use of sprites as the vertical positioning is the same on both systems.

However, when positioning sprites horizontally, the NTSC machines have 512 possible positions (0-511) with positions 24-343 actually displayed in the screen window. Scrolling a sprite smoothly off the screen right is simply a matter of incrementing its horizontal co-ordinate beyond 343 ($157), up to a maximum of 511 ($1FF). Scrolling it on from the left is a matter of starting the horizontal co-ordinate at 0 and incrementing it until it appears fully on the screen at position 24 ($18).

For PAL versions, this is exactly the same, except that positions 504-511 ($1F8-$1FF) don't actually exist.

The problems start when trying to move horizontally expanded sprites on and off the screen. On both NTSC and PAL machines, there is a different right hand co-ordinate limit at 488 ($1E8) for NTSC and 480 ($1E0) for PAL.

Scrolling an expanded sprite onto the left hand side of the screen means starting at position 489 ($1E9) NTSC or 481 ($1E1) with PAL. Because the PAL machines don't have positions 504 to 511, scrolling must involve incrementing to 503 and then restarting at 0 whereas with NTSC, this limit is 511.

This difference in sprite standards between both sides of the Atlantic must be borne in mind if writing software which may be used in the USA and is one good use of the standards flag at $02A6 (422) mentioned earlier in the column.

USING PET PERIPHERALS ON THE 64

Many people have written to me asking if PET peripherals can be used on the 64 and, apart from the standard Commodore cassette machine (the C2N) which is used on all PETs and Vics, the immediate answer is no.

But there is hope! At the time of writing (beginning of February) Commodore have started production of an IEEE interface cartridge which is totally transparent in use and will allow the full Commodore range of disks and printers to be used.

OTHER 64 PRODUCTS

Firstly, for those wanting to use the RS232 port on the 64, the Vics RS232 cartridge will plug into the user port and provide all the necessary voltage conversions for using RS232 peripherals.

There are many software suppliers already preparing software for the 64 and by the time you read this the software market should be starting to build up. Commodore already have a 64 Programmer's Reference Guide, which I haven't yet seen but is reported to be very much better than the Vic reference guide.

Their first software product is the brightly packaged, "Start Programming with Gortek and the Microchips"; a pack to teach young children computer programming through a series of cassette programs in story book style.

The popular "Introduction to BASIC part 1", initially introduced for the Vic, is available at £14.95 and is a more "adult" method of learning computer programming on the 64.

The first word-processor for the 64 is "Easy Script" which is available on disk for £75.00, and by the beginning of April its companion "Easy Spell" should also be available.

The infamous "Simons BASIC", which will give the 64 a much more powerful and structured BASIC for an extra £50.00 should also be available (on disk) at the same time.

There is also to be an assembler package for £50.00 and a range of educational programs at only £9.99 each and some games at only £4.99.

With uncertainty surrounding the production of the MAX (the tiny computer containing the same facilities as the 64 apart from the keyboard, BASIC and the RAM!) I suspect that there will be few games cartridges available for a while. Until then, we'll have to make do with cassettes.

FINALLY

There are a couple of other odds and ends to finish this first 64 column.

As I write this, the number of 64s in the UK is still very small. Apart from dealer's demonstration models and those in software houses, there can only be a handful of the potential market who actually have a machine in front of them! Indeed, I'm one of them - I've not actually seen a 64 since this time last year when I got my hands on a prototype for a few days.

There have been many problems besetting Commodore's production factory in Germany, and 64s are still not coming into the UK at a satisfactory rate. Things should improve soon, and Commodore may be setting up a UK factory in the not too distant future. Therefore, much of what is in the column is either from investigations with the prototype, rummaging through my disassembly of the ROMs (a kindly 64 owner dumped their contents onto cassette for me!) or rummaging through circuit diagrams, literature and so on. If you have any comments, criticisms or ideas about the 64, let me have them and I'll put them in the next column.

Some other news: I've already mentioned the imminence of an IEEE cartridge and the Programmer's Reference Guide (which is also in short supply) but there should soon be a speech synthesis unit for the 64 available.

Commodore in the USA have set up their own Speech Technology unit and have developed their own voice board which is reported to have a female voice and sounds much better than others which are available for other machines.

Finally, finally - what of the future? There are rumours of a portable 64 with built in screen and disk drives!

--oOo--

BUBBLE SORT DEMONSTRATION PROGRAM

By David Barratt.

The following program gives a graphical display of a bubble sort in action. The display is formatted to suit the 8032.

```
100 PRINT"<2clr><rvsN>":DIMA$(22): C$="<hcsr><25dn>"  -*
105 S$="                          " : BA$="<18lft>"
110 U=31 : POKE59468,14 : C=80 : L=25 : GOSUB 460
115 PRINT TAB(U)"###################   [*-see notes - Ed]
120 PRINT TAB(U)"<rvs> d e m o  s o r t <off>"
125 PRINT TAB(U)"<rvs>###################<3dn>" : U=4
130 PRINT TAB(U) "This is a simple demonstration of how
    the computer sorts ";
135 PRINT TAB(U) "things.<dn>"
140 PRINT TAB(U) "This particular program uses the
    <rvs>bubble sort<off>'. ";
145 PRINT TAB(U) "The bubble sort is one <dn>"
150 PRINT TAB(U) "of the slowest and simplest sort
    algorithms, ";
155 PRINT TAB(U) "of the many that are <dn>"
160 PRINT TAB(U) "available. Because speed is not a
    critical factor in this";
165 PRINT TAB(U) " demo, this<dn>":PRINT TAB(U) "method
    will suffice.<3dn>"
170 PRINT TAB(U+13) "DO YOU WANT TO ENTER YOUR OWN DATA
    (Y/N)?"
175 M=34754 : POKE M,113 : POKE M-80,93 : POKE M-160,112
    : POKE 34607,115
180 FOR I=1 TO 12 : POKE M-160+I,64 : NEXT
185 PRINT TAB(68) "<3dn>D. BARRATT"
190 GET E$ : IF E$= "" GOTO 190
195 IF E$ <> "N" AND E$ <> "Y" GOTO 190
200 PRINT"<clr><rvsN><3dn>how many elements to be
    sorted ? (maximum = 22) ";
205 MAX=2 : GOSUB 525 : NUM=VAL(RET$)
210 IF NUM > 22 OR NUM=0 THEN RET$="" : GOTO 200
```

```
215 IF E$="N" GOTO 245
220 MAX=14 :   FOR O= 1 TO NUM
225 PRINT TAB(U) "<dn>enter data element"O" ";:GOSUB 525
230 RET$=RET$+"                         " : A$(O)= LEFT$(RET$,14)
235 NEXT O  : GOTO 250
240 :
245 FORI=1 TO NUM   : READ A$(I) :  NEXT
250 FOR DEL=0 TO 999 : NEXT
255 PRINT"<clr><rvs>sorting<off>" : TAB=((80-LEN(A$(1)))
    /2)
260 FOR I= 1  TO NUM      :  PRINTTAB(TAB) A$(I)  :
    NEXT
265 :
270 FOR A= 1 TO     NUM-1
275 FOR B= (A+1) TO NUM
280 IF  A$(B) >= A$(A) GOTO 305
285 GOSUB 335
290 TEMP$= A$(A)
295 A$(A)= A$(B)
300 A$(B)= TEMP$
305 NEXT B,A
310 POKE 158,0 : PRINT"<hcsr><rvs>finished"
315 POKE 59467,16 : POKE 59466,5 : R%=25+RND(I)*225 :
    FOR N=1 TO 3
320 POKE 59464,R%  : FOR M=1 TO 50 : NEXT M
325 POKE 59464,0   : FOR M=1 TO 50 : NEXT M,N
330 END
335 :
340 NO= NOT NO : IF NO THEN F= B : N= A : GOTO 350
345 F= A : N= B
350 F$= LEFT$(C$,F+1) : T1$= F$  : S1$= " "+A$(F)+" "
355 N$= LEFT$(C$,N+1) : T2$= N$  : S2$= " "+A$(N)+" "
360 :
365 FOR C=0 TO 18      : PRINT F$ TAB(TAB+C) S1$
370 :                  : PRINT N$ TAB(TAB-C) S2$ : NEXT
375 :
380 L= LEN(A$(N))+1    : M= LEN(A$(F))+1
385 T3$="<up>"+LEFT$(S$,L)+"<dn>"+LEFT$(BA$,L)+A$(N)+"
    <dn>"+LEFT$(BA$,L)+LEFT$(S$,L)
390 T4$="<up>"+LEFT$(S$,M)+"<dn>"+LEFT$(BA$,M)+A$(F)+"
    <dn>"+LEFT$(BA$,M)+LEFT$(S$,M)
395 T3$= T3$+" "       : T4$= T4$+" "
```

```
400 :
405 S= 1 : SE= -1     : IF N < F THEN S= -1 : SE= 1
410 FOR D= F TO N+SE   STEP S
415 T1$=   LEFT$(C$,LEN(T1$)+S)
420 T2$=   LEFT$(C$,LEN(T2$)-S)
425 PRINT T1$  TAB(TAB+C-1) T4$
430 PRINT T2$  TAB(TAB-C+1) T3$
435 NEXT
440 :
445 FOR C= 19 TO 0 STEP-1     : PRINT N$ TAB(TAB+C-1) S1$
450 :                         : PRINT F$ TAB(TAB-C-1) S2$
    : NEXT
455 RETURN
460 :
465 :
470 REM  DRAW CENTRALIZED BORDER (C= COLUMNS  L= LINES)
475 :
480 :
485 IF C<3 OR C>80 OR L<2 OR L>25 THEN RETURN :  LIMITS
    EXCEEDED.
490 SC =32768+INT((80-C)/2)+80*INT((25-L)/2)
495 TL =SC            : TR =SC+C-1        : BL =SC+(L-1)*80
     : BR =BL+C-1
500 POKE TL,112       : POKE BR,125
505 FOR  I=1  TO L-2 : POKE TL+80*I,93 : POKE BR-80*I,93
     : NEXT
510 POKE TR,110       : POKE BL,109
515 FOR  I=1  TO C-2 : POKE TR-I,64     : POKE BL+I,64
     : NEXT
520 RETURN
525 :
530 :
535 REM GENERAL INPUT 1.0
540 :
545 :
550 RET$=""
555 FOR I=0 TO 45 : GET T$ : IF T$="" THEN PRINT"<rvs>
    <lft>"; : NEXT : GOTO 565
560 GOTO 570
565 FOR I=0 TO 45 : GET T$ : IF T$="" THEN PRINT"<off>
    <lft>"; : NEXT : GOTO 555
570 :
```

```
575 PRINT "<off>";
580 IF T$= CHR$ (13) THEN PRINT "·<lft>" :      RETURN
585 IF T$= CHR$ (20) THEN   GOSUB 610
590 IF LEN(RET$)  =  MAX    GOTO  555
595 IF T$ <" " OR T$ >"Z"   GOTO  555
600 RET$= RET$+T$
605 PRINT "<off>" T$; : T$="" : GOTO 555
610 IF LEN(RET$) < 1 THEN RETURN
615 RET$=LEFT$(RET$,LEN(RET$)-1) : PRINT " <2lft><off>";
620 RETURN
625 :
630 SAVE"@1:DEMO SORT",8:VERIFY"DEMO SORT",8 :STOP
635 DATA FLOPPY TAPE,PASCAL,FORTRAN,BASIC,COMMODORE,
    MEGAFLOPS
640 DATA COMAL,SYNTAX ERROR,INTERFACE,COMPUTER,FLOPPY
    DISK
645 DATA BINARY,HEXADECIMAL,OCTAL,FLOATING POINT,
    ARITHMETIC
650 DATA HOLLERITH,PAGE ZERO,ERGONOMICAL,RANDOM ACCESS,
    MEMORY,ALGOL
```

Editor's note: This program was processed to create an ASCII text file for printing. As a result, some adjustment of upper/lower case may be required.

Furthermore, lines 115 & 125 should have a graphic character instead of the '#'. This character has a screen code (POKE) of 98 (see Newsletter p361, November for method of entry). The <rvsN> in lines 100 & 200 is to create a control character. Entry may be accomplished by terminating the quote mode with a quote character, which you then delete, type <rvs> then 'N' (or is it 'n' ?) then to get back into quote mode, enter another quote and delete it. The correct operation should be in graphics mode with closed line spacing.

--oOo--

COMMODORE 64 HI-RES GRAPHICS

By Brian Grainger

Those who have finally got their '64' will realise that although the machine is capable of high resolution graphics the manual with the computer makes no mention of how to use them. The purpose of the following program is to demonstrate how to plot with high resolution graphics.

I suggest that you do not get put off by the length of the program. The meat of the program is very short indeed. I have REMmed it extensively to try and give some indication of how the program works. I had two aims. Firstly to indicate how to use some of the facilities of the '64'. Secondly, for those beginning in machine code, I hope that my REMs will help to show how it can be used to do arithmetic.

I have not attempted to explain the values used in POKE statements. Further explanation of these will be found by reading my articles on VIC addressing and the '64' Bit Map Mode (if the Editor publishes them).

Finally I would like to say that the program IS a demonstration. Full high resolution facilities for the '64' will require more functions than 'PLOT' and more efficient use of memory could be made if the RAM below the BASIC ROM was used for Hi-Res screen storage. Well, that's the preliminaries. Here is the program:-

```
 5 REM ********************************
 6 REM * PROGRAM TO DEMONSTRATE HIGH *
 7 REM * RESOLUTION PLOTTING ON THE  *
 8 REM *       COMMODORE 64          *
 9 REM ********************************
10 :          POKE 56,64 : CLR
11 REM * PROTECT HIRES AREA FROM PROG*
12 REM ********************************
13 :   INITIALISE= 49152 : PLOT= 49197
14 REM * THAT SET POINTERS TO M/C    *
15 REM ********************************
16 :     VC=13*4096+13*256:V=13*4096
```

```
17 REM * THAT SET A POINTER TO:-      *
18 REM * CIA2 PRA (VC), VIC CHIP (V) *
19 REM ******************************
20 PRINT"<clr><2dn>LOCATING MACHINE CODE IN $C000+"
21 FORM=49152TO49310
22 READA$:A=0
23 FORI=1TO2:B=ASC(MID$(A$,I)):A=16*A+B-48+7*(B>57):
   NEXT
24 POKEM,A:PRINT"<hcsr>    <hcsr>";A
25 NEXT
26 REM * SET UP M/C IN $C000 RAM AREA*
27 REM ******************************
50 :        POKEVC,PEEK(VC)AND254
51 REM * SET UP VIC CHIP TO POINT TO *
52 REM * BANK 1 ($4000-$7FFF)        *
53 REM ******************************
60 :          POKEV+24,8
61 REM * SET UP HIRES DISPLAY BASE   *
62 REM * TO VIC CHIP START + $2000   *
63 REM *        ** $6000 **          *
64 REM * ALSO THE  VIDEO MATRIX      *
65 REM * (WHICH HOLDS COLOUR INFO IN *
66 REM * BIT MAP MODE) IS SET TO VIC *
67 REM * CHIP START + $0000          *
68 REM *        ** $4000 **          *
69 REM ******************************
70 :       POKEV+17,PEEK(V+17)OR32
71 REM * SELECT BIT MAP DISPLAY MODE *
72 REM ******************************
80 :           SYS INITIALISE
81 REM * THAT CLEARED THE HIRES AREA *
82 REM * + SET COLOUR DEFINED BY M/C *
83 REM ******************************
87 REM
88 REM ******************************
89 REM * USE LINES   100-499     TO  *
90 REM * DEFINE AND PLOT FUNCTION    *
91 REM * (CRAM AS MUCH ON A LINE AS  *
92 REM * POSSIBLE TO SPEED THINGS UP)*
93 REM ******************************
97 REM
```

```
  98 REM *******************************
  99 REM
 100 X=10:Y=100
 110 FORS=0TO6.5STEP0.02:SYS PLOT,INT(X*SIN(S)+160.5),
     INT(Y*COS(S)+100.5):NEXT
 120 X=X+5:Y=Y-5:IF X<=160 GOTO 110
 500 REM
 501 REM *******************************
 502 REM
 600 T=TI
 610 IF TI<T+600 GOTO 610
 620 PRINT"
 630 PRINT<clr>"       PLOTTING IS NOW COMPLETED"
 640 PRINT
 650 PRINT"  TO RETURN TO NORMAL OPERATION PRESS"
 660 PRINT"                 <rvs>RETURN<off>"
 670 PRINT"   OTHERWISE THE PLOT WILL RETURN AT"
 680 PRINT"           INTERMITTENT INTERVALS"
 690 :              POKE 198,0
 700 :        POKE VC,PEEK(VC) OR 1
 710 :             POKE V+24,20
 720 :     POKE V+17,PEEK(V+17) AND 223
 730 :                 T=TI
 740 :     GETA$:IF A$=CHR$(13) THEN END
 750 :        IF TI<T+300 GOTO 740
 760 REM * AFTER A DELAY OF 10 SECS    *
 770 REM * THAT SET UP A MESSAGE,      *
 780 REM * CLEARED THE KEYBOARD BUFFER*
 790 REM * AND RESET TO CHARACTER      *
 800 REM * DISPLAY MODE                *
 810 REM * IF 'RETURN' PRESSED WITHIN *
 820 REM * 5 SECS THE PROGRAM ENDS.    *
 830 REM ******************************
 840 :        POKEVC,PEEK(VC)AND254
 850 :              POKEV+24,8
 860 :       POKEV+17,PEEK(V+17)OR32
 870 :               GOTO 600
 880 REM *                            *
 890 REM * RETURN TO BIT MAP MODE     *
 900 REM ******************************
9994 REM
```

```
 9995 REM ******************************
 9996 REM * M/C TO INTIALISE HIRES MEM*
 9997 REM * + COLOUR TO BLACK ON WHITE*
 9998 REM ******************************
10000 DATA A9,00:    :REM    LDA #$00
10010 DATA 85,BB:    :REM    STA $BB
10020 DATA A9,60:    :REM    LDA #$60
10030 DATA 85,BC:    :REM    STA $BC
10040 DATA A0,00:    :REM    LDY #$00
10050 DATA A9,00:    :REM L2 LDA #$00
10060 DATA 91,BB:    :REM L1 STA ($BB),Y
10070 DATA C8:       :REM    INY
10080 DATA D0,FB:    :REM    BNE L1
10090 DATA E6,BC:    :REM    INC $BC
10100 DATA A5,BC:    :REM    LDA $BC
10110 DATA C9,80:    :REM    CMP #$80
10120 DATA D0,F1:    :REM    BNE L2
10124 REM *****************************
10125 REM * THAT SET $6000-$7FFF TO 0*
10126 REM * I.E. CLEARED HIRES AREA  *
10127 REM *****************************
10130 DATA A9,40:    :REM    LDA #$40
10140 DATA 85,BC:    :REM    STA $BC
10150 DATA A9,01:    :REM L4 LDA #$01
10160 DATA 91,BB:    :REM L3 STA ($BB),Y
10170 DATA C8:       :REM    INY
10180 DATA D0,FB:    :REM    BNE L3
10190 DATA E6,BC:    :REM    INC $BC
10200 DATA A5,BC:    :REM    LDA $BC
10210 DATA C9,44:    :REM    CMP #$44
10220 DATA D0,F1:    :REM    BNE L4
10230 DATA 60:       :REM    RTS
10234 REM *****************************
10235 REM * THAT SET $4000-$43FF TO  *
10236 REM * $01. NYB 0 (1) GIVES     *
10237 REM * BACKGROUND COLOUR (WHITE)*
10238 REM * NYB 1 (0) GIVES POINT    *
10239 REM * COLOUR (BLACK)           *
10240 REM *****************************
10241 REM (c)1983 ICPUG
```

```
19995 REM *****************************
19996 REM * M/C TO EVAL X,Y COORDS + *
19997 REM * PLOT A POINT AT X,Y ON   *
19998 REM *       HIRES  AREA        *
19999 REM *****************************
20000 DATA 20,79,00::REM    JSR $0079
20010 DATA 20,FD,AE::REM    JSR $AEFD
20020 DATA 20,8A,AD::REM    JSR $AD8A
20030 DATA 20,F7,B7::REM    JSR $B7F7
20040 DATA 84,B0:   :REM    STY $B0
20050 DATA 85,B1:   :REM    STA $B1
20051 REM *****************************
20052 REM * THAT GETS A ',' AND      *
20053 REM * EVALUATES THE FOLLOWING  *
20054 REM * EXPRESSION BEFORE TURNING*
20055 REM * THE RESULT INTO AN       *
20056 REM * INTEGER AND STORING. THIS*
20057 REM * IS THE 'X' COORDINATE    *
20058 REM *****************************
20060 DATA 20,79,00::REM    JSR $0079
20070 DATA 20,FD,AE::REM    JSR $AEFD
20080 DATA 20,8A,AD::REM    JSR $AD8A
20090 DATA 20,F7,B7::REM    JSR $B7F7
20100 DATA 84,BB:   :REM    STY $BB
20110 DATA 85,BC:   :REM    STA $BC
20111 REM *****************************
20112 REM * THAT GETS A ',' AND      *
20113 REM * EVALUATES THE FOLLOWING  *
20114 REM * EXPRESSION BEFORE TURNING*
20115 REM * THE RESULT INTO AN       *
20116 REM * INTEGER AND STORING. THIS*
20117 REM * IS THE 'Y' COORDINATE    *
20118 REM *****************************
20119 REM
20120 REM *****************************
20121 REM * TO SET A POINT AT X,Y    *
20122 REM * WITH DISPLAY BASE START  *
20123 REM * AT $6000 ONE MUST 'SET'  *
20124 REM * THE BIT GIVEN BY:        *
20125 REM * 7-(X-8*INT(X/8))         *
20126 REM * IN THE MEMORY LOCATION   *
20127 REM * GIVEN BY:                *
```

```
20128 REM * 6*4096+Y-8*INT(Y/8)+     *
20129 REM * +40*8*INT(Y/8)+8*INT(X/8)*
20130 REM ****************************
20131 REM    (c)1983 ICPUG
20138 DATA A9,60:   :REM    LDA #$60
20139 DATA 85,8C:   :REM    STA $8C
20140 REM ****************************
20141 REM * MEM.POINTER HIGH NOW SET *
20142 REM * TO 6*4096                *
20143 REM ****************************
20149 DATA A5,BB:   :REM    LDA $BB
20150 DATA 29,07:   :REM    AND #$07
20160 DATA 85,8B:   :REM    STA $8B
20161 REM ****************************
20162 REM * MEM.POINTER LOW NOW SET  *
20163 REM * TO Y-8*INT(Y/8)          *
20164 REM ****************************
20170 DATA 45,BB:   :REM    EOR $BB
20171 REM ****************************
20172 REM * THAT GIVES 8*INT(Y/8)    *
20173 REM ****************************
20180 DATA A2,03:   :REM    LDX #$03
20190 DATA 0A:      :REM L5 ASL A
20200 DATA 26,BC:   :REM    ROL $BC
20210 DATA CA:      :REM    DEX
20220 DATA D0,FA:   :REM    BNE L5
20230 DATA 85,BB:   :REM    STA $BB
20231 REM ****************************
20232 REM * THAT GIVES 8*8*INT(Y/8)  *
20233 REM ****************************
20240 DATA 65,8B:   :REM    ADC $8B
20250 DATA 85,8B:   :REM    STA $8B
20260 DATA A5,BC:   :REM    LDA $BC
20270 DATA 65,8C:   :REM    ADC $8C
20280 DATA 85,8C:   :REM    STA $8C
20281 REM ****************************
20282 REM * BY ADDING TO MEM.POINTER *
20283 REM * THAT GIVES A MEM.POINTER *
20284 REM * OF 6*4096+Y-8*INT(Y/8)+  *
20285 REM * +8*8*INT(Y/8)            *
20286 REM ****************************
```

```
20290 DATA A5,BB:    :REM    LDA $BB
20300 DATA A2,02:    :REM    LDX #$02
20310 DATA 0A:       :REM L6 ASL A
20320 DATA 26,BC:    :REM    ROL $BC
20330 DATA CA:       :REM    DEX
20340 DATA D0,FA:    :REM    BNE L6
20341 REM ******************************
20342 REM * THAT GIVES 8*8*INT(Y/8)*4*
20343 REM * =32*8*INT(Y/8)            *
20344 REM ******************************
20350 DATA 65,8B:    :REM    ADC $8B
20360 DATA 85,8B:    :REM    STA $8B
20370 DATA A5,BC:    :REM    LDA $BC
20380 DATA 65,8C:    :REM    ADC $8C
20390 DATA 85,8C:    :REM    STA $8C
20391 REM ******************************
20392 REM * BY ADDING TO MEM.POINTER *
20393 REM * THAT GIVES A MEM.POINTER *
20394 REM * OF 6*4096+Y-8*INT(Y/8)+  *
20395 REM * +40*8*INT(Y/8)           *
20396 REM ******************************
20400 DATA A5,B0:    :REM    LDA $B0
20410 DATA 29,07:    :REM    AND #$07
20420 DATA AA:       :REM    TAX
20421 REM ******************************
20422 REM * THAT GAVE X-8*INT(X/8)   *
20423 REM * AND STORED IT TEMPORARILY*
20424 REM ******************************
20430 DATA 45,B0:    :REM    EOR $B0
20431 REM ******************************
20432 REM * THAT GAVE 8*INT(X/8)     *
20433 REM ******************************
20440 DATA 65,8B:    :REM    ADC $8B
20450 DATA 85,8B:    :REM    STA $8B
20460 DATA A5,B1:    :REM    LDA $B1
20470 DATA 65,8C:    :REM    ADC $8C
20480 DATA 85,8C:    :REM    STA $8C
20481 REM ******************************
20482 REM * BY ADDING TO MEM.POINTER *
20483 REM * THAT GIVES A MEM.POINTER *
20484 REM * OF 6*4096+Y-8*INT(Y/8)+  *
20485 REM * +40*8*INT(Y/8)+8*INT(X/8)*
20486 REM ******************************
```

```
20490 DATA E8:       :REM    INX
20500 DATA 38:       :REM    SEC
20510 DATA A9,00:    :REM    LDA #$00
20511 REM ***************************
20512 REM * GET READY TO SET BIT    *
20513 REM * POSITION                *
20514 REM ***************************
20520 DATA 6A:       :REM L7 ROR A
20530 DATA CA:       :REM    DEX
20540 DATA D0,FC:    :REM    BNE L7
20541 REM ***************************
20542 REM * ACCUMULATOR NOW HAS BIT *
20543 REM * 7-(X-8*INT(X/8)) SET    *
20544 REM ***************************
20550 DATA 01,8B:    :REM    ORA ($8B,X)
20560 DATA 81,8B:    :REM    STA ($8B,X)
20561 REM ***************************
20562 REM * THE NEEDED MEM. LOCATION *
20563 REM * HAS 7-(X-8*INT(X/8)) SET *
20564 REM ***************************
20570 DATA 60:       :REM    RTS
```

