"CURSOR"

The official Newsletter of the

COMMODORE COMPUTER USERS GROUP [QLD]

NOVEMBER 1984 VOL.1 NO.5

CLUB ROOMS *MILTON STATE SCHOOL, BAYSWATER ROAD, MILTON*

CONTENTS

Registered by Australia Post - Publication No. QBG3958

Diary for December

Group meeting on Tuesday, 4th December 1984, at 7.30 pm in our club rooms. Visitors are welcome!

QUESTIONS & ANSWERS

Twice postponed, our panel of experts will really be there to answer your questions!

During the business part of our meeting one of our members will run a separate mini-workshop for our Junior Members.

Immediately after the main business part of the meeting is finished there will be a Beginners Corner, run on this occasion by Col Ramsay.

Workshop meeting on Sunday, 16th December 1984, from 1 pm till 5 pm in our club rooms. To get the maximum benefit from the workshop it is recommended that you bring your own computer equipment.

Please note that workshop meetings are for **members only**!

SMOKERS: Smoking is *NOT ALLOWED* in the class rooms! If you must smoke, go to the kitchen or the play ground.

Regional Meetings

Cannon Hill Sub-branch meets every 2nd and 4th Saturday of the month at 7.30 pm, in the Cannon Hill State School. For further information ring Barry Wilson (VIC-20) at 399 6204 or Augy Norman (C-64) at 399 2080, after hours.

Springwood Sub-branch meets on the 3rd Thursday of the month at 7.30 pm, in the Springwood Central Primary School, Dennis Rd., Springwood. Contact Terry Steer at 200 5926 (after hours) for further details.

Pine Rivers Sub-branch meets on the 2nd and 4th Sunday of the month (1 pm - 5 pm) at the Strathpine High School (rear entrance). Ring Clayton Lancaster at 285 4157 (after hours) for further information.

PLEASE NOTE: The above sub-groups intend to have meetings throughout December and January! For further details ring the conveners of the sub-group in question.

SPECIAL INTERESTS GROUPS

Business Sub-Group meets after the main meeting in Milton (first Tuesday in the month) and at the West End State School on the 3rd Tuesday of the month at 7.30 pm. Contact Ken Charters at 341 7222 during business hours for further information.

Primary Education Sub-Group meets after the main meeting in Milton (first Tuesday of the month). Venue for intermediate meetings still to be decided upon. Contact Bill Weeks at 208 8620 (working hours) or at 341 2823 (after working hours).

IMPORTANT NOTICE: Copying of Commercial Software is ***not allowed*** at our meetings or workshops. Failure to comply with this regulation will result in loss of membership!

NEW SUB-GROUPS

With the phenomenal growth rate of our group in 1984 it is becoming more and more necessary to form additional suburban sub-groups.

Based on the club mailing list we feel that sub-groups need to be started in the following areas:

ASHGROVE AREA
NUNDAH AREA
REDCLIFFE PENINSULA
KENMORE / JINDALEE AREA
MT. GRAVATT / SUNNYBANK AREA

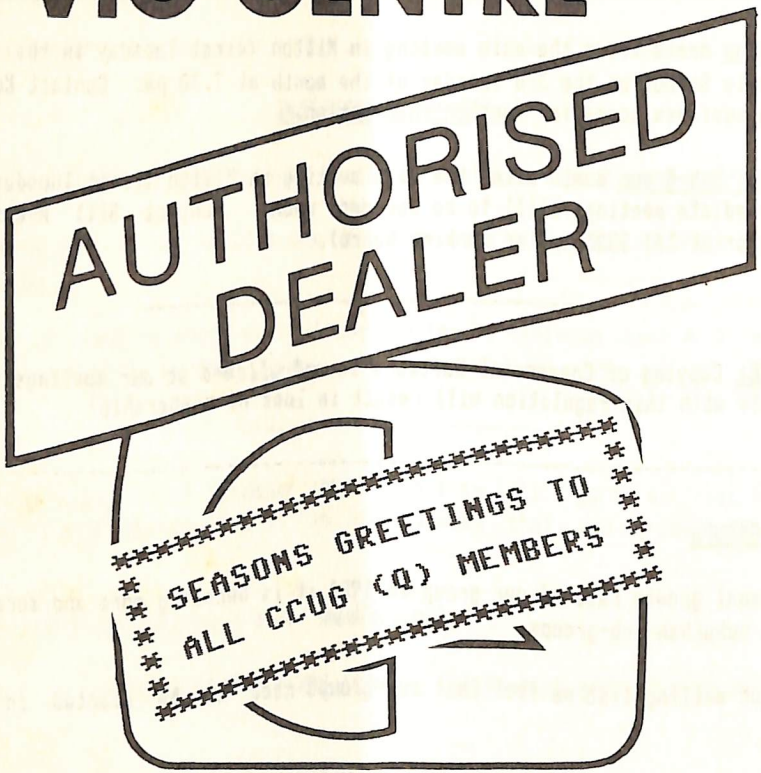
Please note that, beyond organizing a meeting place (preferably a local school), there is very little administrative work involved in running a sub-group.