--o0o--

MEMBERS' INSURANCE

Members are now covered by insurance when conveying their CBM equipment to and from ICPUG Club meetings. The total sum insured at any one time is £ 10,000. Copies of the insurance policy are available to regional group secretaries (please apply to the Discounts Officer, John Bickerstaff, 45, Brookscroft, Linton Glade, Croydon, CR0 9NA).

--o0o--

THE COMMODORE SHOW

The dates for this year's Commodore Computer Show are June 9th to 11th at the Cunard Hotel, Hammersmith, London. ICPUG will have a presence there and volunteers to help on the stand will be required - we shall also require an organiser to co-ordinate the tasks.

--o0o--

TECHNICAL TIPS

Using the 1540 with the 64.

Numbers appear everywhere these days - what the above title means is 'How to get Commodore's latest home computer - the Commodore 64 - to work with the disk drives supplied for their earlier release (the Vic-20) these drives being known as the 1540 type' but the editor would have complained about such verbosity!

There are slight timing differences between the Vic-20 and C-64 machines which means that the Vic disk drives won't work with the C-64 UNLESS one of the following solutions is adopted :

1) Change the disk drive ROM for one that turns the 1540 into a 1541 (see below) this is highly recommended for all 64/1540 users. Your dealer should be able to help you out with cost/availability an fitting of this chip.

2) A simple poke, inside the C-64, both before and after all disk accesses cures the timing problem. This is POKE53265,11 before access and POKE53265,27 after. The first poke switches off the screen display (a tip that can be used in other programs) which has the slight disadvantage that you cannot see the second poke numbers as you type them in, nor see the 'READY' after your prog has loaded. Never mind, relying on the drive light to tell you the program has loaded cures the second problem. If the screen re-appears after typing the pokes, well done it was correct, else try again. This method is only recommended for C-64 users temporarily borrowing a Vic-20 owner's drive!

The 1541 drive.

More numbers!! This title means 'Commodore realise the difficulties encountered with Vic-20 and C-64 computers using the Vic-20 drive (see above) and in future will manufacture a drive that is compatible with both machines with the user not being required to worry about ROMs and things like that - this drive is called the 1541 and it will have a label on the front to let you know about it'.

Now what was I going to say, I've forgotten!

C2N and C-64.

Honestly it's not my fault that all these numbers keep coming up - whatever happened to the old days when computers were called things like PET, **PPL*, T*ND*, etc!! The C2N is the special Commodore cassette unit used by the complete range of CBM machines. C-64 users will be aware of what happens during cassette activity but for the rest of us PET/Vic users a reference is essential. The screen of the C-64 goes blank during cassette activity of any kind, this is to allow the processor chip full access to the incoming bits of information, without being halted by the video controller chip.

When loading from tape, the screen goes blank, then the 'FOUND' message is displayed and after a pause, the screen goes blank whilst the program is loaded. This pause during display of the 'FOUND' message can be shortened by pressing the Commodore key. Pet programs can be loaded, and run (if they have been written in BASIC), although the Commodore key has to be pressed after the found notice is displayed.

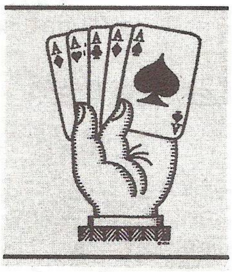
Unfortunately the C-64 will not load Vic-20 progs from cassette - although the best minds of ICPUG are working on the problem (Mike Todd said he'd have a look).

If you have access to a PET, and can load programs from your Vic to the PET - using toolkits JOIN, or something similar, they can be re-SAVEd for loading into the C-64. There is absolutely no problem (fingers crossed) using PET/Vic diskettes once you have a 1541 drive.

CBM 8024 Printer Bug.

Hmmm! what's this I wonder? Answer - a Mannesman Tally printer with a tendency to stagger printing leftwards down a page instead of vertically. You can test a printer easily to check if it is OK with this little program:

```
10 OPEN4,4
20 FOR I=1 TO 10
30 PRINT#4,SPC(50)"********************"
40 PRINT#4,SPC(50)"************************"
50 NEXT
60 PRINT#4:CLOSE4
```



If you've got a nice neat left hand edge then everything's OK. If not, your dealer can help you with a replacement ROM (unfortunately this ROM doesn't have a pound sign, replaced by a dollar, or a left arrow (replaced by a rectangular blob). Dont worry about the ragged right edge!!).

T.C.

--oOo--

ROUND THE REGIONS

The launch of the North Gloucestershire region at the micro exhibition at the Queen's Hotel Cheltenham on Dec 14th was very successful and a number of useful contacts were made. The Cheltenham ladies college extended an invitation for the monthly meetings to be held there on the last Thursday of the month at 7.30pm. Members should use the Archway entrance in St. George's Road. Meetings start at 7.30 and close at 10 pm. The first meeting was on Jan 27th and members were very warmly greeted (with glasses of sherry) by Mjr. T.I.Hardie and four other members of staff from the mathematics and geography disciplines. Mjr. Hardie is in charge of computer studies at the college. Sixteen other members and potential members came along and these included 2001, 3032, 4032, 8032 and Vic users. After an opening welcome by Robin Harvey (branch organizer), we started to get to know each other and explore the many facilities placed at our disposal. It is hoped to start a group newsletter, and a questionaire will ask group members to offer ideas and outline problems for solving. There will be no meetings Mar-July-Aug as the college is not in session then. Dates for the diary are:-

28th April: 26th May: 23rd June: 7.30 Cheltenham ladies college. Enquiries to Robin Harvey - branch organizer Cheltenham (0242) 27588, or Alison Schofield - branch secretary, Cheltenham (0242) 580789.

The Chelmsford regional group has been recently formed and meets on the first and third Tuesday of each month at 7p.m. Interest is mostly Vic-20 and further details can be obtained from Tony Surridge, 97, Shelley Road, Chelmsford, Essex, CM2 6ES. Tel: (0245) 81878 (evenings).

Another new regional meeting place is in the West Midlands. Joe Bowman tells me that at their first meeting about 17 people attended. Already they have a tape library and circulate a local newsletter. Details of meetings from Joe at 6, the Oval, Albrighton, Wolverhampton, West Midlands.

The South West Scotland region caters for local Vic-20 and C-64 users and hold meetings on the first and third Thursdays of each month. The venue is Symington Primary School, Brewlands Road, Symington, at 7p.m.

Watford region were treated to a demonstration of Superspell, described p88 January issue and reviewed in this, and have arranged for a demonstration of the new database program for the C-64 by Tom Cranstoun and Simon Tramner. A future meeting is to have Robin Bradbeer speak on the 'next ten years in computing'. Contact Stephen Rabagliatti for further details.

Please note the change of address for Mrs. Janet Rich: Rose Cottage, 20, Old Court, Springhill, Cam, Gloucester, GL11 5TF. Tel: Dursley (0453) 47708.

New region: Contact Alan Morris, 30, Kelso Road, Bury St. Edmunds, Suffolk, IP33 2EL. Tel: (0284) 61870.

The centrespread shows the current regional list and if any errors or omissions are noted, please inform the Regional Co-ordinator.

--oOo--

INDEPENDENT
COMMODORE
PRODUCTS
USER
GROUP

REGIONAL CO-ORDINATOR
Terry Devereux
32 Windmill Lane
Southall
Middlesex UB2 4ND
Telephone: 01 574 2709

The following list of Regional Groups is kept as up-to-date as is possible. However, apologies are made for mis-information, errors and general hic-cups. Any corrections, queries, or offers of help in any of the areas listed, should be sent, together with all new information to Terry at the address indicated above. Any address list is immediately obsolete directly it's printed!!

BERKSHIRE

RG078
Brian Jones
Dept. of Mathematics and Computing
Slough College of Higher Education
Wellington Street
Slough
Berkshire
Slough: 34585 ext:81

DURHAM

RG071
Jim Cocallis
20 Worcester Road
Newton Hall Estate
Durham
* Meets at Lawson School Birtley *

GLOUCESTERSHIRE

RG080
Mrs Alison Schofield
78 Hesters Way Road
Cheltenham,
Gloucester, GL51 OR2
580789
..or Robin Harvey on..
27588

RG088
Mrs Janet Rich
Rose Cottage
20 Old Court
Springhill
Cam
Gloucester GL11 5TF
Dursley 0453 47708
* Holds informal meetings on the last Friday each month at the above venue *

HAMPSHIRE

RG069
South Hants Regional Group
Tony Cooke
7 Russell Way
Petersfield, Hants GU31 4LD

RG093
North Hants Regional Group
Ron Geere
109 York Road
Farnborough
Hants GU14 6NO
* Meets on the 3rd Wednesday each month at 70 Reading Rd. Farnborough, Hants *

NORTHAMPTONSHIRE

RG098
Peter Ashby
215 Lincoln Way
Corby
Northamptonshire
Corby: 05363 4442

SUFFOLK

RG073
Alan Morris
30 Kelso Road
Bury St Edmunds
Suffolk
Bury St.Eds 61870

DERBYSHIRE

RG079
Raymond Davies
105 Normanton Road
Derby
Derbyshire
0332 41025
* Meets on the 1st Monday each month at Davidson Richards Ltd 14 Duffield Road, Derby *

ESSEX

RG090
A G Surridge
97 Shelley Road
Chelmsford
Essex CM2 6ES

RG087
Walter Green
151 The Hatherley
Basildon, Essex

RG094
Carol Taylor
101 Courtland Avenue
Cranbrook
Ilford, Essex
* Meets at the Grange Remedial, Woodman Path, Hanault, Essex at 20.00hrs *

NORTHERN IRELAND

RG101
David Bolton
19 Carrickburn Road
Carrickfergus
Co Antrim BT38 7ND
Northern Ireland

DEVON

RG107
Matthew Stibbe
The Lawn
Lower Woodfield Road
Torquay, Devon
* No details received *

LONDON

RG097
Barry Miles
Dept of Business Studies
Polytechnic of North London
Holloway Road, London N7
* PET Group of the ACC which meets every other Tuesday at the Polytechnic at 18.00 hrs

MIDDLESEX

RG084
Geoff Squibb
108 Teddington Park Road
Teddington, Middlesex
01 997 2346

NORTHUMBERLAND

RG075
** Proposed Group **
Graham J Saunders
Starling House
22 Front Street
Guidepost, Northumberland

YORKSHIRE

PG070
Bob Wood
13 Bowland Crescent
Ward Green
Barnsley
South Yorkshire
0246 811585

KENT

South East Regional Group
Wng Cdr Mick Ryan
164 Chesterfield Drive
Riverhead
Sevenoaks, Kent
* Meets on the 3rd & 4th Thursday each month at Charles Darwin School, Jail Lane, Biggin Hill at 19.30 hrs *

RG082
Ron Moseley
Rosemont
Lord Romney Hill
Weavering
Maidstone
Kent
0622 37643
* Meets on the 1st Wednesday each month at 1945 hrs for 2000 hrs *

MIDLANDS

RG076
M J Merriman
12 York Street
Stourport-on-Severn
Worcestershire
* Meets on the last Thursday each month at above venue *

RG099
Wolverhampton & Telford District VIC Users' Group
Joe Bowman
6 The Oval
Albrighton
West Midlands
* Monthly meetings, free program library, advice on hardware & software, Send a large SAE for full details *

OXFORDSHIRE

RG104
John Temple
'Kibanda'
Rose Bank
Bloxham, Oxon

RG105
Ian Blyth
40 Wilmot Close
Witney
Oxon OX8 7NL
Witney: 5171 Home
Witney: 3671 Work

WARWICKSHIRE

RG086
J A McKain
P.P.I.Limited
177 Lozells Road
Birmingham
021 544 0202

RG077
Will Light
22 Ivybridge Road
Styvechale, Coventry
Warwickshire
0202 413511

HERTFORDSHIRE

RG092
Herts/Watford Regional Group
Stephen Rabagliati
C/O Institute of Grocery Distribution
Grange Lane
Letchmore Heath
Watford, Herts
* Meets on the second Monday at the above venue *

RG083
Herts/Stevenage Regional Group
Brian Grainger
73 Minehead Way
Stevenage, Herts
Stevenage: 0438 727925
* Informal meetings held on the last Wednesday of each month at The Provident Mutual Assurance, Purwell Lane, Hitchin, Herts *

NORFOLK

RG100
VIC 20 Group
J Blair
7 Beach Road
Cromer
Norfolk
** Trying to get started *

RG103
Peter Petts
Bramley Hale,
Wretton,
Kings Lynn
Norfolk
Stoke Ferry 500692

WALES

RG089
John Poole
6 Ridgeway Close
Connah's Quay
Clwyd CH5 4LZ

RG072
Simon Kniveton
Penpompren Hall
Talybont, Dyfed
Talybont: 303

RG085
F J Townsend
The Hill
Rhydowen
Llandyssul
Dyfed SA44 4QD
05455 5291

LANCASHIRE

RG096
Tony Bond
27 Ince Road
Thornton
Liverpool
L23 4UE
051 924 1505
*Informal Meetings every month at The Merchant Taylor School for Boys, Gt Crosby. 2nd Thursday. *

RG091
David Jowett
197 Victoria Road East
Thornton-Cleveleys
Blackpool
Lancashire
Cleveleys: 86108
* Meets every 3rd Thursday at Arnold School,Blackpool *

RG081
John Ingham
72 Ardwick Street
Burnley
Lancashire, BB10 1BJ
** Wishes to set up a VIC Group *

SCOTLAND

RG102
John Smith
19 Brewlands Road
Symington
Kilmarnock
Ayreshire, KA1 5RW
0563 830407

RG074
Dr Jim BacBrayne
27 Paidmyre Crescent
Newton Mearns
Glasgow
041 639 5696

RG106
A J Quin
Dept of Environment Studies
Glasgow College of Building and Printing
60 North Hanover Street
Glasgow
G1 2BP
041 332 9969

RG107
John Shankland
2 Strathdoon Place
Ayr
Strathclyde

FORTH COLUMN

I mentioned in the review of Supersoft's PET-Forth from AB Computers that certain user variables were missing and they are in fact essential to the definition in Brodie's book to do a non-destructive stack print. DEPTH returns the number of 16-bit elements (or cells in Forth terminology) on the stack and is defined thus:

```
: DEPTH ( -- n ) SP@ 136 SWAP - 2 / ;
```

The stack may then examined with .S using the definition:

```
: .S CR DEPTH IF SP@ 2 - 134 DO I @ . -2 +LOOP
              ELSE ." STACK EMPTY" THEN ;
```

DEPTH is a required word in the '79-Standard' extension and is included with the Supersoft package.

Several useful Forth screens have been sent to me by Frank Chambers and I reproduce them here in shortened form for your use. Frank uses FullForth+ from IDPC Co. and his printer is an Epson. Some of his definitions may need revision for specific implementations.

```
: P-ON  P-ON 27 EMIT 82 EMIT 0 EMIT ; ( US Character set )
```
Redefine the FullFORTH+ printer-on command to set the US character set on the Epson printer, EMPH also sets emphasised print:

```
: EMPH    P-ON 27 EMIT 197 EMIT 27 EMIT 82 EMIT 0 EMIT ;
          ( US + emphasis )
: ARRAY <BUILDS 1+ DUP  +  DUP HERE OVER ERASE ALLOT DOES>
SWAP DUP + + ; ( #CELLS ARRAY NAME) ( CELL# NAME -- ADDR )
( init to 0. Cell# 0 is available)
[ : ERASE     0 FILL ; if you don't have it ]
```
The following are convenient definitions for cursor controls:
```
: UP    145 EMIT ; : DN    17 EMIT ;
: LT    157 EMIT ; : RT    29 EMIT ;
: CLR   147 EMIT ; : HOM   19 EMIT ;
: OFF   146 EMIT ; : RVS   18 EMIT ;
: LF     10 EMIT ; ( for Epson )
```

```
 : DNS    0 MAX -DUP IF 0 DO DN LOOP ENDIF ;
 : UPS    0 MAX -DUP IF 0 DO UP LOOP ENDIF ;
 : LTS    0 MAX -DUP IF 0 DO LT LOOP ENDIF ;
 : RTS    0 MAX -DUP IF 0 DO RT LOOP ENDIF ;
 : XY    ( X, Y -- )    HOM DNS RTS ;
```

RANDOM from May Practical Computing, generates a pseudo-random number suitable for gamesand the like, and CHOOSE limits the range to lie between 0 and the input number.

```
   ( Random number gen from PC 5/82 )
             0 VARIABLE RND HERE RND !
 : RANDOM    RND @ 31421 * 6927 + DUP RND ! ;
 : CHOOSE    RANDOM U* SWAP DROP ; ( U1 -- 0 <= U2 < U1 )
             ( PRINTS 2 SEQ. SCREENS SIDE BY SIDE )
```

PLIST prints two sequential screens side by side, saving much paper and trees, and DPLIST is the same thing with double-strike.

```
 : PLIST ( SCR# -- )    P-ON SCR !
      27 EMIT 82 EMIT 0 EMIT ( U.S. CHARS.) 15 EMIT
                       ( CONDENSED CHARS)
      25 0 DO 2 SPACES I 2 .R 3 SPACES I SCR @ (LINE) TYPE
      22 SPACES I 2 .R 3 SPACES I SCR @ 1+ (LINE) TYPE
   CR LF LOOP 18 EMIT ( CONDENSED OFF ) LF LF LF LF P-OFF ;

 ( PRINTS 2 SEQ. SCREENS SIDE BY SIDE WITH DOUBLE STRIKE )
 : DPLIST ( SCR# -- )    P-ON SCR !
              27 EMIT 82 EMIT 0 EMIT ( U.S. CHARS )
              27 EMIT 199 EMIT ( DOUBLE STRIKE ) 15 EMIT
              ( CONDENSED CHARS)
              25 0 DO 2 SPACES
              I 2 .R 3 SPACES I SCR @ (LINE) TYPE 22 SPACES
              I 2 .R 3 SPACES I SCR @ (LINE) TYPE
              CR LF LOOP 18 EMIT ( CONDENSED OFF ) 27 EMIT
              200 EMIT ( DOUBLE STRIKE OFF )
              LF LF LF LF P-OFF ;
```

SLIST allows listing to the screen without line numbers and prevents unwanted scrolling off the screen of the top of the text. [Normally Forth uses a 'screen' of 16 lines of 64 characters. On an 80-column screen, no problem, but for 40-columns... Presumably FullFORTH+ uses a 40 x 25 screen - Ed].

```
    : SLIST ( SCR# -- )    CLR BLOCK
              -31768 -32768 DO DUP C@ 191 AND I C! 1+ LOOP
              DROP DN 12 RTS RVS ." LIST " ;
```

SAVE and RESTORE are screens by IDPC Co. which allow one to save the compiled versions of all words added to the dictionary on disk blocks and to restore them without having to LOAD and recompile from disk. The following will tell how many blocks will be required before actually SAVEing:

```
            HERE FENCE @ - 8 + B/BUF / 1+ CR . CR
```

SAVE takes the number of the first block where it is desired to record, and RESTORE requires the same number.

```
 ( DICTIONARY SAVE 26/6/82 )
 : SAVE    FENCE @ 8 - SWAP DUP BLOCK HERE OVER !
     CURRENT @ OVER 2+ ! CONTEXT @ OVER 4 + !
  ' FORTH 4 + @ OVER 6 + ! FENCE @ SWAP 8 + B/BUF 8 - CMOVE
    UPDATE 1+ HERE FENCE @ - 8 + B/BUF /
     -DUP IF OVER + SWAP DO B/BUF + DUP I BLOCK B/BUF CMOVE
    UPDATE LOOP ELSE DROP ENDIF DROP FLUSH ;


 ( DICTIONARY RESTORE 26/6/82 )
                 HEX FIRST 400 - DP !
 : RESTORE    FENCE @ 8 - SWAP DUP BLOCK DUP 6 + @ ' FORTH
    4 + ! DUP @ DP ! DUP 2+ @ CURRENT ! DUP 4 + @ CONTEXT !
  8 + FENCE @ B/BUF 8 - CMOVE 1+ HERE FENCE @ - 8 + B/BUF /
    -DUP IF OVER + SWAP DO B/BUF + I BLOCK OVER B/BUF CMOVE
          LOOP      ELSE DROP ENDIF DROP ;
          DECIMAL
```

Finally, the British Computer Society has a computer competition for schools which includes a special prize for software written in Forth. Special prizes are being offered by various companies and include £ 4,000 worth of goodies from Commodore.

R.D.G.

--oOo--

THE HARJIN 80-COLUMN CONVERSION FOR THE 4000-SERIES PET

By Brian Atherton.

A couple of months ago I placed an advertisement in the ICPUG magazine for some Vic bits and pieces I had for sale. I had recently sold my Vic-20 and bought a 4016 PET with a 12" screen which I had upgraded to a 4032. As a result of the advert I did not sell anything, but a number of ICPUG members telephoned to enquire and I soon found myself a member of a grapevine exchanging software, useful tips and ideas. This has proved to be a most worthwhile feature of my ICPUG membership and I've certainly learned a lot. (Incidently, any member who would like to telephone me is most welcome to do so. I can be contacted on Doncaster (0302) 539142).

One of the fellow ICPUG members with whom I consequently made contact was Robin Harvey. He has recently designed and marketed through 'Harjin Services' a board which converts a 4000-series machine with a 12" screen into 80-column. I regularly use both an 8032 and 8096 machine at work and the prospect of being able to run 80-column software on my 4032 was of course attractive. My reasons for buying a 40-column machine in the first place were both cost and the wish to run 40-column software, especially games for the children. I was able to let Robin have my 4032 pcb one day and it was returned the following day fully tested as a switchable 40/80-column device. I have since spoken to other ICPUG members who have had this conversion and they are as delighted as me with the effectiveness.

The conversion consists of a small neat board which plugs into the 'E' ROM socket which it replaces. The board is professionally designed and uses through-plating on its double-sided design. A number of other chips are fitted and a number of connections are hard-wired. The conversion does not interfere with the machine's PCB tracks and uses existing links. A single switch on the new board converts from 40- to 80-column screen. There is no need to switch-off and the system automatically resets when switched over. If preferred the switch can be brought out and mounted on the PET case, but I have not found this necessary. As required, the modification can be supplied such that on power-up the PET is in 'graphics' or 'business' mode in both 40- and 80-column operation or in different mode for each screen size. The choice is left to the user to specify when ordering.

There are no alterations to the keyboard, so it must be made clear that this conversion permits a 4000-series PET to operate with an 80-column screen. This means three keys on the 8000-series 'business' keyboard are not implemented. These are 'Tab', 'Escape' and 'Repeat'. I have found this presents no real difficulty with the software I use and the 80-column proprietary software I have tried (including Visicalc and Wordpro 3) run as normal. Under software control 'TAB' and 'ESC' can be achieved with CHR$(9) and CHR$(27) respectively, and CHR$(137) will as expected function as a tab set.

All considered, this is a device I can recommend to other members. A 24-hour turnaround is normally achieved and anyone interested should contact Robin Harvey, HARJIN Services, 30, Wimborne Close, Cheltenham, Glos. Tele (0242) 27588. The inclusive charge for the switchable conversion is £ 90.00 and a permanent upgrade from 40- to 80-column is offered at £ 50.00. Also offered is 4016 to 4032 upgrading at £ 30.00 if this is done at the same time.

--oOo--

DISK FILE - SECTOR 5

(c)ICPUG 1983 By Mike Todd

Owing to an injury sustained whilst trying to keep fit (i.e. playing squash!) I've been unable to get as much work done as I'd wanted to and so this edition of DISK FILE is more of a short chat about odds and ends.

Firstly, I want to take up the recently publicised statement that the "@-replace" bug doesn't exist!

This "bug" was reported by Commodore in the days of DOS1 and it appeared as a corruption of a program or data-file which became inexplicably linked to another file.

The cause was explained by Commodore as being the result of a bug in the "@-replace" section of the DOS. In fact, the DOS source code even has a note pointing out were the bug might be!

It appeared that DOS happily allocated blocks for the new version of the file until it ran out of space on the disk. When this occurred, the operation aborted, but not without the last block of the new version having its link set to point to a block already in use. For some reason, this new version became accessible and the final link "hooked" into a different file.

Although this was only speculation as to the cause, many corrupted disks seemed to conform to this scenario. Unfortunately, it seems to be very difficult to reliably reproduce the problem.

Obviously, it is possible that the false linking may not be spotted for some time and the corruption may not always be associated with an "@-replace" operation having taken place.

It may be that DOS1 is the main culprit, although Commodore insist that the bug exists on all their floppy DOS versions!

So what is the truth? Does the bug exist or does it not exist? I would say that I have been using "@-replace" on DOS2.1 for many months now with, apparently, no sign of problems, although I have had the odd problem with files becoming inter-linked. Was this due to the "@-replace" bug?

There are several points in the "@-replace" operation which could harbour a bug, and so here's a brief description of how the "@-replace" function actually works.

(1) When you OPEN a file (which also occurs when you use SAVE), the DOS checks to see if the file already exists in the directory. If it does, a check is made to see if the file-name started with "@". If it does, the "@-replace" procedure is implemented.

(2) First, the file type code in the directory entry has bit 5 set to a 1. This normally has bit 7=1 to indicate a valid file and bits 0-2 have a code indicating the file type.

If the entire file type is zero, then the file is considered to be deleted, but if bit 7 is zero and there is a type code in bits 0-2 then the file has not been closed correctly, and this is shown by an asterisk appearing by the file type in the directory.

Bit 6 of this file code is used as a file locking bit, and any file with bit 6=1 cannot be scratched and the directory entry appears with a "<" following the file type. Not many people know this!! This doesn't stop an "@-replace" or the file being deleted if the disk is re-headered. It isn't implemented on DOS1 and is not really of much use.

Anyway, back to the plot!

(3) The DOS now takes the starting track/sector pointer of the new file and stores it in a safe place within the directory entry. It then procedes to handle the file in the normal way, allocating blocks and storing data.

(4) Only when the entire contents of the new version of the file is safely on disk, and the file is closed does the new starting track/sector pointer replace the old one and its temporary storage location in the directory is wiped. The old file is then read through block by block and each block is de-allocated within the BAM.

If the new file was not correctly written to disk, either because of a full disk or a write error, the old track/sector pointer is left alone. By preserving the original entry until the new version is successfully on disk, any failure to write the new version correctly will not result in the complete file being wrecked. At least the original is still left.

If there is an error, the DOS leaves bit 5 of the file type and the replacement track/sector pointer in the directory. It shouldn't; because, if for some obscure reason (I can't think of one off hand!) the close routine operates again on that file, the flag in the file type could force the replacement track/sector pointer to replace the old pointer. Future accesses to that file will therefore access the replacement file which is not complete, and will link into existing data on the disk.

This is pure speculation and arises from examining the DOS2.1 ROMs. Looking at DOS1, there are even more problems likely since the "@-replace" is not flagged explicitly. Instead, if the close routine is asked to close any file which has already been closed (i.e. bit 7=1) it is assumed that this file was an "@-replace" file.

The track/sector pointer still exists in the directory entry, and so the old and new pointers are swapped with the same effect as already described.

I would guess that the problem with DOS1 is more likely than the DOS2.1 problem occurring. I've not really had the time to investigate the problem fully on DOS, but experiments with DOS2.1 cannot reproduce the problem.

"@" SUMMARY

I've only postulated some ideas, and would welcome any comment from members who may be able to offer concrete information on this bug. If the bug does exist, I would guess it's tied up with the erroneous closing of files by the DOS.

Those who handle their files with care, making sure that there's always enough space on a disk, that files never get left open, that aborts during writing files are always handled correctly, should have few problems and to them, the bug should not show itself.

Of course there's always the possibility that the bug doesn't exist - I very much doubt it though - and it can only be assumed not to exist when that fact is proven beyond all possible doubt. Until then, take care!

HOW MANY BLOCKS LEFT?

In handling disks, it's often useful to know how many blocks are left on a disk.

Unfortunately, the DOS is often careless in allocating blocks, and we often find it showing less blocks free than we thought; in other words, the total number of blocks allocated to files plus blocks free doesn't always add up to the total disk capacity. If this does happen, I'd suggest that the disk is validated using the "V" command ("COLLECT" on BASIC 4).

If you actually need to know the number of blocks free from within a program, the following short routine will provide the answer. I wouldn't recommend using it whilst other files are open as it can result in the channels being de-allocated in the disk drive!

Line 100 opens a file to the directory and the "......." forces the directory to return with no files actually being listed (unless you've actually got a file called "......."!)

Line 200 skips the preamble to the heading line, and 300 skips the rest of that line. Line 400 skips the first two (link) bytes of the blocks free message and then reads the number of blocks free, in byte format, into A$ and B$.

Line 500 simply converts the two bytes values into a variable. Note the use of A$+CHR$(0) to avoid ASC being asked to take the value of a null string.

```
100  OPEN 2,8,0,"$0:........"
200  GET#2,A$,A$,A$,A$,A$,A$
300  GET#2,A$ : IF A$ <>"" THEN 300
400  GET#2,A$,A$,A$,B$
500  F= ASC(A$+CHR$(0)) + ASC(B$+CHR$(0))*256
600  CLOSE 2
```

FINALLY - SOME NOTES ON THE 2031

I don't know how many members have 2031 (single disk) drives, but these can't be used with BASIC 2. This is because the 2031 fails to respond to the controller within the time limit imposed by BASIC 2.

All's well with BASIC 4, but for BASIC 2 there's a "fixed" $F000 ROM from SUPERSOFT (10-14 Canning Road, Wealdstone, Harrow, Middx; 01-861-1166).

These drives also issue an interface clear (IFC) signal when they are switched on which means that any other devices on the IEEE bus will be reset. Incidentally, this also occurs with the Vic drive, the 1540.

There seem to be several bugs with this drive, including directory corruption when using the BASIC 4 DIRECTORY command (always perform this a second time to prevent problems with the SAVE command) - there's also problems with reading sequential files, for which I gather there's a replacement ROM available to fix the bug.

--oOo--

COMMODORE 64 - CHARACTER MODE DISPLAYS

By Brian Grainger

The purpose of this article is to explain how the VIC chip prints characters on the screen. I'm afraid it tends to get technical in places but I will do my best to explain as much as possible in English! Having understood the article you stand a better chance of being able to keep and swap various character displays as well as being able to define your own character set. Just to add to the facilities (confusion!) there are three character display modes on the 64 - (i) Standard character mode, (ii) Multicolour character mode and (iii) Extended colour mode.

To start let me make some definitions:

VIDEO MATRIX.

The video matrix gives the information to, the VIC on which characters to display on the screen at any time. It consists of 1000 bytes of memory, one for each screen location. The character at screen top left is given by the value in the 1st byte and the character at screen bottom right is given by the value in the 1000th byte. Each byte value is used to point to the location in the CHARACTER BASE which defines the dot pattern for the character to be displayed at the corresponding screen location. As a byte can only have 256 different values it should be clear that a screen display can only show a maximum of 256 different characters.

CHARACTER BASE.

The character base is used to give the VIC the pattern dots to be displayed for each of the 256 defined characters. Each character is displayed as an 8 x 8 dot matrix so 64 bits (8 bytes) are required to define one character. Thus 256*8 bytes (2K) are required to define the 256 characters. The CHARACTER BASE therefore takes up 2K of memory. Each character definition uses 8 consecutive bytes of the CHARACTER BASE. Working from left to right, top to bottom of the character, a dot is indicated by setting the corresponding bit to 1 and no dot is indicated by setting a bit to 0. A capital 'A' is thus given by:

```
---**--- = 00011000 = $18 = 24
--****-- = 00111100 = $3C = 60
-**--**- = 01100110 = $66 = 102
-******- = 01111110 = $7E = 126
-**--**- = 01100110 = $66 = 102          (c)1983 ICPUG
-**--**- = 01100110 = $66 = 102
-**--**- = 01100110 = $66 = 102
-------- = 00000000 = $00 = 0
```

The 8 consecutive bytes for 'A' therefore have the values values: 24, 60, 102, 126, 102, 102, 102, 0
Note to PET people - the dot patterns on the 64 are not the same as on the PET, as the above example shows.

COLOUR MATRIX.

The COLOUR MATRIX is a rather loose description. It is used to give the VIC colour information on certain display modes. It is a loose description because in some modes the colour information comes, not from the COLOUR MATRIX, but the VIC registers or even the VIDEO MATRIX. The COLOUR MATRIX consists of 1K of memory. The first 1000 bytes are used in conjunction with the 1000 bytes of the VIDEO MATRIX and in Standard character mode each byte defines the colour of the corresponding screen character. Since there are only 16 colours, only 4 bits (the lower nybble) are used to define colours.

VIC REGISTERS.

The VIC chip has associated with it 47 bytes of memory which I call the VIC registers. Each register contains different information which the VIC uses. e.g. Sprite (MOB) positions, memory pointers, MOB collision detection bits and colour information. The registers are usually denoted by their number i.e. Registers 0-46 ($00-$2E). When accessing the registers with the microprocessor you must use the microprocessor addresses.

Having introduced all these definitions the next thing to do is to identify where in the memory map these items exist.

VIDEO MATRIX LOCATION.

Each byte of the VIDEO MATRIX has an address which can be defined by a 14-bit address code WHEN VIEWED BY THE VIC. The corresponding microprocessor address is given by this value PLUS the value of the VIC start address (given by the offset in $DD00).

Bits 13-10 of the VIDEO MATRIX VIC address are given by Bits 7-4 respectively of VIC register $18. Bits 9-0 of the VIDEO MATRIX are changed by the VIC itself to access the individual characters in the video matrix.

The VIDEO MATRIX start address is thus given by Bits 7-4 of VIC register $18 followed by 10 '0' bits. On switching the 64 on, bits 7-4 of VIC register $18 are=0001. The default VIDEO MATRIX start is thus:

00 0100 0000 0000 = $0400 (VIC address)

By setting VIC register $18 the programmer can set the start of the VIDEO MATRIX to be any of 16 different values within the VIC address space of 16K.

Lowest VIDEO MATRIX start is $0000
Highest VIDEO MATRIX start is $3C00
Interval between consecutive start points is $0400.

By combining the 16 options within the VIC address space with the 4 bank options for the VIC the VIDEO MATRIX can be located anywhere in the 64K except for the character generator ROM areas.

Note:- The VIDEO MATRIX requires only 1000 bytes for the screen yet consists of 1024 bytes. The last 24 bytes are used for Sprite pointers.

CHARACTER BASE LOCATION.

Each byte of the CHARACTER BASE has an address given by a 14-bit code, WHEN VIEWED BY THE VIC. The value of the microprocessor address is given by adding this value to the VIC start address (defined by the offset in $DD00). Bits 13-11 of the CHARACTER BASE VIC address are given in bits 3-1 respectively of VIC register $18. Bits 10-3 vary according to which block of 8 bytes (which make up the character) are being addressed. It is the values of these bits that are held in a byte of the video matrix to define which character to display. Bits 2-0 are controlled by the VIC and define which byte (row of dots) of the character to display.

The CHARACTER BASE start address is thus given by bits 3-1 of VIC register $18 followed by 11 '0' bits. When switching on, the 64 bits 3-1 of VIC register $18 are 010. The default CHARACTER BASE start is thus:

```
01 0000 0000 0000 = $1000
```

by setting Bits 3-1 of VIC register $18 the programmer can modify the CHARACTER BASE start address to one of eight options.

The lowest CHARACTER BASE start address is $0000
The highest CHARACTER BASE start address is $3800
The interval between consecutive start addresses is $0800.

By varying the choice of start address with the bank accessed by the VIC chip, the CHARACTER BASE can be set in any 2K block of the memory. The character generator ROM takes up 4 of these options however. The graphic character set is mapped at $1000-$17FF and $9000-$97FF (microprocessor addresses). The lower case character set is mapped at $1800-$1FFF and $9800-$9FFF (microprocessor addresses). The character ROM cannot be read by the microprocessor at these addresses as the character ROM is only accessible to the VIC. The microprocessor would read the RAM in these areas. It IS possible for the microprocessor to read the character generator at a completely different address but that is another article!

COLOUR MATRIX LOCATION.

The COLOUR MATRIX is fixed in memory and its position cannot be changed. It is located at microprocessor addresses $D800-$DBFF

VIC REGISTERS LOCATION.

The VIC registers are also fixed in memory. The microprocessor addresses them from $D000-$D02E, one byte for each of the 47 registers.

Now that the locations are defined all that remains is to indicate how the VIC works in each of the character modes.

STANDARD CHARACTER MODE.

To set up Standard character mode, bits 5 and 6 of VIC register $11 and bit 4 of VIC register 16 should all be 0. On switch-on the 64 comes up in standard character mode.

In this mode each character on the screen is defined by the values in the VIDEO MATRIX. These values point to the CHARACTER BASE which define the dot patterns to display for each screen character. Where a bit 1 is addressed in the CHARACTER BASE the VIC will display a point in the colour defined by the COLOUR MATRIX byte corresponding to the character position on the screen. Where a bit 0 is addressed in the CHARACTER BASE the VIC will display a point in the colour defined by bits 3-0 of VIC register $21 (Background colour 0).

MULTICOLOUR CHARACTER MODE.

This is probably the most complicated display mode. It is obtained by setting bits 5 and 6 of VIC register $11 to 0 and bit 4 of VIC register $16 to 1.

Instead of each character being defined as 8 dots across by 8 dots down, it is regarded as 4 dot pairs across by 8 dots down in this mode. The VIDEO MATRIX and CHARACTER

If bit 3 of a COLOUR MATRIX byte is 0 then the corresponding character is displayed exactly as in standard character mode EXCEPT only bytes 2-0 of the COLOUR MATRIX byte are applicable giving access to 8 instead of 16 foreground colours. This allows standard characters to be displayed in multicolour mode.

When bit 3 of the COLOUR MATRIX byte is 1 the CHARACTER BASE information is treated as dot pairs and not individual dots. The colour displayed (over the two dot positions) by the various possible dot pair combinations can be found from the following table:

Dot Pair	Colour Displayed
00	Colour defined by bits 3-0 of VIC register $21
01	Colour defined by bits 3-0 of VIC register $22
10	Colour defined by bits 3-0 of VIC register $23
11	Colour defined by bits 2-0 of the COLOUR MATRIX byte.

Notes:

1) Commodore like to refer to the first two dot pairs as background colours and the latter two dot pairs as foreground colours. I personally find this extremely confusing to define what is foreground and background when 4 possible colours are available! It is important, however, when displaying Sprites in multicolour mode.

2) You are not advised to display characters from the character ROM in multicolour mode (unless it is as a standard character with bit 3 of the COLOUR MATRIX byte set to 0). Strange results will occur if you do as the character ROM defines characters as individual dots, not dot pairs.

EXTENDED COLOUR MODE.

Multicolour mode allowed up to four colours per character to be displayed at the expense of lower character resolution. Extended colour mode allows up to five colours per character to be displayed, this time at the expense of lowering the total number of different characters displayable on one screen.