If you are interested in starting a group in your local area please ring either Norm Chambers or Ralph De Vries for further information. We will only be too pleased to assist you as much as possible in the formation of a group in your district.

CHRISTMAS PARTY

Our annual X-mas party will take the form of a picnic this year. It will be held on Sunday, 9th December from 10 am till 5 pm at Wellington Point. Bring the whole family!!! For further details ring Max Bean on 208 1225, after hours.

THE VIC CENTRE



commodore COMPUTER

For professional service and support call in to CW Electronics.
Commodore's longest established Queensland Dealer.
Largest range of VIC & C64 software in Queensland.

THE ONLY PLACE WORTH ITS SALT IN BRISBANE

416 LOGAN RD., STONES CORNER BRISBANE 397 0888

EDITORIAL

And here is our last newsletter for 1984, and our biggest one so far. The next issue will appear in January 1985.

For the benefit of our many new members it is pointed out that no meetings are held during January. Our normal meeting schedule resumes in February. Details as usual in "CURSOR".

During this year Commodore has certainly come of age. We have seen the establishment of a Brisbane office by Commodore with a very helpful staff, and also the fact that C-64's can now be purchased by State Schools. This is of course largely due to the tremendous increase in software supplies for the C-64, both quantity - and quality wise.

Our group has also improved quite substantially, with more members, and better facilities and services. We feel that 1985 will bring further improvements. Please members, don't be afraid to speak up, if you feel that the group can be improved in this or that area. Your committee needs your input. Don't be like one of our ex-members who never contributed anything to the running of our group, but decided to form his own computer club. This is a free country of course, but it seems such a waste of resources.

This year we were fortunate enough to have a stand allocated to us at the computer show, which was held at the Crest Hotel from 8th to 11th of November. The stand was constantly manned by members of our group, and we dealt with a great many inquiries about the group and Commodore in general. Although it was a bit tiring on the feet, it was felt that it was a very worth while excercise, which should result in substantial membership increases.

As regards equipment we are seeing the gradual demise of the VIC-20 and the dramatic rise of the C-64. In 1985 we should see the introduction of the C-16 as a VIC replacement, as well as the new Plus/4. Most commentators have mixed feelings on the Plus/4, particularly as regards the built-in software, which seems to be totally inadequate. Another rumour has it that Commodore is about to unveil a 128K 80-column computer, which is compatible with the C-64, as well as a very sophistcated 32-bit computer which does just about everything except boil an egg. Regrettably they are only rumours though.

All in all it should be an interesting year for Commodore and our group.

In closing I would like to express my sincere thanks to all those who have contributed articles to this newsletter, and to our advertisers who have supported us throughout 1984.

Ralph De Vries

NOTES & GOSSIP

CHRISTMAS GIFT SUGGESTIONS

As a Commodore computer owner there should be no scarcity in gift suggestions this year, although we have found that with the pending disappearance of the VIC-20 there are few if any new products for this computer now. If you are a VIC owner it pays to visit the show rooms of our advertisers to see what is still available. C.W. Electronics seem to have the widest range of VIC software.

Following here are some suggestions for C-64 users. Please support our advertisers - after all, with your current membership card in your pocket, you will be able to obtain your club discount as well. Another important point: start your computer shopping early. If you leave it till the last moment chances are that your favourite program will be out of stock. You are warned!

HARDWARE

If you are the owner of a Datasette a 1541 Disk Drive should be on the top of your shopping list. Stocks are reasonable again, and despite the fact that many harsh words have been spoken about this vital piece of equipment, it appears that current stocks are a lot more reliable. If you intend to use your C-64 mainly for business purposes it might be worth while to consider the MSD dual drive system. (Appr. $1200). Solidly made, and used in conjunction with a parallel interface, it will give you much faster loading and no swapping of program - and file disks anymore. A word of warning though: certain programs which use new techniques of program protection will not load with this drive!
Printers are next on the list. Commodore now make the 801, 802, and 1101 printers. Just released as well is Commodore's CPS801; a 4-colour printer. No further details known at this stage. All these printers connect directly via the serial port to your computer. If you should decide to buy a non-Commodore printer you will have to purchase a separate printer interface such as the Cardprint +G, which cost appr. an extra $100.00. Choice is very wide of course, but one of the most popular printers is the Star Gemini-10X, a very good workhorse indeed.