Extended colour mode is obtained by setting bit 5 of VIC register $11 to 0, bit 6 of VIC register $11 to 1 and bit 4, of VIC register $16 to 0.

In this mode bits 6 and 7 of the value from the VIDEO MATRIX are used to define which colour is displayed when a bit 0 is found in the CHARACTER BASE. This leaves bits 6-0 to define which character is displayed, giving 64 possible characters instead of 256 as in Standard character mode. It should be clear from the use of the VIDEO MATRIX that the first 64 characters defined in the CHARACTER BASE can be displayed. The other 192 cannot.

The CHARACTER BASE and COLOUR MATRIX are used exactly as they are in Standard character mode.

When a bit 1 is found in the CHARACTER BASE the colour displayed is as given in the COLOUR MATRIX byte corresponding to the screen position.

When a bit 0 is found in the CHARACTER BASE, the colour displayed is given by the following table:

VIDEO MATRIX Bit 7	VIDEO MATRIX Bit 6	Colour Displayed
0	0	Colour defined in bits 3-0 of VIC register $21
0	1	Colour defined in bits 3-0 of VIC register $22
1	0	Colour defined in bits 3-0 of VIC register $23
1	1	Colour defined in bits 3-0 of VIC register $24

Thus in this mode although only 64 different characters can be displayed at a time, four background colours and one foreground colour can be used for each character displayed.

Well that sums up all the character display modes. I hope to supply, in a future article, some BASIC statements to set up the various pointers and registers outlined above to the values required. For the moment some idea can be gained by looking at my Hi-res graphic program elsewhere in the Newsletter.

--oOo--

SHADOWFAX - THE COMPUTER WITH BOUNCE

Hello! I'm SHADOWFAX an 8032 Commodore Business Machine. Yesterday was my big day. My friend Quickbeam (a 2001 old-ROM machine) and I were off to the Micro-computer exhibition at the Queen's hotel in Cheltenham Glous. to take part in the launch of the North Gloucestershire branch of ICPUG. My two humans (Ray and Alison) and their pals Robin and Chris, along with several others, were escorting a Fat-40 (what a rude name for so lovely a creature), a Vic-20 and others like Q and I, to the show. Like all humans the world over, they were rather excited. Funny things humans - something called emotion they suffer from.

However to get back to my story, I was the last to be carried to the car and, just four feet short, my human developed a 'SYNTAX ERROR' and before I could cry 'RETURN' from about four foot six up, I hit the concrete pavement with a CCRRAASSSHHH! My whiteplug shattered to BITS. My human (12.5 stone, if an ounce) came down on my head. My bezel slipped over one ear - and there I was covered in blood - keyboard, screen and all!

The frightful indignity was, they were more concerned with my human than me. But Chris, bless his heart, picked me up and when I was back indoors, cleaned my face and chassis. Robin also came to the rescue and once they fitted a new plug and fixed my bezel in place (well how'd you feel with your bezel slipped all over your ear!) I was able to tell them I was OK and READY to go. Mind you if I'd had Speakeasy fitted I'd have said rather more.

Well 'The Show Must Go On' is the Commodore way - so off I went and from 12 noon to 10pm displayed with true Commodore grit. People came to look at this remarkable machine with 'Bounce'. Mind you, I don't mind telling you, I felt a bit shattered. My Spacemaker lost it's cool and my chips thought it was 'frying tonight-time' but I put a brave face on it and carried on regardless. I can't say the same for my poor human. Poor Ray - his nose was as bright as a cherry and his lips and mouth were nastily cut - this skin stuff of theirs is pretty poor material.

I gather the group got off to a flying start and in Club News you can read all about our meetings for the future. We've been really fortunate to find a first class home and already have a great variety of interesting members and potential members. The group should have a lot of bottle. (No not that kind, we are not meeting at a Pub.) It's a jolly good thing my human insured me. They registered me with R.J.Dee, (John to you) Insurance services of Bristol. I'm due for a face lift and thorough medical at Milequip of Hare Lane, Glos. Mr Penny, my protector there, was the one who found me my humans. They aren't a bad couple (my humans) - a bit thick at times, but they mean well.

My regards to all you PETs, Vics and other COMMs around the land. Just remember this - I know that when humans carry you around, well to put it crudely, your chips are in your chassis but fear not - you're tougher than they are. You've got Bounce....!

SHADOWFAX (The Great).

--oOo--

REVIEW

Forth for PET/CBM AB Computers

FORTH for CBM/PET is an implementation of the FIG Forth model with full 79-Standard extensions co-authored by Cargile and Riley. It is distributed in the US by AB Computers (as is Frank Levinson's Visible Music Monitor), and is available in the UK from Supersoft. The disk is accompanied by an 80-page A4 size spiral bound handbook.

The handbook contains an introductory chapter for those new to the Forth language entitled 'If you've never tried Forth...' which by example explains some of the more common features of the language and some of its unique attributes.

The chapter which follows gives information to cope with the differences between Forth's inherent disk operating system and Commodore's in-built DOS. The editor commands and extensions are described, followed by an explanation of additional words specific to this implementation. The assembler words are explained including extensions to the assembler vocabulary.

The Forth language is designed to be highly portable between different machines and as a result its operating system is adequate, but rudimentary. Now it seems a pity to lose some of the features that make the Commodore machines superior to others, so the authors have thoughfully defined words to retain such features as the clock functions, string labels, BASIC file handling and screen editing. Each new word to define these additional features is explained and the chapter concludes with the '79 Standard extensions.

An index to the disk is listed (Forth's equivalent of a directory) followed by a memory map of the dictionary. A system memory map is also included. The handbook concludes with the standard Fig-Forth glossary (a public domain publication).

A page of errata and clarifications cover some of the omissions of detail and correct some errors of fact. There is. however. a fair sprinkling of minor typographical errors.

The disk itself contains the Forth implementation, plus 150 Forth screens (a Forth screen is a 1K block. i.e. four CBM blocks, of source code which is compiled on LOADing.). This disk has been specially 'got at' since the Forth disk operating system does not write into the Commodore operating system's BAM and correctly update the directory. This has been done to protect the program from corruption, but a disk formatted for Forth would not have this feature, so keep your Forth-formatted disks clearly labelled.

While on the subject of the BAM. another side effect is that the feature of DOS 2.0 and later. whereby a disk with a different ID is initialised automatically, is no longer active. Forth's DOS requires a change of disk to be followed by I0 or I1. Additional words are available to enable interchange of screens between 4040 and 8050 drive units.

There are a few items on disk which receive little or no mention in the handbook: these are worth exploring for they include such things as arrays, an 'uncompiler' printer output control words and a couple of Forth demonstration programs. The latter are presumed to be to demonstrate a programming technique, for the programs themselves are of no consequence.

Earlier I briefly mentioned the CBM clock facility. In BASIC one has TI and TI$ as reserved words to handle this, but since Forth is extensible we can really go to town on this. TI simply puts on the stack a double-precision number equivalent to reading the jiffy clock. HMS will take this number and replace it with three numbers equivalent to hours, minutes and seconds in 24-hour format. EMITHMS emits (i.e. prints) these three values in 12 or 24-hour format according to the contents of the variable CLOCK. TIME is a definition embracing the above which combines the above to read the jiffy clock, convert and print it in the appropriate format. CTIME displays an updating clock until the stop key is pressed. The user may wish to redefine this word to put the time at some pre-defined screen position, or use some other exit condition. There are many more definitions to aid time setting, plus two double-precision variables possibly for alarm clock or start-stop use.

If you are interested in longer time periods. there is on disk a sort of universal calendar. This consists of a perpetual calendar display, but with the facility to give time interval between two given dates, or the date given a date and an interval before or after. Careful choice of words (Forth words that is), enables the user to enter a plain english request.

So what are its deficiencies ? Well I did not go to the extent of scrutinising the dictionary to see if the complete required word set was present, but I noticed the absence of the user variables R0 and S0. If they are needed, presumably they could be found and defined accordingly. Similarly in the '79-standard extensions FIND was conspicuous by its absence, but is readily defined from its near relative -FIND.

MARGIN is a user variable added to implement the word INPUT$, the implementation's equivalent to the INPUT command. and MON is added to give access to the machine-code monitor.

Finally. ICPUG have had this gripe before and I won't pursue it in this review, the US prices are $50 for Forth and $30 for the Metacompiler. The UK prices are £ 50 and £ 30 respectively. We are back to that one-to-one exchange rate again. If you consider this a small overhead to pay for a quick-delivery item then you have a value-for-money Forth.

R.D.G.

--oOo--

THE VIC/64 ROM LISTS

By Mike Todd

I've had many requests for information about what goes on inside the ROMs on both the Vic and the 64, and also how their ROMs compare with those of the PET ROM sets. So I've prepared a list of all major ROM routines in the Vic and 64 and on the following 9 pages you'll find descriptions of what's inside the BASIC interpreter on the Vic and 64, with BASIC 2/3 details as a useful reference. I'll be giving lists of the kernal or operating system routines at a later date.

The labels given are, as far as I can ascertain, the ones used by Commodore and where I've been unable to find an "official" label, I've invented my own.

These labels make referencing the routines, and indeed the system variables, very much easier. The system variable labels can be found in the tables in the July '82 Newsletter.

I've also adopted a couple of conventions which also make the preparation of these lists very much easier. For instance A, X and Y always refer to the registers and any reference in brackets - for instance (A), (A/Y) or (TXTPTR) - always refers to, and should be read as, "the contents of".

In the example of (A/Y), generally this would be a pointer with (Y) containing the high order byte and (A) the low. In other words, in standard 6502 format.

Numbers are stored in a variety of forms. As two byte integers, or as five byte floating point numbers, and these floating point numbers appear in two forms. The first is in the form of a packed floating point number which also contains its sign and this is the way they are stored in memory and I've referred to these as MFLPT (memory floating point). When these numbers are prepared for the arithmetic routines, they are "unpacked" into a 5-byte FLPT (floating point) number and a sign byte.

Strings are stored, starting at the upper limit of RAM working downwards, and references to these strings contain a two-byte pointer to the first character and one byte which holds the length of the string. These three bytes are collectively known as a DESCRIPTOR and are the means by which strings are manipulated.

But, just to make life awkward, these descriptors themselves are often accessed by DESCRIPTOR POINTERS which simply point to where the descriptor is stored.

A major difference between the Vic/64 structure and other Commodore ROM sets is the sharp differentiation between the BASIC interpreter and the operating system.

The BASIC interpreter handles all BASIC text, and if necessary passes information to the operating system and accesses the operating system routines via a KERNAL. This is a "jump" table which is always in the same place in different systems and which accesses routines for outputting and inputting data.

Whilst the PET ROMs also have this kernal concept, it wasn't well publicised and such routines as the kernal SAVE and LOAD routines also read part of the BASIC text to set up the necessary parameters. This is contrary to the whole concept of a kernal.

The Vic and 64 both adhere to the concept and set up all necessary parameters from within the interpreter and then access the kernal routines via the jump table.

In addition, to make some of the kernal (and indeed interpreter) routines more flexible, some are "vectored" through "hooks" or "windows" held in RAM. These are simply addresses held in RAM which normally point straight back into the routine that they are a part of. But, by changing them, it is possible to intercept the routines to provide an extra element of flexibility.

Wherever this happens, I've noted it by "Vic/64 through (0300) to C43A". This means that, 0300/0301 contain a pointer to C43A, and in general, the address of the routine accessed in this fashion is one instruction (3 bytes) ahead of the indirect JMP which calls it.

One of the concepts of the kernal input and output routines is that they should always indicate a successful operation by clearing the carry bit of the status register (C=0). If there has been an error, C is set to 1 and the calling routine should identify this and handle the error appropriately.

Although the kernal has its own error handling routine which will generate such messages as "TOO MANY FILES" or "ILLEGAL DEVICE NUMBER", the BASIC error handler BIOERR is used to determine how the interpreter should handle an error condition.

That is why, all accesses to the kernal from BASIC are done through a series of routines from $E109 on the Vic and $E10C on the 64.

You will have noticed that the BASIC interpreter "spills over" into the kernal ROM which starts at $E000. The actual operating system starts at $E4A0 on the Vic and at $E4DA on the 64, and this fact should be remembered if you're switching ROMs in and out!

All numbers referred to in the list are in hex format, and I've omitted the "$" sign for simplicity. If a number is given in decimal (for instance in the tables of constants) these are preceded by a "#".

I've also highlighted the actual "keyword" routines by "***" at the start of the line. These are the routines called immediately a keyword token is encountered in the BASIC text.

These lists are given for information only. It is one of the fundamental concepts behind the kernal jump table that

ALL accesses into the ROMs are done via this table. This way, machine code software becomes portable between different machines or even different versions of the same machine.

It is often tempting to pinch a useful ROM routine to save including the code in your own program, but if Commodore should make changes to the ROMs in later models, the software will have to be re-written unless the kernal table has been used.

It is this concept of a kernal which has contributed to the portability of CP/M software. By having a standard jump table as part of the operating system, it doesn't matter how or where the input and output routines operate, the main program just knows that, calling a specific routine, for instance, will output one character to screen, disk or whatever.

Of course, many aspects of the PET, the Vic and the 64 are very different indeed, but if all input and output within machine code programs were to be handled through this jump table with no calls directly into the ROMs, there is no access directly into the peripheral chips or the screen and differences in screen size were also borne in mind, there's no reason why such programs shouldn't be transportable.

But, human nature being what it is, this rarely happens!

Anyway, and finally, I couldn't possibly include every single ROM entry point. Indeed, there's probably little point unless you really want to take the ROMs to bits (metaphorically of course).

If your "favourite" routines aren't there, or you're trying to convert machine code from one machine to another and need a conversion from one address to another which is not listed, you'll have to wait. ICPUG hope (I said HOPE!) to produce a total comparative ROM list later in the year.

B2/3	VIC	64	LABEL	DESCRIPTION
----	C000	A000		Cold and Warm reset vectors (to INIT & BASSFT) & message - "CBM BASIC"
C000	C00C	A00C	STMDSP	BASIC COMMAND vector table - two byte addresses
C046	C052	A052	FUNDSP	BASIC FUNCTION vector table - two byte addresses
C074	C080	A080	OPTAB	BASIC OPERATORS vector & priority table - two byte address + one priority
C092	C09E	A09E	RESLST	COMMAND keyword table - last char in each word has B7=1
C11D	C129	A129	MSCLST	MISC keyword table - last char in each word has B7=1
C134	C140	A140	OPLIST	OPERATOR keyword table - last char of each has B7=1
C141	C14D	A14D	FUNLST	FUNCTION keyword table - last char in each word has B7=1
C192	C19E	A19E	ERRTAB	Error messages - last char of each message has B7=1
----	C328	A328	ERRPTR	Error message pointers - points to first character of each message in ERRTAB
----	C364	A364	OKK	Messages - "OK" "ERROR" "IN" "READY" "BREAK"
C2AA	C38A	A38A	FNDFOR	Find FOR entry on stack or skip them to find GOSUB entry if called by RETURN
C2D8	C3B8	A3B8	BLTU	Move block of memory up - check sufficient memory then ...
C2DF	C3BF	A3BF	BLTUC	Move memory block between (LOWTR)&(HIGHTR) up to new block ending at (HIGHDS)
				NB: (HIGHTR) and (HIGHDS) must point to byte AFTER each block
C31B	C3FB	A3FB	GETSTK	Check stack for space to accomodate (A)x2 entries - "OUT OF MEMORY" if none
C328	C408	A408	REASON	Check address (A/Y) is lower than bottom of strings - if not then ...
C355	C435	A435	OMERR	Print "OUT OF MEMORY" error
C357	C437	A437	ERROR	Print error message (Vic/64 through (0300) to C43A) then ...
C37E	C469	A469	ERRFIN	Print "ERROR" (or "BREAK" if entered from STPEND) then ...
C389	C474	A474	READY	RESTART of BASIC - output "READY" then ...
C392	C480	A480	MAIN	Input line & identify BASIC line or command (Vic/64 through (0302) to C483)
C3AB	C49C	A480	MAIN1	If BASIC line then get line number and convert keywords in line to tokens
C3B1	C4A2	A4A2	INSLIN	Insert text from BASIC buffer into program - line number in (LINNUM) on entry
				-line must have keywords changed to tokens & (Y)=length of line
				-if (BBUFF)=00 then line will be deleted - routine exits to MAIN
C439	C52A	A52A	FINI	After inserting new line into text - do RUNC, LNKPRG & re-enter at MAIN
C442	C533	A533	LNKPRG	Rebuild BASIC text link pointers
C46F	C560	A560	INLIN	Input line to BASIC input buffer - 00 is placed at end
C481	----	----	RDCHR	Input char to ACC (also inverts flag in 0D to suppress output if char is 0F)
C495	C579	A579	CRUNCH	Change keywords to tokens - line in BBUFF - set (TXTPTR)=BBUFF - (Y)=line
				length & (TXTPTR) pointing to BBUFF-1 on exit (Vic/64 through (0304) to C57C)
C52C	C613	A613	FNDLIN	Search BASIC text from start for line number in (LINNUM) ... or ...
C530	C617	A617	FNDLNC	Start search at (A/X) - C=1 if found & (LINPTR) points to line header
C55B	C642	A642	SCRATH	*** NEW - check syntax then ...
C55D	C644	A644	SCRTCH	Reset first byte of text to 00 - set (VARTAB)=(TXTTAB)+2 then ...
C572	C659	A659	RUNC	Reset execution to start of program (STXPTR) and then to CLEARC ...
C577	C65E	A65E	CLEAR	*** CLR - check syntax then ...
C579	C660	A660	CLEARC	Set (FRETOP)=(MEMSIZ) - abort I/O - set (ARYTAB) & (STREND)=(VARTAB) then ...
C590	C677	A677	LDCLR	Do RESTOR - reset (TEMPPT) - reset stack

```
B2/3 VIC  64  LABEL   --------------------------------DESCRIPTION------------------------------------

C5A7 C68E A68E STXPT   Set (TXTPTR) to (TXTTAB)-1  - ie reset execution to start of program
C5B5 C69C A69C LIST    *** LIST
C630 C71A A71A QPLOP   Handle LIST character - if ordinary character or token in quotes then print
                       if normal token, then expand and print it (Vic/64 through (0306) to C71A)
C658 C742 A742 FOR     *** FOR -  save (TXTPTR), (CURLIN) and final value on stack then ...
C6C4 C7AE A7AE NEWSTT  Check for STOP key then handle next BASIC statement from text
C6DA C7C4 A7C4 CKEOL   Check end of line is also end of text - if not then get next line parameters
C6F7 C7E1 A7E1 GONE    Execute statement within line (Vic/64 through (0308) to C7E4)
C700 C7ED A7ED GONE3   Interpret BASIC command and execute it
C730 C81D A81D RESTORE *** RESTORE  - reset (DATPTR) to start of BASIC
C73F C82F A82F STOP    *** STOP     - flag "BREAK" message for exit then do END
C741 C831 A831 END     *** END      - flag exit to "READY"  then ...
C744 C834 A834 FINID   If not in direct mode - save (TXTPTR) in (OLDTXT) then ...
C751 C841 A841 STPEND  Save (CURLIN) in (OLDLIN) and exit to READY if END or ERRFIN if STOP
C76B C857 A857 CONT    *** CONT    - if   (OLDTXT+1)=0 then "CAN'T CONTINUE" ...
                                     else (TXTPTR)=(OLDTXT) (CURLIN)=(OLDLIN)
C785 C871 A871 RUN     *** RUN     - if followed by number do CLR and then GOTO
C790 C883 A883 GOSUB   *** GOSUB   - save (TXTPTR), (CURLIN) & GOSUB flag on stack then GOTO
C7AD C8A0 A8A0 GOTO    *** GOTO    - read number from text to (LINNUM) then ...
C7B0 C8A3 A8A3 GOTOC   Scan for end of current line - search for line (LINNUM) & set (TXTPTR)
C7DA C8D2 A8D2 RETURN  *** RETURN - check syntax then ...
C7DC C8D4 A8D4 RTC     Clear stack up to first GOSUB entry - then set (TXTPTR) & (CURLIN) from stack
C800 C8F8 A8F8 DATA    *** DATA - scan text for end of statement & update (TXTPTR) accordingly
C80E C906 A906 DATAN   Set scan for statement delimiter (colon or zero byte) then do search ...
C811 C909 A909 REMN    Set scan for line delimiter (zero byte) then do search ...
C813 C90B A90B SERCHX  Search text for char in X or 00 - exit with (Y)= number of bytes to delimiter
C830 C928 A928 IF      *** IF  - evaluate expression .. if FALSE (ie zero) perform REM
C843 C93B A93B REM     *** REM - scan for end of line delimiter & update (TXTPTR)
C848 C940 A940 DOCOND  Continue IF - if TRUE (non zero) then do command (or GOTO if digit follows)
C853 C94B A94B ONGOTO  *** ON - get number from text & scan for line number .. do GOTO or GOSUB
C873 C96B A96B LINGET  Read FXPT number from text .. result in (LINNUM) .. number in range 0-63999
C8AD C9A5 A9A5 LET     *** LET - find target variable .. set (FORPNT)=pointer .. evaluate expression
C8CC C9C4 A9C4 PUTINT  Put (FAC) as INTEGER variable in memory pointed to by (FORPNT)
C8DE C9D6 A9D6 PTFLPT  Put (FAC) as FLPT    variable in memory pointed to by (FORPNT)
C8EB C9E3 A9E3 PUTTIM  Set TI$ from string - set (INDEX1) to point to string & (A)=length (6 chars)
C937 CA2C AA2C GETSPT  Put string descriptor, (FAC+3) points, in string var pointed to by (FORPNT)
C98B CA80 AA80 PRINTN  *** PRINT# - call CMD then unlisten IEEE bus and restore default I/O
C991 CA86 AA86 CMD     *** CMD - set CMD output device from file table then call PRINT routine
C9A5 CA9A AA9A STRDON  PRINT routine - print string & check for end of PRINT statement
C9AB CAA0 AAA0 PRINT   *** PRINT - identify PRINT parameters (TAB etc) .. evaluate expression
C9C3 CAB8 AAB8 VAROP   Output variable - if number, convert to string .. output string
```

B2/3	VIC	64	LABEL	DESCRIPTION
C9E2	CAD7	AAD7	CRDO	Output CR/LF (Vic/64 doesn't send LF if (CHANNL) > 127)
C9EF	CAE8	AAE8	COMPRT	Handle COMMA TAB and SPC in PRINT statement
CA1C	CB1E	AB1E	STROUT	Output string - (A/Y) must point to start .. end with zero byte or inv commas
CA1F	CB21	AB21	STRPRT	Output string - (FAC+3) must point to string descriptor
CA22	CB24	AB24	OUTSTR	Output string - set (INDEX1) to point to start of string & (A)=string length
CA39	CB3B	AB3B	OUTSPC	Output cursor right (or space if output not directed to screen)
CA3D	CB3F	AB3F	PRTSPC	Output space always
CA40	CB42	AB42	OUTSKP	Output cursor right always
CA43	CB45	AB45	OUTQST	Output question mark
CA45	CB47	AB47	OUTDO	Output char in A
CA4F	CB4D	AB4D	TRMNOK	Handle bad data - output appropriate message for INPUT/READ/GET
CA7D	CB7B	AB7B	GET	*** GET - confirm not direct mode .. identify GET# .. INPUT one char
CAA7	CBA5	ABA5	INPUTN	*** INPUT# - set input device .. do INPUT .. unlisten IEEE bus & restore I/O
CAC1	CBBF	ABBF	INPUT	*** INPUT - output string if present .. receive input
CAED	CBEA	ABEA	BUFFUL	Read input - if (BBUFF)=0 and KB: B2/3 aborts, Vic/64 skips to next statement
CAFA	CBF9	ABF9	QINLIN	Prompt (if I/O is normal) and receive line of input to BASIC input buffer
CB07	CC06	AC06	READ	*** READ - set flag for READ and set (X/Y)=(DATPTR)
CB0E	CC0D	AC0D	INPCON	READ entry for INPUT - flag INPUT and set (X/Y)=BBUFF (ptr to input buffer)
CB10	CC0F	AC0F	INPCO1	READ entry for GET (already flagged and (X/Y)=BBUFF (ptr to input buffer)
CB36	CC35	AC35	RDGET	General purpose read routine - GET routine
CB44	CC43	AC43	RDINP	General purpose read routine - INPUT routine
CBB9	CCB8	ACB8	DATLOP	General purpose read routine - READ routine
CBFC	CCFC	ACFC	EXINT	Message - "?EXTRA IGNORED" CR
CC0D	CD0C	AD0C	TRYAGN	Message - "?REDO FROM START" CR
CC20	CD1E	AD1E	NEXT	*** NEXT - get variable .. confirm FOR entry on stack .. calculate next value
CC62	CD61	AD61	DONEXT	If loop valid set (CURLIN) & (TXTPTR) from stack .. continue execution
CC8B	CD8A	AD8A	FRMNUM	Evaluate expression from text ("TYPE MISMATCH ERROR" if not numeric)
CC8E	CD8D	AD8D	CHKNUM	Confirm numeric result ("TYPE MISMATCH ERROR" if not)
CC90	CD8F	AD8F	CHKSTR	Confirm string result ("TYPE MISMATCH ERROR" if not)
CC9F	CD9E	AD9E	FRMEVL	Evaluate expression in text - (FAC) will hold numeric result & (FAC+3) will point to descriptor of a string result
CD84	CE83	AE83	EVAL	Handle single term in expression .. identify function/number/pi etc
CDA3	CEA8	AEA8	PIVAL	# 3.1415965 in MFLPT - PI
CDA8	CEAD	AEAD	QDOT	Continue handling single term in expression - handle non-variable term
CDEC	CEF1	AEF1	PARCHK	Evaluate bracketted expression in text
CDF2	CEF7	AEF7	CHKCLS	Confirm char pointed to by (TXTPTR) is right bracket - "SYNTAX ERROR" if not
CDF5	CEFA	AEFA	CHKOPN	Confirm char pointed to by (TXTPTR) is left bracket - "SYNTAX ERROR" if not
CDF8	CEFD	AEFD	CHKCOM	Confirm char pointed to by (TXTPTR) is comma - "SYNTAX ERROR" if not
CDFA	CEFF	AEFF	SYNCHR	Confirm char pointed to by (TXTPTR) is same as A - "SYNTAX ERROR" if not
CE03	CF08	AF08	SYNERR	Output "SYNTAX ERROR"

B2/3	VIC	64	LABEL	DESCRIPTION
CE08	CF0D	AF0D	DOMIN	Set up monadic minus (or NOT function) for later evaluation
----	CF14	AF14	RSVVAR	Identify variable pointed to by (FAC+3) as reserved (eg ST or TI) C=0 if not
CE0F	CF28	AF28	ISVAR	Find var named in text - (FAC+3) will point to descriptor if string
				(FAC) will hold value as FLPT if numeric (integer or FLPT)
CE2E	CF48	AF48	TISASC	Convert TI to ASCII string - (FAC+3) will point to descriptor of ASCII string
CE89	CFA7	AFA7	ISFUN	Identify function type & evaluate it
CE93	CFB1	AFB1	STRFUN	If string function then save string descriptor on stack and evaluate
CEB3	CFD1	AFD1	NUMFUN	If numeric function then evaluate expression in brackets and perform function
CEC8	CFE6	AFE6	OROP	*** OR - set OR flag then do AND
CECB	CFE9	AFE9	ANDOP	*** AND - set AND flag - convert FLPT to FXPT - do AND then FXPT to FLPT
CEF8	D016	B016	DOREL	*** < = > - check variable type then ...
CEFD	D01B	B01B	NUMREL	Perform numeric comparison or ...
CF10	D02E	B02E	STRREL	Perform string comparison
CF63	D081	B081	DIM	*** DIM
CF6D	D08B	B08B	PTRGET	Identify variable named in text and put name in (VARNAM) then ...
CFC9	D0E7	B0E7	ORDVAR	If ordinary var use name in (VARNAM) to find it - if not found, do NOTFNS
CFF7	D113	B113	ISLETC	Is character in A a letter? Set C=1 if it is
D001	D11D	B11D	NOTFNS	Create new variable if not found - if called during expression evaluation
				(checks PTRGET called from ISVAR) then will not create and returns zero
D00C	D128	B128	NOTEVL	Create variable with name (VARNAM) - (VARPNT) points to variable on exit
D078	D194	B194	FMAPTR	Set (ARYPNT) to start of array .. enter with (COUNT)=no. of array dimensions
D089	D1A5	B1A5	N32768	# 32768 in FLPT
----	D1AA	B1AA	FACINX	Convert (FAC) to integer in (A/Y) range -32767 to +32767
D08D	D1B2	B1B2	INTIDX	Evaluate text expression for integer range 0-32767 - result in (FAC+3)
D09A	D1BF	B1BF	AYINT	Convert (FAC) to integer in range 0-32767 - result in (FAC+3)
D0AC	D1D1	B1D1	ISARY	Get ARRAY parameters from text (number of subscripts etc)
D0F3	D218	B218	FNDARY	Find ARRAY whose name is in (VARNAM) and whose parameters were read from text
D13C	D261	B261	NOTFDD	If array not found, create it from parameters on stack
D1EA	D30E	B30E	INLPN2	Locate element within array and set (VARPNT) to point to it
D228	D34C	B34C	UMULT	Compute number of bytes in subscript (Y) of array starting at (VARPNT)
D259	D37D	B37D	FRE	*** FRE - do garbage collect .. FRE= (FRETOP)-(STREND)
D26D	D391	B391	GIVAYF	Convert integer in (A/Y) to FLPT in (FAC) - range 0-32767
D27A	D39E	B39E	POS	*** POS - return value of (CPOS)in (FAC)
D27C	D3A2	B3A2	SNGFT	Convert (Y) to FLPT in (FAC) in range 0-255
D280	D3A6	B3A6	ERRDIP	Check for direct mode "ILLEGAL DIRECT" if it is - ie if (CURLIN+1)=FF
D28D	D3B3		DEF	*** DEF - create FN variable
D2BB	D3E1		GETFNM	Check syntax of FN and locate FN descriptor - set (DEFPNT) to point to it
D2CE	D3F4		FNDOER	*** FN - handle FN variable .. get FN descriptor then ....
D2FD	D423		SETFNV	Set (TXTPTR) to start of FN in text .. evaluate expression .. reset (TXTPTR)

B2/3	VIC	64	LABEL	DESCRIPTION
D33F	D465	B465	STRD	*** STR$ - evaluate expression .. convert result to ASCII string
D34F	D475	B475	STRINI	Make room in string space for string descriptor in (FAC+3) & (A)=length - exit with new descriptor in (DSCTMP) and (DSCPNT)=pointer to old descriptor
D361	D487	B487	STRLIT	Scan string starting at (A/Y) and create descriptor - exit with (FAC+3) pointing to descriptor - string must end with zero byte or inv commas
D3AF	D4D5	B4D5	PUTNW1	Save descriptor on descriptor stack & update pointer
D3CE	D4F4	B4F4	GETSPA	Set (FRETOP) & (FRESPC) for new string of length in (A)
D400	D526	B526	GARBA2	Garbage collect - eliminate all unwanted strings in memory
D497	D5BD	B5BD	DVARS	Search var/arrays for next string descriptor to be saved by garbage collect
D4E0	D606	B606	GRBPAS	Move string up to overwrite unwanted strings (used by garbage collect)
D517	D63D	B63D	CAT	Concatenate two strings in expression then continue to evaluate expression
D554	D67A	B67A	MOVINS	Transfer string whose descriptor is pointed to by (STRNG1)
D57D	D6A3	B6A3	FRESTR	Confirm string mode then ...
D580	D606	B606	FREFAC	Perform string housekeeping - enter with ptr to string descriptor in (FAC+3) and exit with (A)=length and (INDEX1) pointing to start of string
D5B5	D6DB	B6DB	FRETMS	Update string descriptor stack pointer
D5C6	D6EC	B6EC	CHRD	*** CHR$
D5DA	D700	B700	LEFTD	*** LEFT$
D600	D72C	B72C	RIGHTD	*** RIGHT$
D611	D737	B737	MIDD	*** MID$ - if only one numeric parameter set right limit to FF
D63B	D761	B761	PREAM	Pull string descriptor ptr to (DSCPNT) and string parameter to (A) from stack
D656	D77C	B77C	LEN	*** LEN
D65C	D782	B782	LEN1	Do string housekeeping then force numeric mode - exit with (Y)=string length
D665	D78B	B78B	ASC	*** ASC - get first character in string and convert to FLPT
D675	D79B	B79B	GTBYTC	Evaluate expression in text .. convert to byte parameter (0-255) in (X)
D687	D7AD	B7AD	VAL	*** VAL - confirm string expression then ...
D68F	D7B5	B7B5	STRVAL	Convert string starting at (INDEX1) & (A)=length to FLPT number in (FAC)
D6C6	D7EB	B7EB	GETNUM	Read parameters for POKE/WAIT from text - (INDEX1)=1st integer (X)=2nd value
D6D2	D7F7	B7F7	GETADR	Convert FAC to integer in (INDEX1) in range 0-65535
D6E8	D80D	B80D	PEEK	*** PEEK
D707	D824	B824	POKE	*** POKE
D710	D82D	B82D	WAIT	*** WAIT - (BASIC 2/3 - if WAIT 6502,X then output "MICROSOFT!" X times)
D72C	D849	B849	FADDH	Add 0.5 to (FAC)
D733	D850	B850	FSUB	Subtract (FAC) = MFLPT at (A/Y) - (FAC)
D736	D853	B853	FSUBT	*** SUBTRACTION routine - (FAC) = (ARG) - (FAC)
D76E	D862	B862	FADD5	Part of addition normalisation routine
D773	D867	B867	FADD	Add (FAC) = MFLPT at (A/Y) + (FAC)
D776	D86A	B86A	FADDT	*** ADDITION routine (FAC) = (ARG) + (FAC) - enter with (A)=FAC exponent
D88A	D97E	B97E	OVERR	Output "OVERFLOW ERROR"

B2/3	VIC	64	LABEL	DESCRIPTION
D88F	D983	B983	MULSHF	Multiply by a byte - used by multiplication routine whenever byte =00
D8C8	D9BC	B9DC	FONE	# 1 in MFLPT
D8CD	D9C1	B9C1	LOGCN2	Count for 4 constants in LOG series
D8CE	D9C2	B9C2		# 0.434255942 in MFLPT constant #1 for LOG series
D8D3	D9C7	B9C7		# 0.57658454 in MFLPT constant #2 for LOG series
D8D8	D9CC	B9CC		# 0.961800759 in MFLPT constant #3 for LOG series
D8DD	D9D1	B9D1		# 2.885390073 in MFLPT constant #4 for LOG series
D8E2	D9D6	B9D6	SQR05	# 0.707106781 in MFLPT SQR(0.5)
D8E7	D9DB	B9DB	SQR20	# 1.41421356 in MFLPT SQR(2)
D8EC	D9E0	B9E0	NEGHLF	# -0.5 in MFLPT
D8F1	D9E5	B9E5	LOG2	# 0.693147181 in MFLPT LOG(2)
D8F6	D9EA	B9EA	LOG	*** LOG FAC = LOG(FAC)
D934	DA28	BA28	FMULT	Multiply FAC = MFLPT at (A/Y) * (FAC)
D937	DA2B	BA2B	FMULTT	*** MULT routine - FAC = (ARG) * (FAC) enter with (A)=FAC exponent
D965	DA59	BA59	MLTPLY	Multiply by a BIT - used by multiplication routine
D998	DA8C	BA8C	CONUPK	Load (ARG) from MFLPT at (A/Y) .. convert MFLPT to FLPT in (ARG)
D9C3	DAB7	BAB7	MULDIV	Test (FAC) & (ARG) for under/overflow .. set sign for MULT and DIVISION
D9E0	DAD4	BAD4	MLDVEX	If overflow then output OVERFLOW ERROR .. if UNDERFLOW force zero
D9EE	DAE2	BAE2	MUL10	Multiply (FAC) by 10 - (FAC) = (FAC) * 10
DA05	DAF9	BAF9	TENC	# 10 in MFLPT
DA0A	DAFE	BAFE	DIV10	Divide (FAC) by 10 (FAC) = (FAC) / 10 - NB result always positive
DA13	DB07	BB07	FDIVF	(FAC) = (ARG) / MFLPT at (A/Y) - enter with (X)=sign of result
DA1B	DB0F	BB0F	FDIV	(FAC) = MFLPT at (A/Y) / (FAC)
DA1E	DB12	BB12	FDIVT	*** DIVISION routine - (FAC) = (ARG) / (FAC) enter with (A)=FAC exponent
DAAE	DBA2	BBA2	MOVFM	Load (FAC) with MFPLT at (A/Y) .. convert MFLPT to FLPT in (FAC)
DAD3	DBC7	BBC7	MOV2F	Load temp store at (TEMPF2) from (FAC) .. convert FLPT to MFLPT
DAD6	DBCA	BBCA	MOV1F	Load temp store at (TEMPF1) from (FAC) .. convert FLPT to MFLPT
DAE0	DBD4	BBD4	MOVMF	Load memory at (X/Y) from (FAC) .. convert FLPT to MFLPT
DB08	DBFC	BBFC	MOVFA	Load (FAC) from (ARG)
DB18	DC0C	BC0C	MOVAF	Load (ARG) from (FAC)
DB27	DC1B	BC1B	ROUND	Round off (FAC) - if overflow byte (FACOV) has B7=1
DB37	DC2B	BC2B	SIGN	Check (FAC) - (A)=1 if +ve .. (A)=0 if zero .. (A)=FF if -ve
DB45	DC39	BC39	SGN	*** SGN
DB64	DC58	BC58	ABS	*** ABS
DB67	DC5B	BC5B	FCOMP	Compare (FAC) with FLPT at (A/Y) - on exit sets (A) according to -
				(A)=0 if memory=(FAC) (A)=01 if memory<(FAC) (A)=FF if memory>(FAC)
DBA7	DC9B	BC9B	QINT	Convert (FAC) to integer (4 bytes) in FAC+1 to FAC+4
DBD8	DCCC	BCCC	INT	*** INT - convert (FAC) to integer then back to FLPT

B2/3	VIC	64	LABEL	DESCRIPTION
DBFF	DCF3	BCF3	FIN	Read number from text and put in (FAC)
DCBF	DDB3	BDB3	N0999	# 99999999.9 in MFLPT constant for string conversion
DCC4	DDB8	BDB8	N9999	# 999999999 in MFLPT constant for string conversion
DCC9	DDBD	BDBD	NMIL	# 1000000000 in MFLPT constant for string conversion
DCCE	DDC2	BDC2	INPRT	Output "IN" and line number from CURLIN
DCD9	DDCD	BDCD	LINPRT	Output integer in (X/A) (0-65535) - convert to FLPT in (FAC) then --
DCE3	DDD7	BDD7	FACOUT	Output (FAC) as ASCII string
DCE9	DDDD	BDDD	FOUT	Convert (FAC) to string - (A/Y)=ptr to start of string - end with zero byte
DD74	DE68	BE68	FOUTIM	Convert TI to string - (A/Y)=ptr to start of string - end with zero byte
DE1D	DF11	BF11	FHALF	# 0.5 in MFLPT
DE1E	DF12	BF12	ZERO	# 0 (zero) in MFLPT
DE22	DF16	BF16	FOUTBL	Table of constants (powers of 10) used to convert (FAC) to string
DE5E	DF71	BF71	SQR	*** SQR (FAC) = SQR(FAC)
DE68	DF7B	BF7B	FPWRT	*** POWER routine (FAC) = (ARG) ^ (FAC) - set (A)=FAC exponent on entry
DEA1	DFB4	BFB4	NEGOP	Negate (FAC)
DEAC	DFBF	BFBF	LOGEB2	# 1.44269504 in MFLPT constant for EXP routine
DEB1	DFC4	BFC4	EXPCON	Counter for 8 constants in EXP series (set to 07)
DEB2	DFC5	BFC5		# 0.0000214987637 in MFLPT constant #1 for EXP series
DEB7	DFCA	BFCA		# 0.00014352314 in MFLPT constant #2 for EXP series
DEBC	DFCF	BFCF		# 0.00134226348 in MFLPT constant #3 for EXP series
DEC1	DFD4	BFD4		# 0.00961401701 in MFLPT constant #4 for EXP series
DEC6	DFD9	BFD9		# 0.0555051269 in MFLPT constant #5 for EXP series
DECB	DFDE	BFDE		# 0.240226385 in MFLPT constant #6 for EXP series
DED0	DFE3	BFE3		# 0.693147186 in MFLPT constant #7 for EXP series
DED5	DFE8	BFE8		# 1 in MFLPT constant #8 for EXP series
DEDA	DFED	BFED	EXP	**** EXP - (FAC) = EXP(FAC)
DF2D	E040	E043	POLYX	Evaluate series for function - (A/Y) points to counter at start of constants
DF77	E08A	E08D	RMULC	# 11879546.4 in MFLPT constant for RND evaluation
DF7B	E08F	E092	RADDC	# 0.000000039276778 in MFLPT constant for RND evaluation
DF7F	E094	E097	RND	*** RND (FAC) = random number determined according to (FAC) as follows --
DF86	E09B	E09E	RND0	Seed is "random" registers in I/O chips if (FAC) = 0
DF9D	E0BB	E0BE	QSETNR	Seed is last random number in (RNDX) if (FAC) > 0
DFB2	E0D0	E0D3	RND1	Seed is ABS value of (FAC) itself if (FAC) < 0
----	E0F6	E0F9	BIOERR	Handle input/output error within BASIC
----	E109	E10C	BCHOUT	Output character (uses CHROUT - FFD2) - exit via BIOERR if there's a problem
----	E10F	E112	BCHIN	Input character (uses CHRIN - FFCF) - exit via BIOERR if there's a problem
----	E115	E118	BCKOUT	Set up for output (uses CHKOUT - FFC9) - exit via BIOERR if there's a problem
----	E11B	E11E	BCKIN	Set up for input (uses CHKIN - FFC6) - exit via BIOERR if there's a problem
----	E121	E124	BGETIN	Get one character (uses GETIN - FFE4) - exit via BIOERR if there's a problem

NOTE: BASIC 2/3 handles SAVE, LOAD, OPEN, CLOSE and SYS as part of the operating system and goes straight to these routines via the kernal jump table at FFC0 onwards. These BASIC 2/3 kernal routines read the parameters from text and call the appropriate routines within the operating system

On the other hand, Vic and 64 BASICs treat the kernal routines purely as operating system routines and their SAVE, LOAD, OPEN, CLOSE and SYS routines are actually part of the BASIC interpreter. In these cases, BASIC reads and sets up the parameters from text and then calls the kernal routines as appropriate. The kernal routines assume that these parameters are set and do not "consult" the text.

B2/3	VIC	64	LABEL	DESCRIPTION
FFDE	E127	E12A	SYS	*** SYS Vic/64 sets (A), (X), (Y) and status registers from (SYSA), (SYSX), (SYSY), & (SYSS) before entering the machine code routine.
----	E144	E148	SYSRTS	Called automatically by the RTS at the end of a SYS routine. The resulting register contents are put back into (SYSA), (SYSX), (SYSY) and (SYSS).
FFD8	E153	E156	SAVET	*** SAVE read parameters & do SAVER (Vic/64 sets (A) to pint to the zero page address holding the start address for SAVER - normally (TXTTAB) and (X/Y)is set to (VARTAB) as the end address.)
F6A1	E15C	E15F	SAVER	SAVE to specified device (Vic/64 goes via kernal SAVE routine at FFD8)
FFDB	E162	E165	VERFYT	*** VERIFY flag VERIFY .. do LOAD routine .. check for no errors
FFD5	E165	E168	LOADT	*** LOAD read parameters & do LOADR (Vic/64 (X/Y)=(TXTTAB) as start address)
F322	E172	E175	LOADR	LOAD from specified device (Vic/64 goes via kernal LOAD at FFD5)
F2F9	E1A7	E1A7	LDDIRC	If direct LOAD, set (VARTAB) from address of last byte LOADed and go to FINI
F404	E1B5	E1B5	LDIND	If LOAD from within program, set (TXTPTR) to start and goto LDCLR
FFC0	E1BB	E1BE	OPENT	*** OPEN set file parameters from text and then ...
F524	E1BE	E1C1	OPENR	OPEN file (Vic/64 goes via kernal OPEN routine at FFC0)
FFC3	E1C4	E1C7	CLOSET	*** CLOSE set file parameters from text and then ...
F2AC	E1C7	E1CA	CLOSER	CLOSE file (Vic/64 goes via kernal CLOSE routine at FFC3)
F43E	E1D1	E1D4	SLPARA	Get parameters from text for LOAD/SAVE - set default values if required
F460	E1FD	E200	COMBYT	Confirm comma and then read integer (0-255) from text
F50E	E203	E206	DEFLT	Abort calling subroutine if end of statement (ie leave default values)
F516	E20B	E20E	CMMERR	Confirm character is comma and not end of line - "SYNTAX ERROR" if not
F4CE	E216	E219	OCPARA	Get parameters from text for OPEN/CLOSE - set default values if required
DFD8	E261	E264	COS	*** COS - (FAC) = COS(FAC)
DFDF	E268	E26B	SIN	*** SIN - (FAC) = SIN(FAC)
E02B	E2B1	E2B4	TAN	*** TAN - (FAC) = TAN(FAC)
E054	E2DD	E2E0	PI2	# 1.570796327 in MFLPT PI/2 constant for TRIG evaluation
E059	E2E2	E2E5	TWOPI	# 6.28318531 in MFLPT PI*2 constant for TRIG evaluation
E05E	E2E7	E2EA	FR4	# 0.25 in MFLPT constant for TRIG evaluation

```
B2/3 VIC  64  LABEL    ______________________________DESCRIPTION_____________________________________

E063 E2EC E2EF SINCON  Counter for 6 constants in SIN series (set to 05)
E064 E2ED E2F0         # -14.3813907       in MFLPT         constant #1  for SIN series
E069 E2F2 E2F5         #  42.0077971       in MFLPT         constant #2  for SIN series
E06E E2F7 E2FA         # -76.7041703       in MFLPT         constant #3  for SIN series
E073 E2FC E2FF         #  81.6052237       in MFLPT         constant #4  for SIN series
E078 E301 E304         # -41.3417021       in MFLPT         constant #5  for SIN series
E07D E306 E309         #   6.28318531      in MFLPT   PI*2 constant #6  for SIN series
E08C E30B E30E ATN     *** ATN  -  (FAC) = ATN(FAC)
E0BC E33B E33E ATNCON  Counter for 13 constants in ATN series (set to 0B)
E0BD E33C E33F         # -0.000684793912   in MFLPT         constant #1  for ATN series
E0C2 E341 E344         #  0.00485094216    in MFLPT         constant #2  for ATN series
E0C7 E346 E349         # -0.161117018      in MFLPT         constant #3  for ATN series
E0CC E34B E34E         #  0.034209638      in MFLPT         constant #4  for ATN series
E0D1 E350 E353         # -0.0542791328     in MFLPT         constant #5  for ATN series
E0D6 E355 E358         #  0.0724571965     in MFLPT         constant #6  for ATN series
E0DB E35A E35D         # -0.0898023954     in MFLPT         constant #7  for ATN series
E0E0 E35F E362         #  0.110932413      in MFLPT         constant #8  for ATN series
E0E5 E364 E367         # -0.142839808      in MFLPT         constant #9  for ATN series
E0EA E369 E36C         #  0.19999912       in MFLPT         constant #10 for ATN series
E0EF E36E E371         # -0.333333316      in MFLPT         constant #11 for ATN series
E0F4 E373 E374         #  1                in MFLPT         constant #12 for ATN series

---- E378 E394 INIT    Initialise BASIC on reset - set BASIC vectors with INITV then ...
E116 E37B E397 INITNV  Set up BASIC variables with INITCZ,  do "BYTES FREE" initial message (INITMS)
                       and  enter BASIC  at  READY     NOTE:  Vic/64  does  this  through series  of
                       subroutines, BASIC2/3 is one routine which also contains the RAM check.
E0F9 E387 E3A2 INITAT  CHRGET routine - copied into RAM on reset
E111 E39F E3BA RNDSED  #   0.811635157  in MFLPT - initial RND seed (transferred to RAM with INITAT)
---- E3A4 E3BF INITCZ  Initialise BASIC RAM - set USRPOK,  FACINT & INTFAC - xfer INITAT & RNDSED to
                       CHRGET & RNDX - set (TXTTAB)=(LORAM) & (FRETOP)=(HIRAM) - first text byte = 0
---- E404 E422 INITMS  Output opening message .. calculate & output bytes free (MEMSIZ)-(TXTTAB)
E1B7 E429 E460 WORDS   "BYTES FREE"
E1C4 E436 E473 FREMES  BASIC2/3 has "### COMMODORE BASIC ###"  - VIC has "**** CBM BASIC V2 ****"
---- E44F E447 BVTRS   Table of BASIC vectors for transfer into RAM at WNDERR
---- E45B E453 INITV   Routine to transfer BVTRS into RAM
---- E467 E37B BASSFT  BASIC soft  reset routine  -  called  by  BREAK routine  if  BRK  instruction
                       encountered or RESTORE key pressed.    NB on the 64,  both BASSFT and INIT go
                       back to READY through vector at $0300 if B7 of X=1. The vector at $0300 would
                       otherwise produce an ERROR message.
---- E476 ---- LDRST   LOAD patch (for indirect LOAD) - Rebuild chaining & do partial reset (LDCLR)
```

SHOP WINDOW

A program generator for the 8000-series machines is available called Codewriter. Program generators enable non-programmers to create programs in response to a series of prompts in plain english. The productivity so obtained can be quite startling. (The brochure claims that you can write your first program in minutes). Codewriter Disk#1 covers screen layout. data entry validation, display of user-defined error messages, screen calculations, searching by any field, whereas Codewriter Disk#2 provides printed reports and menu generators. Disk#1 costs £ 125, disk#2 costs £ 65 from Dynatech Microsoftware Ltd., Summerfield House. Summerfield Road, Vale, Guernsey, Channel Islands. Tel: 0481 47377. Note: Do not add VAT. it cannot be collected in the Channel Islands, send VAT No. instead. Disk#1 price includes the manual.

Codewriter has already sold over 1,000 copies of the 8032 version in its first five months of availability.

The PET makes a useful 6502 machine-code development system and several manufacturers have exploited this potential. Microtrol Engineering Design Ltd. produce the MEDDEV-P which is a file-based assembler and offset loader package for 16K or 32K CBM 'new-ROM' machines. An EPROM programmer is also available and is reviewed elsewhere in this issue. M.E.D. Ltd. are at 640, Melton Road, Thurmaston. Leicester, LE4 8BB. Tel: (0533) 704492.

The second development system features true real-time emulation, modules for 6502, 6802, 6803/6303 and 6809 processors and is self-contained in a 3Ux42E Eurocard case. The interface is IEEE-488. Supplied on floppy disk are editor files. assembler BASIC file, assembler machine-code file for each processor specified, offset loader and handbook as an ASCII file. Further details from Tylek Ltd., 2, Parkview Drive. Cashes Green, Stroud. Glos. GL5 4NQ. Tel: 04536-77257.

--oOo--

REVIEW

Starting Forth Paperback £15.60 incl.
By Leo Brodie. Forth Inc. Hardback £19.10 incl.

This book has become the classic work on Forth. It is written in an easy going style and is illustrated in the american manner which assumes that anyone learning something must be a child and treated as such. and so little amusing sketches abound (somewhat like in the Newsletter !). However. a side-effect of this is that no matter how difficult the concept. it is explained in an intelligible manner. Bear in mind that it is probably easier for a new-comer to learn to program in Forth than it is for someone weaned on a more 'conventional' language.

The foreword is written by the only person who didn't have to learn to program in Forth. its inventor. Charles H. Moore. The text is arranged so that you only have to read what you need to know. Footnotes are used for additional information. sometimes to explain a computing term to a beginner. or perhaps to give further explanation to those seeking more knowledge of the inner workings.

All commands appear summarised twice; first at their introduction and secondly in summary at the end of the chapter. There is no index, but Appendix 4 is a list of Forth words and the page of the Chapter where they are summarised.

The fundamentals of the Forth model are explained in the first chapter so that the reader can follow the use of the dictionary and stack. The arithmetic manipulations follow where postfix notation (otherwise called Reverse Polish Notation) and the stack operators are introduced.

Editing commands are covered in chapter 3. but since these are not part of the Forth required word set. there may well be differences between the book and the version on one's system.

A whole chapter is devoted to the IF...THEN construct. and a further chapter to loops. Three more chapters are devoted to numbers because Forth internally uses integer arithmetic for speed, but if 16-bit is unsufficient, 32-bit integers are readily accommodated and furthermore Forth is not tied to decimal arithmetic. Switching from binary to octal, or base 12 (for feet & inches) is simple.

Variables, constants and arrays are lumped togther in chapter 8. These included double-length variables and constants.

The advanced stuff is covered in chapter 9 'Under the Bonnet' and this gives the newcomer and insight into the organisation of the memory when Forth is in action. Chapter 10 covers the processes of input and output and in addition covers input from disk of text.

An unusual feature of Forth is that you can define not just your own words, but your own defining words (I won't explain that here. Brodie's book uses a whole chapter for it).

The final chapter covers three application programming examples. Each chapter ends with a review of terms and some problems (answers in appendix 1).

Handy hints appear at relevant places in the text and are summarised in the contents. Superficially, the book is full of plus points. but to be critical, I would have preferred to have seen an index. There are occasional minor typographical errors and there is a discrepancy between the question 4 and its answer of chapter 11.

The '79-Standard word XOR (exclusive OR) is not covered, neither does it explain how to delete a partially compiled word which could arise from an error. The non-destructive stack print definition supplied uses some words that are not in the minimum required word set and no assistance is given as to how they may be obtained (ICPUG to the rescue - see Forth Column).

Despite these criticisms I have no hesitation in recommending this book to those inexperienced in the Forth language. Available from Computer Solutions Ltd., Treway House. Hanworth Lane. Chertsey, Surrey. KT16 9LA.

R.D.G.

--oOo--

REDEFINING CHARACTER SETS ON THE COMMODORE 64

By Jim Butterfield

QUESTION: How do I redefine the characters on the 64?

ANSWER: There are so many ways that it's impossible to give a general answer. The following outlines the steps needed.

WHERE DO WE PUT EVERYTHING?

Screen memory, characters and sprites (if you use them), must all fit into the same 16K block of memory. To use your own character set you'll probably want to use the default block from 0-16383. A good block to pick would be the top one, which extends from 49152-65535. We'll pick the detailed locations for characters and screen a little later.

Don't do this yet, but we select the top block by: POKE 56576,4.

Now, within that block we must pick screen and character set locations. Looking at the memory maps we find that the top half (57344-65535) is used for the Kernal ROM. We could still use this area; POKEing to the screen and character set would still make the changes in RAM to cause the screen to change. The only problem might be that we couldn't PEEK these locations - we'd get the ROM instead of the RAM. Some programs like to PEEK the screen to see what's there.

Locations 53248-57343 are not too suitable - the input/output chips are located there, and we'd have to do extra work to PEEK and POKE. That leaves 49152-53247 for the screen and character set.

A character set (the "character base") must start at an even multiple of 1024, and of course must be in the selected memory block. This leaves us with 49152 (start-of-block plus 1024 times multiple 0) and 51200 (offset multiple 2). It doesn't matter which we pick - let's use the first one, which means that our character set will go from 49152 to 51199.

Now for the screen. We must pick a multiple of 1024 that is in range. Skipping over the character set area, we have a choice of 51200 (multiple 2) and 52224 (multiple 3). We'll pick the first, so that screen memory occupies 51200-52223.

Let's look at those multiple numbers again. A character set starting at 49152 is an offset of zero within the 16k block, which is 0*1024. A screen memory block starting at 51200 is at an offset of 2048 within the block (2*1024). To set the video chip to use these actual screen addresses we'll do some arithmetic on the multiples: 2 (the screen) times 16, plus 0 (the character set) gives 32. Thus we POKE 53272,32.

TELLING BASIC WHAT WE'VE DONE

The two POKEs have told the video chip where the screen memory and the character set is located, but we must also tell BASIC about it or it will insist on putting everything in the old addresses from 1024-2032. We tell BASIC about the screen with POKE 648,200.

Why 200? That's the screen memory address (51200) divided by 256.

It wouldn't hurt to clear the screen at this point since we've moved to a new area of memory that's cluttered with unused values.

SUPPLYING A CHARACTER SET

We have told the video chip that there's a character set at 49152, but we haven't put any characters there. You could define your own characters of course, but for the moment, let's copy the characters out of ROM.

The processor will find the character ROM at address 53248-55295. That's odd for two reasons. First, the video chip found it at an entirely different location. We can dismiss this as one of the magics of electronics. Secondly, we normally see the input/output chips at these locations. The character set is hidden behind these chips and we'll need to expose it by POKE 1,51 - but not yet!!

Before we make the I/O cnips vanish, we'll need to lock out the interrupt. The interrupt sequences, which check the keyboard and do other jobs, will get hopelessly confused if the I/O chips are not there. Take my advice: never get the interrupt sequences hopelessly confused - it's not good for your program's health. We can lock out the interrupt with POKE 56333,127 - but we must remember to put it back again.

A DEMONSTRATION PROGRAM

We can do all this in a small BASIC program to show that it all works. Let's start by following the suggested sequence of POKEs:

```
100 REM    CHARACTER SET DEMO

110 POKE 56576,4      : REM set 16k bank
120 POKE 53272,32     : REM set screen and char set
130 POKE 648,200      : REM tell BASIC
```

Next, we wish to move the character set - but let's make this a visual thing and put something on the screen first. This will initially appear as scrambled information since the characters haven't yet been defined. As the characters are copied over from ROM, we'll see them take shape.

```
140 PRINT CHR$(147)   : REM clear sreen

150 FOR J = 1 TO 2
160 PRINT "THE QUICK BROWN FOX JUMPED
               OVER A LAZY DOG'S BACK 1234567890"
170 PRINT CHR$(18);   : REM reverse

180 NEXT
```

Now let's copy the character set:

```
    200 FOR J = 0 TO 2047
    210 POKE 56333,127     : REM lockout interrupt
    220 POKE 1,51          : REM reveal character set
    230 X=PEEK(53248+J)    : REM get eighth of character
    240 POKE 1,55          : REM restore I/O
    250 POKE 56333,129     : REM free interrupt
    260 POKE 49152+J,X     : REM copy character
    270 NEXT J
```

One final touch. We have copied the existing character set, but we haven't defined anything new. Let's prove we can do it now by adding a slash to the letter "O" - this will make it look like the Greek letter THETA, or a sloppy number eight. "O" is the fifteenth letter of the alphabet and each letter takes up eight bytes. So we can calculate that "O" starts at 49152 + (8*15) or 49272. Since we want to change the middle of the letter, not the top, we move down three more bytes and type:

```
    280 POKE 49275,126
```

Check your program (mistakes can be fatal) and then RUN it. It's sufficiently slow that you can see the various characters being formed. The letter come first, followed by the numerics. Some time later, the reverse characters are defined.

CONCLUSION

There are lots of ways of setting up your own character set - this is just one of them. The first time is the hardest but once you've done it successfully, you'll be able to cut your own character without problems.

--oOo--

REVIEW

S3P1 EPROM Burner — Microtrol

I sent off my cheque and order and after a rather long wait and two telephone calls, my S3P1 EPROM burner was delivered in a well protected package from Microtrol Engineering Design, 640, Melton Road, Thurmaston, Leicester, LE4 8BB. 2K EPROMs are now in the £ 3 region and so EPROM burning is now within the range of amateurs like myself. I was attracted to this device by skilful publicity and because it uses the USER port and would enable me to use my disks and printer from the IEEE port. I was a little surprised when it was difficult to fit the connector into the port and required the end pieces of the port connector to be pushed well in on frequent occasions. The box is a very neat, slightly sloping, 6" by 4" by 2" plastic RS 508-481 case, and the controls are well laid out. The device is self-powered and has a very long mains flex but the ribbon connector to the PET could benefit from being at least another 6" longer. The ZIF socket is not of the highest quality and I have found it a little difficult to get EPROMs into it on occasions. It has LEDs to show a.c. power-on, 5v EPROM power-on and 25v blowing power-on. It has a toggle switch to program 2716, 2532, 2516 and 2732 EPROMS. although I have only used mine to program 2716 and 2532 chips.

I was very busy when the machine finally arrived and I tried to use it in short bursts. I put my lack of success down to my intellectual inadequacy, of which I have been made aware on numerous occasions by my two sons. However, an exhaustive and exhausting test session revealed that it was faulty. I returned it to MED, who repaired it promptly and sent it back with a covering letter hinting that I had been guilty of poking around inside the USER port connector but they did not charge. I had not been so guilty in any case. but nevertheless this is a point in their favour!

The returned machine is very easy to use. The instructions come on two well laid-out (if feintly printed) sheets of A4 paper showing that a great deal of thought had been given to the instructions. Once the device is connected to the port and the machine code program has been loaded from tape it is easy to first check if the EPROM is clean, see the contents of an area of memory as a pseudo ASCII dump see the contents of an EPROM you wish to copy and to move data from one area of memory to another or from ROM to memory. It is thus possible to copy EPROMs meant for machines other than the PET, but I have not yet tried this out.

There are some checks built into the system and any attempt to write to a non-existent area of EPROM or an area that already has been programmed will generate an error message. Unfortunately it is possible to carry out a check routine on an EPROM in the ZIF and be told it is clean if the device is not properly attached to the USER port and my connector is just a bit oversize for the hole in the casing. I am loathe to shave it down until it is out of guarantee! It is also possible to be told that an EPROM is clean if you have forgotten to turn on the power to the chip. I have some early EPROMs which had their first few bytes blown into Ø when the machine was not working properly! It is always difficult for designers to know what level of IQ they think is the minimum required to operate their devices and perhaps MED are correct and I am wrong in my assessment of the level of protection that can be reasonably provided.

Since learning to use the programmer, I have blown many EPROMs successfully and find it takes about 4 minutes to blow a 4K EPROM. Having written to the chip, it then waits for the chip to cool and carries out a VER(i)FY check. If all is not well, it will tell you so.

My device cost £ 50 and at that price it is well worth having. I think that the price has now increased and anyone wishing to buy one should first check with MED and perhaps also get a delivery date before placing an order.

W.G.C.Austin

--oOo--

COMMODORE 64 - THE VIC ADDRESS MAP

By Brian Grainger

On the Commodore 64 the chip that does all the work in defining what is displayed on the screen is the so-called Video Interface Chip (VIC). Unlike the microprocessor which can address 64K of memory, the VIC can only address 16K, 0000-3FFF. To get this range of values 14 address lines are required VA0-VA13 (VA = VIC address ?).

To provide the VIC with access to all the 64K of memory requires two further address lines, VA14-VA15. There are 4 possible combinations of values which can be set by these two address lines. Each combination sets the VIC to look at a particular bank of 16K. As 4*16K is 64K the VIC thus has access to all the memory, although only 1 bank at a time.

The address lines VA14. VA15 are located as bit 0 and bit 1, respectively. of the Peripheral Data Register of the Complex Interface Adapter chip (6526) No.2 i.e. CIA2- PRA. CIA2- PRA is located at $DD00 (#56576) in the microprocessor memory map. The lines use inverted logic so we get the following possibilites:

DD00 Bit1	DD00 Bit0	VA15	VA14	Memory Accessible to VIC	Bank
1	1	0	0	0000-3FFF	0
1	0	0	1	4000-7FFF	1
0	1	1	0	8000-BFFF	2
0	0	1	1	C000-FFFF	3

Remember that whatever bank is addressed the VIC memory map is always 0000-3FFF so we can regard the true microprocessor address being the VIC address plus the offset set in DD00.

When the machine is switched on, Bank 0 is selected so the VIC addresses microprocessor locations 0000-3FFF.

IMPORTANT NOTE:

The VIC CANNOT address the RAM in the regions 1000 - 1FFF or 9000-9FFF of the processor map. In these areas the VIC sees the character generator ROM although the processor sees RAM. This means that the character generator is only available in Banks 0 or 2 and CANNOT be removed from these banks.

Because of the VIC's ability to address 4 independent banks it ought to be clear that it is possible to set a character screen in one bank and another character screen in a different bank. We could alternate the displays by changing the settings in DD00. We could display one screen while modifying another. By changing other pointers as well we could alternate a Hi-res display with a character display. This is done in my example Hi-res graphic program detailed elsewhere in this Newsletter. When I come to talk of how character displays, or Hi-res displays are stored you will see that one could have more than one display in a single bank.

As you can see, the VIC is a very versatile chip: one of the superb features of the 64.

--oOo--

BLANK LINES IN LISTINGS from Mike Gross-Nicklaus
Courtesy ICPUG SE Region

Blank lines can be used to split a listing into 'paragraphs', for easier reading. The effect is spoilt if you have to use a line number with a REM or colon. The following tip creates completely empty lines. Add the following to the end of the previous line. or put it on a line by itself. as in the example:

```
 150 REM ""<delete><rvs><shifted M><unshifted J><return>
```

You soon get used to it! The result will be to force a line feed for every <shifted M> used. If you want several. just repeat that key the required number of times.

[Refers to PET/CBM graphic keyboards, e.g. on 4032 - Ed]

--oOo--

ROBIN BRADBEER IN LAS VEGAS

Information from SE Region

Robin Bradbeer, secretary of the North London Hobby Computer Club and leading light of the ALCC, was in Las Vegas earlier this year for the Consumer Electronics Show, 5-9th Jan. He came back with much amazing information about new releases from Commodore. There is a new Hand Held Computer. called the HHC4, in the style of Sharp/Casio. Fitted with 4K of RAM, it can accept a 12K add-on RAM pac. The liquid crystal display provides one line of 24 characters. The BASIC is thought to be BASIC 2, but with enhancements including LPRINT, CLOAD, CSAVE, PRINT and USING - yes, the last two ARE separate but presumably may be used in series. The device can plug in to the VIC RS232 port. and can drive a TV or monitor. It has an interface to a 24-column printer. The price - a staggering $90!

There are three new versions of the 64. called the SX100 and styled like a pint-sized Osborne. The disk drives are 170K units, but half height. There is a built-in 5" screen. The colour version with two drives costs $1595, about £ 997. The monochrome, single drive version costs $995, about £ 620. Commodore report sales to date of the 64 to be 100,000! The P500 has been renamed, before it even saw the light of day! I suppose none of us should be surprised at things like that by now. Its new name is CBM128. There is a new printer/plotter, using 4.5" plain roll paper, and plotting in four colours with 480 horizontal increments and 990 vertical ones. It will plug in to the VIC or the 64, and will cost only $199.5! A rather advanced speech synthesis system for the 64 is good enough to distinguish female and children's voices. It has the extra BASIC statement 'SAY'.

```
e.g. SAY"ENTER YOUR NAME";A$                 me: barry
     SAY"HELLO"A$                            it: "hello barry"
```

This is expected to cost less than $100, and will surely sell in VAST quantities.

CBM are in the final stages of negotiating to build a new British factory, to make 60,000 units a month of Vics and 64s. 50-60% are to be exported to Europe. Will this mean that Commodore computers will be acceptable to the powers wielding the education grant, and thus be able to compete fairly with that other computer ?

Barry Biddles

--oOo--

FOR THOSE WHO WERE INTERESTED...

After our assistant editor's light-hearted comments on p19 (Jan issue), there was an immediate response from 'Reckitt Household and Toiletry Products' (by appointment). Several members expressed interest in the serious side of the application, so I am pleased to be able provide details supplied by Douglas Rushman.

"We have provided ourselves with a selection of toilets from many countries and in these we can simulate the many varied conditions which our products may encounter. We required an easy means of independently programming each of the twenty-nine cisterns with a sequence of flushes which would repeat every twenty-four hours. The overall program also has to prohibit simultaneous flushing of more than three cisterns at a time in order that the drains should not be overloaded and must also allow for occasional postponement of a flush when adjustments or experimental modifications are being made.

On examining possible options we came to the conclusion that a microcomputer would give us the desired flexibility of program and, moreover, would be the most economical way of doing things. In the event, the CBM 4032 concerned has fully lived up to the expectations and carried out all we asked of it. Over a period of some six months continuous operation, the only matter requiring attention has been a periodical re-setting of the TI$ clock, but we are thinking of adding an external time source to avoid even this."

--oOo--

COMMODORE COLUMN

ZILOG processors for Commodore machines.

CBM and ZILOG have exchanged technology sharing agreements which mean that Commodore will be given the know-how to manufacture the Z8000 16-bit processor, in return ZILOG will be given the manufacture of some specialised Commodore chips. We can expect to see the announcement of at least one new CBM machine with a Z8000 in the coming few months. Commodore also expect to increase its semi-conductor manufacture by 100% in 1983.

Shortage of C64's.

It seems that Commodore were experiencing difficulties in getting production quantities of the 64 machines - this delay being caused by the chip production team 'walking out', according to one source. Dealers were worried that users may impulse-buy other machines, sufficient supplies of the C-64 not being expected until April.

8032 - SK.

The new 8032, and 8096, machines in the new case style with detachable keyboards have been designated SK (Special Kase??) machines. The 'SK' has been written in such a style to be confused with '8K', an unfortunate error - one national magazine (PCW) has already reported that this machine is only 8K. Not true, the machines are 32K and 96K respectively.

Price of the machines : 8032-SK £ 995; 8096-SK £ 1195 (both prices ex-VAT).

Incidentally Commodore intend to continue manufacture of the 4000-series machines, as the integral unit with strength and weight are appropriate to the educational and other special application, markets.

8250/8050 software compatibility.

Whilst the 8050 (single-sided, high density disk drives) are compatible with the 8250 (double sided version of 8050) there is a slight problem with two major packages: Visicalc will not load, and Wordcraft will not access directories.

Software on existing 8050 disks can normally be used by transferring the program(s) onto a freshly formatted 8250 disk. Commodore recommend that this should be done with all software to be used on 8250 drives. The procedure is as follows:

1) Place the original disk in drive 0 and a blank disk in drive 1 of the 8250.
2) Type catalog d0 <return> TWICE. (It must be done twice as a disk error will be produced the first time!).
3) Header the disk in drive 1 with the same name and ID as that of 0.

```
     e.g. header"<disk name>",i<id>,d1<return>
```

where <disk name> and <id> are the same as drive 0.

4) type copy d0 to d1 <return>

after a few minutes your copies are ok. Any problems with Visicalc or Wordcraft are resolved.

Silicon Office also has a problem with the 8250 drive and as it is on an uncopiable disk the above will not work. Bristol Software Factory provide a utility for existing users, and the standard distribution disk will now load on the second attempt.

T.C.

--o0o--

MORE ON REGISTER EXCHANGE

Last month (p61) we presented a utility for swapping the Accumulator and Y register, this time we swap the Accumulator and X register. If you read last months article you will remember that we set the interrupt flag before tampering with the stack pointer and did not restore the flag with an implicit CLI instruction as the PLP will clear it or leave it set depending on the entry condition. We need to do this again with the A/X swap.

The code to do this is ;

```
          PHP     ; save status register
          SEI     ; dont let interrupts harm us
          PHA     ; save accumulator
          TXA     ; now do X
          PHA     ; save X register
          PLA     ; skip stacked X reg
          PLA     ; get old Acc into Accumulator
          TSX     ; fiddle stack pointer
          DEX     ; by one ....
          DEX     ; .. two bytes
          TXS     ;  back into stack ptr
          TAX     ; old Accumulator to X register
          PLA     ; old X to Acc
          PLP     ; dec stack pointer
          PLP     ; restore old status
          RTS     ; finished!!
```

We have now covered Acc/Y reg and Acc/X reg swaps - a simpler case is X/Y swapping :

```
          PHP     ; save status
          PHA     ; save accumulator
          TYA     ; save Y...
          PHA     ; ..onto stack
                  ;
          TXA     ; shift X reg via acc...
          TAY     ; ..now in Y register
                  ;
          PLA     ; bring back old Y..
          TAX     ; .. now in X reg
          PLA     ; restore Acc
          PLP     ; restore status
          RTS     ; finished!!
```

By Andy Scott.

--oOo--

REVIEW

Bob Neill's Book of Typewriter Art

By Brian Grainger

I guess everyone has heard the one about the chap who left school experienced in Maths and Art and found a job painting computers! Well Bob Neill, a professional Hypnotherapist. has since the 1960s been combining artistic talents with techniques of 'greyscale portraiture' to produce pictures from his typewriter. What has this to do with ICPUG you ask. Well, replace the typewriter with a PET and a printer and the techniques are brought into the 80's.

This new book of typewriter art includes the instructions for producing 20 different pictures on your printer. or typewriter if a printer is beyond your pocket. Subjects include the Royal Family, various animals, the classic nude and the Pope !

With the help of ICPUG member Nick Higham a PET BASIC computer program is also included which will produce these pictures on a Commodore computer and printer system. Ideally a printer with variable line spacing is required but I got very good results with my 2023 friction feed.

This fascinating little book, which reads like a set of knitting patterns, gives yet another answer to the question, 'What can I do with a computer', and could well create a new hobby for the more artistically inclined among you.

I would have liked to have seen some words on techniques in creating these types of pictures but all things considered, it is different, interesting and relatively cheap.

The price is £ 5 post free. from The Weavers Press, 4. Weavers Cottages. Goudhurst. Kent. TN17 1BL. Tel: Goudhurst(0580) 211138. Cassette copies of Nick's program will be available (at a modest charge) from the publishers if demand is sufficient.

--oOo--

THE IEEE BUG

By Mike Todd

There has recently been some publicity surrounding an apparently trivial bug in the IEEE handling software of the PET. This bug exists in all PET BASICs prior to BASIC 4.

With the PET receiving data from a TALKing device, such as the disk, the PET first sends a TALK command on the IEEE bus with ATN low to indicate that a control byte is being sent. The handshaking of data bytes into the PET is then performed with ATN set high and the TALKER controlling the DAV line and the PET controlling the NDAC and NRFD lines.

Up to this point, all is well. But when the PET has had all the data it wants (either at the end of a GET# or PRINT# command) it must send an UNTALK control byte. To do this it must behave as a TALKER and so set NRFD and NDAC high to allow the devices on the bus to control them.

This should be done with the ATN line already set low to indicate that a control byte is coming. Instead, the PET sets NRFD high, then NDAC high and finally ATN low and this causes great confusion on the bus.

The TALKing device, which has up to now been sending data whenever NRFD goes high, still thinks that it is TALKing (since ATN is still high) and so procedes to send another data byte onto the bus. At this point, the TALKing device waits for the NDAC line to go high to indicate that the PET has accepted the byte. And what does the PET do? It raises the NDAC line - but not to say that it's read the byte, but in anticipation of sending the UNTALK command. As a result, the TALKer thinks that the PET has read the byte and so the byte is lost to the world.

But this is not the only problem! After setting NRFD and NDAC high, the PET pulls ATN low and all devices immediately stand by to receive a control character. If this happens when there is a "sluggish" device on the bus (such as the

Commodore disk drive) there will be a short period before the last data byte on the bus is removed.

In the meantime, other devices on the bus which are waiting for a control character may take this left over data byte as a control character and act accordingly.

This second problem manifests itself most commonly when using the Commodore disk drives and a non-Commodore IEEE printer (such as the EPSON MX series with IEEE interface). The printer is normally device 4 and the control byte to start it LISTENing is $24. It just so happens that the equivalent byte considered as data is the "$" character, and if the disk happens to be sending this character and the timing is such that it is read as a control character, the printer will start LISTENing to everything that follows!

The result - everything that is being transferred from the disk drive to the PET is also printed (twice!).

If you have an EPSON or similar IEEE printer and a Commodore disk drive, you may well have come across the problem already. If not, you can see the effect quite simply by writing a test file to the disk as follows:

```
10 OPEN 2,8,2,"0:IEEETEST,S,W"
20 FOR I=1 TO 10
30 PRINT#2,"$A$B$C$D$E$F$G$H$I$J$K$L"
40 NEXT I
50 CLOSE2
```

Now, read it back as follows:

```
10 OPEN2,8,2,"0:IEEETEST,S,R"
20 GET#2,A$
30 GOTO 20
```

After running for a while, the printer should suddenly burst into life and start mimicking the data being transferred.

The problem also occurs during INPUT#, but is not so common. It doesn't occur when the PET is acting as a TALKer and the disk or printer are LISTENing in the normal way. This is because the NRFD and NDAC lines are held high anyway and no "spurious" handshaking will occur. Well, that's the problem - what about a solution ?

If you're using BASIC 4 then the solution is already built in - the PET sets ATN low immediately before entering the UNTALK routine. So there should be no problems.

If you're writing in machine code, then include the following code immediately before UNTALKing the bus:

```
LDA  $E840
AND #$FB
STA  $E840
```

If you're writing in BASIC, life is a little more difficult (unless you're prepared to modify your ROMs and include the above routine at $F17F in BASIC 2 or $F17A in BASIC 1). Alternatively, it is possible to force an UNLISTEN after every GET# and PRINT# which will force all LISTENers on the bus to stop LISTENing should they be accidentally activated.

This is done by a simple SYS to location 61827 (or 61822 in BASIC 1) after every GET# or PRINT#. I know it's a bit of a "sledgehammer" solution, but unless you've got BASIC 4, it's the only way I know.

Of course, the best way to do this would be to have a variable SY=61827 and executing SYSSY instead of SYS61827.

This solution won't sort out the first problem, only the second. I don't have a solution to the first since I don't know anyone who has experienced the problem nor have I met it myself, but I'd like to hear from anyone who has.

--oOo--

THE TRUTH OF IT ALL

This program accepts a Boolean expression and creates full truth tables for the expression and its variables. The results can be used to analyse logic circuits.

The routine recognises any single letter as a variable, thus accommodating up to 26 in an expression. The logical operators NOT, AND, XOR and OR are represented by - . @ and + respectively in decreasing order of precedence. This may be changed using parentheses with no practical limit on depth. Variable columns are listed in alphabetical order, with 1=true and 0=false. By adding the line:

```
475 IF S(1)=0 THEN 490
```

the output can be restricted to those lines where the expression evaluates true.

```
BOOLEAN EQUATION -B.A+A.-C+(B@C)
 A     B     C          ANS.
 0     0     0     0
 0     0     1     1
 0     1     0     1
 0     1     1     0
 1     0     0     1
 1     0     1     1
 1     1     0     1
 1     1     1     0
```

```
 10 DIMP(80),S(80),V(26),B(26)
 20 INPUT"BOOLEAN EQUATION ";E$
 30 P=0:S=P:B=P
 40 FORI=1TO26:V(I)=P:B(I)=P:NEXT
 50 IFLEN(E$)=0THENEND
 60 FORI=1TOLEN(E$)
 70  X$=MID$(E$,I,1):GOSUB560
 80  IFX>5THEN110
 90  IFX=-1THEN200
100  ONX+1GOTO120,150,160,160,160,160
110  P=P+1:P(P)=X:GOTO200
120 IFS=0THENPRINT"<rvs> PARENTHESES ERROR.":GOTO20
```

```
130 IFS(S)=1THENS=S-1:GOTO200
140 P=P+1:P(P)=S(S):S=S-1:GOTO120
150 S=S+1:S(S)=X:GOTO200
160 IFS=0THEN190
170 IFX>S(S)THEN190
180 P=P+1:P(P)=S(S):S=S-1:GOTO160
190 S=S+1:S(S)=X
200 NEXT
210 IFS=0THEN240
220 IFS(S)=1THENPRINT"<rvs> PARENTHESES ERROR.":GOTO20
230 P=P+1:P(P)=S(S):S=S-1:GOTO210
240 FORI=1 TO P
250 IFP(I)>64THENIFP(I)<91THEN V(P(I)-64)=P(I)-64
260 NEXT
270 PRINT" ";
280 FORI=1TO26
290 IFV(I)=0THEN320
300 B=B+1:V(I)=B
310 PRINTCHR$(I+64);"     ";
320 NEXT
330 PRINT"<lft>ANS."
340 FORJ=1TOEXP(LOG(2)*B)
350 S=0:FORI=1 TO P
360 IFP(I)=<5THEN380
370 S=S+1:S(S)=B(V(P(I)-64)):GOTO470
380 IFP(I)=5THEN450
390 IFS<2THENPRINT"<rvs> ERROR IN FORMULA.":GOTO20
400 ONP(I)-1GOTO410,420,430
410 S(S-1)=INT((S(S-1)+S(S)+1)/2):GOTO440
420 S(S-1)=ABS(S(S-1)-S(S)):GOTO440
430 S(S-1)=INT((S(S-1)+S(S))/2)
440 S=S-1:GOTO470
450 IFS=0THENPRINT"<rvs> ERROR IN FORMULA.":GOTO20
460 S(S)=1-S(S)
470 NEXTI
480 FORI=1TOB:PRINTB(I);"   ";:NEXT:PRINTS(1)
490 IFS<>1THENPRINT"<rvs> ERROR IN FORMULA.":GOTO20
500 C1=1
510 FORI=BTO1STEP-1
520 C2=INT((C1+B(I))/2)
530 B(I)=ABS(B(I)-C1):C1=C2:NEXTI
540 NEXTJ
550 GOTO20
```

```
560 X=-1
570 IFX$="-"THENX=5:GOTO640
580 IFX$="."THENX=4:GOTO640
590 IFX$="@"THENX=3:GOTO640
600 IFX$="+"THENX=2:GOTO640
610 IFX$="("THENX=1:GOTO640
620 IFX$=")"THENX=0:GOTO640
630 IFASC(X$)>64THENIFASC(X$)<91THENX=ASC(X$)
640 RETURN
```

--oOo--

DISCOUNTS

John Bickerstaff.

Changes from the January issue (p63) include Superscript, Superspell and postage. First class mail goes up in April, so please adjust the value of stamps on your stamped jiffy bags and SAE's.

Superspell no longer has a key, but is protected in a similar way to Superscript. When ordering please send your cheque to me made out to me made payable to Precision Software Ltd. (prices p63). On ordering Superspell to be added to Superscript, please send your Superscript disk to me with your cheque, together with sufficient value of stamps for onward transmission. Do not forget to specify the format: 2040/3040/4040 or 8050.

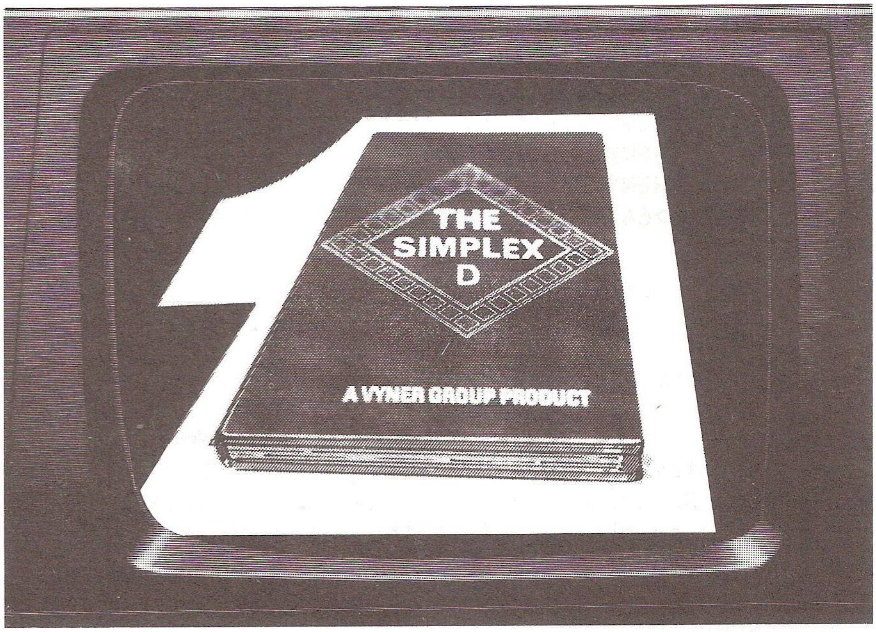
Please note my new address (inside cover and p63, but NOT as p88). Do not telephone after 7.30p.m. weekdays, thank you. Jiffy bags can be purchased from the Post Office.

--oOo--

QUOTE FOR THE MONTH

Be careful that you write accurately rather than much.

--oOo--



The Electronic Cash Book

Micro-Simplex makes Retailers Accounts and Stock Control simple...

Unique features:

- Based on Britain's No. 1 cash book system
- Uses Britain's No. 1 business micro computer
- The only one recommended by Vyners, publishers of Simplex books
- The only one offering all retailers special V.A.T. schemes

Other features include . . .

- Stock control linked to cash registers
- Simple and familiar layouts
- Easy to use
- Automatically produces:
  - (a) Statements to customers
  - (b) Lists of unpaid bills
  - (c) Simple profit and loss accounts

MICRO-SIMPLEX

MICRO-SIMPLEX

OFFICIALLY APPROVED BY COMMODORE

commodore COMPUTER

Contact any Commodore Dealer

or

Micro-Simplex Limited,

8, Charlotte Street West,

Macclesfield, Chesire.

Tel: 0625 615000

SAGA OF A 2031

By Alf Minter

Back in November 1981 L & J Computers told me that the 2031 had landed, and would I like one. With my old-ROM PET it seemed reasonable to upgrade to BASIC 2 and to abandon Computhink in favour of the GENUINE Commodore disk system. So that is what I did.

Trouble came fairly soon. For some unaccountable reason the 2031 did not like to read its demo disk. It also turned out to be unenthusiastic about NEWing a disk, and sometimes did not save programs properly. Quite a lot of this seemed to disappear when I cut an extra write-protect notch in my disks and started using the backs. No explanation, though! The 2031 worked fine on a 4032 PET, and the disks were quite acceptable to the drive in conjunction with this machine. I put all this down, at first, anyway, to my tendency to fiddle with the insides of my PET, and placed my faith in the all-seeing and all-knowing integrity of Commodore and its products.

How wrong I was gradually became apparent. When using Simplicalc, which reads its files by GETting a character at a time, I was getting in a tangle. This was because not every byte was being read. About one in every three hundred bytes seemed to be missing, randomly scattered. A bit of esoteric fiddling with Simplicalc gave me the situation of reading the bytes in pairs, so that if one was lost, the other remained, and a patching-up job was possible. All very tedious.

What really sent me mad was Kevin Pretorius' rather excellent word processor called 'WORDPOWER'. At the time, I was using the original trial version in BASIC, and the way my 2031 left out letters was such that any file was virtually unusable without first combing through a letter at a time. WORDPOWER is now in machine code, and works very well indeed, even with my 2031 and its pranks.

The 2031 went to Pedro Computer Services for checking. Now Mr Pedro really knows his way around PETs, and he could find no real faults. He agreed that the disk drive did not work as it should with BASIC 2, but was OK with BASIC 4. I paid his bill. It so happened that the day I went to see L & J Computers about all this, another customer walked in with exactly the same problem.

Commodore got told, by L & J and myself. How odd, they said. Your dealer should have told you when you bought it. We will try and find out why he didn't tell you. The answer to that is easy. He didn't know because Commodore never told him, or any other dealers, that the 2031 does not work properly on BASIC 2 [see Newsletter p353 Nov. '83 - Ed]. As usual, it is up to the customer to find these things out, the hard way, at his own trouble and expense. After a lot of argy bargy, a modified ROM appeared from Commodore. It turns out that Skyles Electrical Works (USA) and others, know about this bug, and that there is a fix. The fix works fine on PET with 24-pin ROMs. Alas, mine is a 28-pin ROM PET, and you can't fit 24 pins into a 28-pin socket. Wait, they said, a conversion board exists. It is on the way from USA. In due course it turned up. It seems to be the same as that described in the Commodore Newsletter (CPUCN) some months ago. This conversion works, I believe, perfectly well for the character generator ROM. It does not work for any of the other ROM slots as far as I can discover. It certainly did not work in mine.

The behaviour of the 2031 with BASIC 2 summarises as follows:

1 It reads and writes programs perfectly well.
2 It is unsure with >NO, NEW or HEADER.
3 It does not retrieve every byte by GET# or INPUT#.
4 Screen displays saved by machine code come back OK.
5 A program using machine code to GET seems to have more success in retrieving bytes from files.
6 It does not do >VO,Validate or Collect properly. (On one disk it failed to collect up 94 blocks in a nearly full disk.
7 It seems to get on better with the backs of the disks than the fronts [What are you doing using the backs ? - Ed]
8 It seems to retrieve its own written files better than those written by a another machine.
9 Files that have been overwritten by it, on the same disk

many times, come back better than new files, even though the files were overwritten by use of the dreaded '@'.

10 The 28-24 pin conversion board only works for the character generator ROM.

Lastly, the most helpful people in all this were L & J Computers and Pedro Computer Services. They did everything to help that they possibly could. That the problem is still unresolved is not their fault. If anyone else has a 2031 running on BASIC 2 and an upgraded old-ROM PET, I shall be pleased to hear how they are faring.

A L Minter, 2, Whitefriars Way, Sandwich, Kent, CT13 9AD

--oOo--

MEMBERS PRIVATE SALES & WANTS

CBM 3032 for sale £ 400.00, apply to Richard Herriott, or Ramon Osner at Roxburghe International Ltd., 166, High Street, Hounslow, TW3 1BU. Tel: 01-570 5783. Roxburghe also offer various printing facilities to members for return postage and cost of stationery for larger volumes. For details contact Richard or Ramon. Also for sale BASIC80 Interpreter and Compiler with manuals for CBM/SOFTBOX + CP/M. Cost £ 400.00 new 15 months ago; offers to Richard or Ramon, address as above.

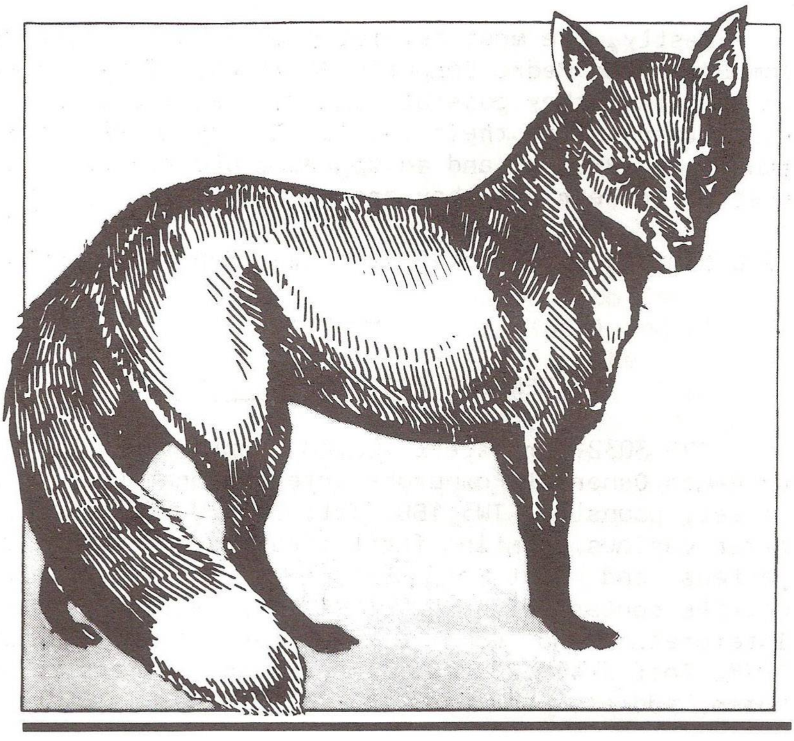
Ex-demonstration equipment: Vic-20, joystick, cassette deck, Facemaker, Defenda, 16K memory expansion, Sargon II chess, Super expander and custom-made case. £ 300, or nearest offer. Please write to Ian Hamilton, 58, Eastcote Lane, South Harrow, Middx. HA2 8DH.

For sale: PET2001-8, very good condition, upgrade ROM (2.0), Programmers Toolkit, many tape$ and much documentation. £ 250 from Ed Carlsen, tel: Eriswell (063 881) 3410 (Suffolk).

--oOo--

FOX ELECTRONICS

FOX ELECTRONICS



FOX ELECTRONICS

FOX ELECTRONICS 141, Abbey Road, Basingstoke, Hants. RG21 9ED

TEL: BASINGSTOKE 20671 (AFTER 6 P.M. - TIM OR JOAN)

ALL PRODUCTS ARE INCLUSIVE OF VAT. OVERSEAS CUSTOMERS PLEASE ADD £2.50 POST AND PACKING.

FULLY GUARANTEED FOR ONE YEAR.

Deliveries 10 days from receipt of order.

FOX ELECTRONICS

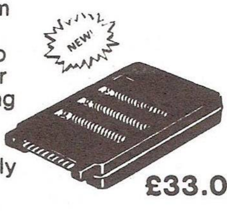
FOX ELECTRONICS

Products for the VIC 20

The VIXEN RAM CARTRIDGE.
for the Vic 20
Switchable between 16K or 8K & 3K. Gives you the option of full 16K RAM or 8K and 3K RAM in one package. When added to the standard Vic gives 16384 bytes of extra memory in blocks 1 and 2 or 3092 bytes of extra memory into the 3K memory block AND 8192 bytes switchable between memory blocks 1 and 3. Simply plugs into the rear expansion port and fully compatible with all motherboards and modules available. No re-addressing of existing BASIC programs needed.

£39.95

TANDEM
Expandable Expansion System, gives 4 expansion slots for Vic 20 cartridges. Custom designed case. Plugs directly into computer. Further expanded by using Tandem System ROM socket. No extra power supply needed.



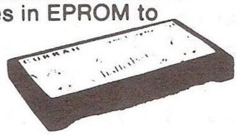
£33.00

VIC LIGHT PEN
A high quality light pen which plugs straight into your Vic with no special interface needed.

£19.50

or for PET 12" Screen **£22.50**

CHATTERBOX. Speech synthesizer with an infinate vocabulary of spoken words out of a number of sound units. Fully programmable and simply plugs into Vic or motherboard. Includes a series of software routines in EPROM to facilitate the programming.



£57.00

Please send me ....

ITEM	QUANTITY	PRICE	TOTAL

Name ....
Address ....
....

CLUB DISCOUNT - £1.00per item.   No P & P

REVIEW

SYSRES Solidus International Corp. (UK) Ltd.

In the beginning there was the 'Toolkit'. In its day, it was wonderful. Later there followed a range of imitations with comparable performance and similar or somewhat extended facilities. With the arrival of the large-screen PET/CBM models which use a CRT controller chip and vectored I/O, further enchancements became viable. The first support chip to exploit this was called 'POWER', and could be considered a second-generation device. SYSRES can be regarded as a development of the genre.

SYSRES comes on floppy disk with the best handbook and documentation that I have seen, even beating the Visicalc manual in presentation. The manual is about 120 pages secured in a smartly covered 3-hole clip binder. It is well-written, clearly laid out and contains a wealth of detailed information on each SYSRES facility.

SYSRES adds over a thousand new functions to BASIC (so the manual claims, for I didn't count) and these are variants on 33 new command words, plus 11 DOS Support commands with seven existing BASIC commands improved. For this reason I can only single out some of the more noteworthy features for inclusion in this review. Here is a quick guided tour of some of the facilities:

The master SYSRES disk is copy-protected but allows the user to make up to three 'boot disks' on to blank unformatted diskettes. This process takes about 10 minutes for each disk and requires access to a dual disk drive. Use of the 2031 is not recommended for this procedure. Loading the 'boot' disk is claimed not to affect the program in memory. but the editor, being an expert at evaluation soon crashed the system. On receiving a '?CAN'T CONTINUE ERROR' an attempt to check that variables were not destroyed by executing a 'GOTO', caused a break to monitor. The program was still intact, but apparently not variables.

SYSRES is not suitable for 8K PETs - it uses about 8K. ?FRE(0) on a 32K machine gave 23549 bytes free when resident in normal memory. SYSRES can be booted up to $9000 up. or 32K or 16K, or 4K of it at $9000 up and the rest in normal RAM.

The auto-repeat key is somewhat unique giving different responses for different key functions. Cursor controls & space take off fast when a key is held pressed (1/3 sec delay with rapid rate). For all others, except return, the delay is 3/4 sec with only half-speed repetition rate. The return key does not repeat. Clever isn't it ? Machines with BASIC4 having the internal auto-repeat have this inferior facility over-ridden by SYSRES.

SYSRES incorporates the now familiar bi-directional scrolling feature of POWER, MICROMON. TRIOS. etc. The handbook claims that this feature works also in the machine language monitor.

MON allows correct entry to the monitor to get the best from SYSRES's features.

The screen editor has a number of enhancements above that normally resident. Editing of text files, e.g. assembler source code is permissible, and for that purpose leading spaces can be accepted. As a consequence of this, formatted listings can be produced automatically. The CBM assembler editor commands GET and PUT are inherent.

SYSRES has a built-in screen-print facility. If the printer is ASCII rather than a CBM model, reverse-video characters are printed underlined, as are shifted spaces.

Auto-shutdown during RUN reduces the effect of SYSRES's presence on program execution speed.

DOS Support & Extensions.

SYSRES gives a choice of several 'wedge' keys, ← > @ ! just to help you feel at home. Computed string expressions may be used in the DOS command message. and the handbook gives some interesting examples to exploit this feature.

An improved '>$0' command prevents accidental addition to program lines the directory line if return is hit. but disk file related commands may be entered before the filename and directory filename quotes and file type will be accepted in the command. Accidental activation of scrolling feature is also prevented. **Crash**: There was a big disaster when a program was stopped and '@$1' entered to get the directory. A fatal system crash occurred. This was repeatable and does not occur with normal DOS Support. Unfortunately I regard this as a very serious short-coming. for it is a procedure that I repeat quite often. (see later -Ed)

The 'save-with-replace' command overcomes the 'now-you-see-it now-you-don't' bug by first performing SCRATCH and then SAVE - hope the power doesn't glitch meanwhile !

Most facilities have the option of output to screen or printer by simply preceding the command with an asterisk. Printouts may also be directed to disk (and in some cases cassette) files for subsequent processing. This includes any BASIC, SYSRES or monitor command.

SYSRES additionally supports non-CBM printers. Device #5 is used to handle ASCII printers and results in the special symbols in the PET/CBM character set being indicated by overstriking. SYSRES even goes as far as overstriking zero with a slash to differentiate it from letter 'O'. and providing paging. skipping six lines for page perforations. The choice of #5 for the ASCII printer can be readily changed by use the SETP command.

The '@L' command sent to disk will enable listing of a program or file to screen. The sequence @L"<filename>"P or PRG will display a program on disk to screen without affecting the one in memory. Using @L"<filename>"S (or SEQ) will display a sequential file and the suffix R or REL will cause a relative file to be displayed in record-by-record format. with record number and contents.

AUTO will not just provide incrementing line numbers. but can include the predetermined program line. e.g.

'DATA'. by including a string expression in the command syntax.

BLOAD performs a binary load from disk, but without disturbing the program pointers. BRUN is similar. but when loaded a JSR is performed to the first address. BRUN allows parameter passing via the CHRGET routine.

CHANGE goes one better than other versions; apart from choice of delimiter. and range of lines. SYSRES permits the change to be constrained to the start of a line. or the end of a line. or the remainder of the line. In addition. pattern matching is possible with '#' as a wild character. All-told there are some 700 valid combinations of the search/replace options. taking six pages of the handbook to cover in depth. With SYSRES it is even possible to send changes to a disk file.

FIND has similarities to CHANGE in its syntax and features.

CLOSE when given without a file number. properly closes all disk files, terminates all logical files and restores normal CMD.

CMD works normally. but does not send 'READY' to the file.

EXEC is a powerful feature which enables a sequence of commands to be entered as if they had been typed at the keyboard. In a program these could be a sequence of responses to the INPUT command.

KEY and KEYS enable one to control the setting of user-defined function keys. The resulting functions can be quite complex.

LIST appears normal at first. but does not output 'READY'. One can, however, pause the listing and later resume. Furthermore listings can be 'formatted' to the output device.

Throughout the handbook, the keys quoted for the business keyboard to control the listing are at variance with the program version reviewed. ':' causes the listing to pause OK, but the '←' key must be used to resume or slow the rate.

MERGE is a true merge (overlay) and complements the APPEND command.

RENUMBER is a selective renumberer, with unreferenced line numbers remaining unchanged, but flagged with '&'. Six pages of techniques using this command are supplied.

The RUN command ignores any garbage on the screen (how often have you entered 'RUNDY.' ?) and turns off the interrupt functions during execution.

TRACE- This sounds as though it might be conventional- but no; SYSRES has a trace routine where the user can specify display of program execution or variable assignments, or both. Imagine- each time a variable changes value- its new value is displayed as one steps through the program. Certain exceptions can be made for GET, READ, INPUT and NEXT optionally- TRACE can be activated under program control and by now you may have guessed that the output can also be directed to a printer. However, in these circumstances the CMD command will not work.

WHY is SYSRES's answer to the 'Toolkit's HELP command. Two interrogation commands are provided. For example if an error occurs when doing a READ of some DATA items and the data has an error- normally the 'ILLEGAL QUANTITY ERROR' will indicate the line number of the READ statement, which is not where the error is ! The WHY command displays the LOCATION that the computer was looking at when the error occurred. WHY? gives the program LINE which the computer was executing. For a DATA error- WHY? gives the READ line- and WHY gives the DATA line- DEF and FN errors can be identified in the same manner.

Conflicts.

Potential conflict with competing utilities, such as PET Rabbit. Toolkit. DOS Support, etc., is avoided. SYSRES simply disables them. SYSRES uses locations 0 to 3 and under some conditions conflicts with the USR function, but such conflict would be rare.

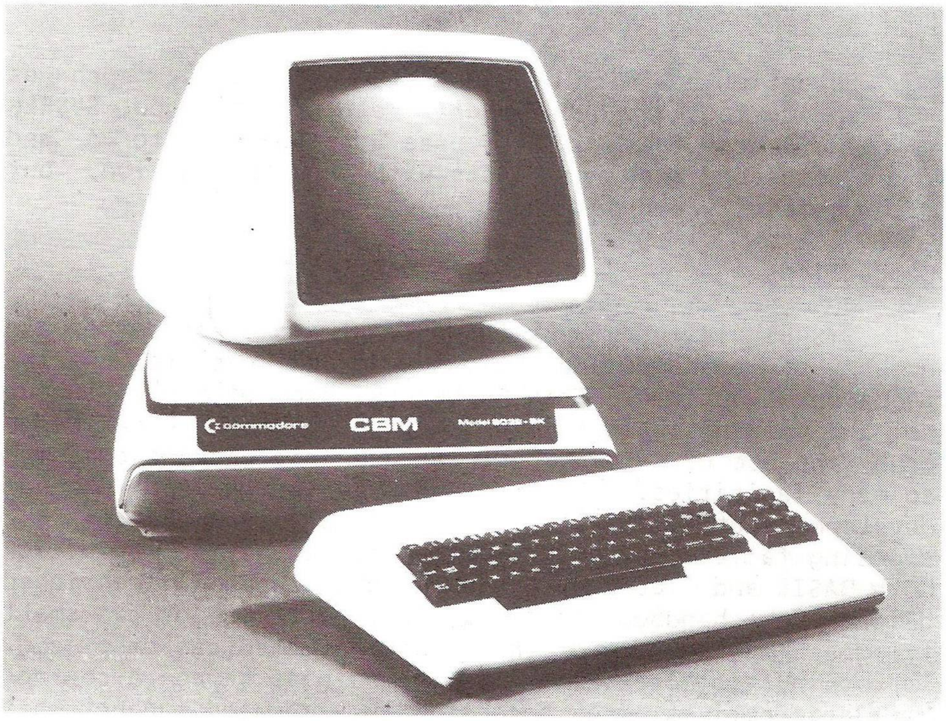
Conclusions.

SYSRES is very well thought out and despite the vast number of commands and options. is it easy to use the fundamental options. This is because many of you will have met the commands before. as in DOS Support/Universal Wedge or by the use of self-evident names. This package contains so many facilities. it supersedes nearly all other program development utilities. with the possible exception of the spooling facilities of 'PLUSDOS'. The utility is useful for both BASIC and machine-code development. As I mentioned before. the handbook is near excellent. One or two small typographic errors and the occasional americanisms, e.g. 'crunching' for 'tokenising'. The business keyboard control key alternatives are consistently wrong and suggest that the key choice has been improved, but the handbook not updated. With a program of such size and complexity, I suppose the odd software bug is inevitable and until I investigate further,* I presume this to be the cause of the earlier-mentioned system crash. Despite this, all credit is due to Solidus for what they describe as 'The ultimate resident program manipulation system for PET/CBM microcomputers'.

The normal 1-off price is £ 59 plus V.A.T. and delivery. but substantial discounts are available from our Discounts Officer. John Bickerstaff. tel: 01-651 5436. Solidus International Corporation (U.K.) Ltd., are at Mill House. Wandle Road. Beddington. Croydon, CRO 4SD. Tel: 01-688 5164.

(* 2nd cassette buffer conflict) R.D.G.

--oOo--



The Commodore 8032 and 8096 professional micro-computers now feature a futuristic sytle housing designed to compulsory IEC specifications.

The housing, made from rigid ABS plastic, is the same design as that manufactured for the "top of range" business series also launched this month.

REVIEW

COMPUTE!'s First Book of VIC. COMPUTE! Books

Following close in the tracks of COMPUTE!'s First Book of PET/CBM' (reviewed p286, Sept '82) comes 'COMPUTE!'s First Book of VIC'. The PET/CBM edition comprised essentially of updated reprints from COMPUTE! magazine c1980. The Vic material is naturally more recent, but this time includes some previously unpublished material. April '82 being the most recent. Some articles stem from the sister journal 'Home and Educational COMPUTING!'.

The book starts with a chapter on introductions; the background story of the Vic's development, introduction to computing for the new-comer, and a few elementary programs involving joysticks and paddles. The chapter ends with Breakout and Pong games programs using paddles. Chapter two has five more games and the remaining four chapters get down to more serious computing matters.

Chapter three is twice the size of any other and is concerned with programming techniques. The remaining chapters cover colour and graphics, maps and specifications, and machine language respectively. The book concludes with an index.

As with the PET version, the book is spiral bound, slightly larger than the Newsletter, but over 200 pages thick. The cover is in colour and is quite stunning. Corrections subsequent to the original text have been incorporated, even details such as changing Basic to BASIC have been attended to.

The book is obtainable from COMPUTE! Books, PO Box 5406, Greensboro. NC 27403 for $12.95 plus $2.00 surface mail or $4.00 airmail. Payment may be made via VISA, MasterCard or American Express. Delivery is about 4-5 weeks.

R.D.G.

--oOo--

❋ Advantages of joining include:-

❋ Access to free software library - you send the disk/tape and pay the return postage.

❋ Advice with programming and hardware problems.

❋ ROM chips (BASMON, PLUSDOS) developed by ICPUG SE members, which take the drudgery out of program development.

❋ Many other utility programs etc.

❋ Discount service for hardware and software purchase.

❋ Entitlement to join one of our 20 plus local groups and meet other members

❋ Hints on using routines in the operating system and interpreter.

❋ Six issues yearly of the ICPUG Newsletter (50 plus pages of articles, information and reviews, etc.) free to members.

❋ Back Issues are available at £1.00 each (Note:- Membership starts with the JANUARY issue.)

❋ Compendium of 1979 - 1980 best extracts plus memory map for original and upgrade ROMs (Basics 1 to 3). A few still available at a reduced price of £2.50 (PET members ONLY)

❋ Machine Code (6502) Course

❋ Opportunity to exchange your personally developed programs with those of other members.

❋ A powerful and recognised (but INDEPENDENT) voice at Commodore.

❋ Member's projects, add-on hardware, for use with the computer (E.G. A modem for telephone line communication.)

❋ Our full support in the event of problems with suppliers.

Join us at ICPUG *Today!*

If you are not convinced .... then contact any one of those listed below and receive details of your local group

Attend a trial meeting - there's no obligation, except perhaps a charge of 50 pence which contributes towards the cost of hiring the hall.

SUBSCRIPTION RATES:-	*United Kingdon*	- *£ 7.50 per annum*
	Europe and Eire	- *£11.00 per annum*
	Overseas	
	Surface Mail	- *£11.00 per annum*
	Air Mail	- *£20.00 per annum*

All subscriptions should be sent to the Membership Secretary at the address given below.

Membership Secretary	Regional Organiser	Technical Representative
ICPUG	ICPUG	ICPUG
30 Brancaster Road	32 Windmill Lane	11 Colison Place
Newbury Park	Southall	Tenterden
Ilford	Middlesex	Kent
Essex 1G 7EP	UB2 4ND	TN30 7BU

ANNUAL SUBSCRIPTION

I enclose a cheque/postal order/money order for the sum of .... which represents my Annual Subscription to ICPUG.

Name:(Block Letters, please ....

Address:(Block Letters please) ....

....

....

Tick User Category ....

PET ☐

VIC ☐

BOTH ☐

Please detach this section and send it together with your remittance.

Printed and distributed by COMPUPRINT COMPUTERS LTD.
4 Sands Road, Swalwell, Tyne and Wear. Tel 0632 888936