Joysticks are very plentiful at present. Remember though that quality goes hand in hand with price. The WICO range for example are expensive, but they will outlast cheap ones many times over.

Another very popular item is the Koala Pad, which now sells below $100.00
The group has just aquired a special printer program, which allows Koala Pad pictures to be printed out on 1525/801, 1526, and several non-Commodore printers such as Gemini, Epson etc.

SOFTWARE & BOOKS

Here the choice is positively overwhelming. Following are just a few suggestions, but see your computer dealer for the latest releases.

In business software we would just like to mention those programs which we have tried out recently. The Multiplan* spreadsheet is expensive (appr. $150.00), but is just about tops with it's 400-odd page instruction book. Two data base programs tested include The Consultant* ($179.00), first rate with a very good instruction book, and The Manager* ($100.00), which is just about as good, but with woefully inadequate documentation for newcomers to this type of program. Still, there is rather a substantial difference in price!
The most popular wordprocessors for the C-64 are Easy Script and Paper Clip, both appr. $100.00.

C.W. Electronics market a complete business accounting package for under $250.00, thus making it possibly the best value for money. It consists of the Tot'l Business 3.6 Program and the Tot'l General Ledger. The Business 3.6 Program (upgraded version) was very well reviewed in the Australian Commodore Magazine of Oct. 1984.
To get a similar series of business programs in other brands will cost you at least twice as much.

In games the choice is so huge, that all we can do is mention just a few which we have seen recently. From Epyx we saw Impossible Mission* ($49.95 on disk), a rather splendid game with excellent graphics, voice synthethis et all. A real beauty! Imagineering are now distributing "Electronic Arts" software in Australia. All disks are $60.00 each, which is not cheap, but some are rather better than the average. Some that we have seen include Music Construction Set, Archon, and Pin Ball Construction Set*. This last one is quite interesting, because it allows you to design by means of "Icons" your own pin ball layout. A very clever bit of programming indeed. Ozisoft are distributing the range of Kawasaki music synthesizer programs. This includes an overlay keyboard, and some half dozen disk-based programs to go with it. We have not previewed this series, but we can promise you that you won't get much change out of $200.00 should you decide to buy the complete range.

In books the choice is so great, that we can only give you some general guide lines. In general look for books published in 1984! Some of the earlier books were rushed onto the market, thus in many cases the full potential of the C-64 was not developed as yet, and mistakes in listings were also rather common.
The major publishers of books on Commodore are Prentice Hall and their subsidiary Robert Brady, as well as COMPUTE! BOOKS, the book publishing subsidiary of Compute's Gazette, distributed in Australia by Holt-Saunders Pty Ltd.
See the advertisement of The Bookshelf Library Supplies in this newslettter for special club prices on books.
As well CW Electronics distribute the Abacus books, reviewed in previous issues of this newsletter.

* These programs were kindly lent by Chandlers Pty Ltd , 43 Adelaide St., Brisbane, for review purposes.

Contrary to previous reports the Computer House of 50 Queen St. does not give discounts to members of our group.

Mermaid Computers of 2518 Gold Coast H'way, Mermaid Beach, 4218 offer a discount to members of our group on production of a current membership card.

Please Note: If you wish to hire the group's 1526 printer, please contact Roger Haigh at 339 8037, after hours.

Our new Disk Librarians are Ken Charters and Max Bean. Members wishing to buy Public Domain Software or borrow Commercial Disk Software should write to Ken Charters at 16 Westgate Ave. Springwood 4127.

In last month's issue of "CURSOR" there was a review of CBS educational software by Dale Gilbert. We omitted to mention that our review copies were supplied by Chandlers of 43 Adelaide St. Brisbane.

Welcome to the Rockhampton Commodore Users Group, which is now up and running.
Their address is P.O.Box 6033, North Rockhampton, 4701.

A Programmers Register is to be established by this group. Any member with expertise in Sound, Graphics, Machine Code, Business Applications programming etc. should contact Greg Perry, to have their name included in this register.

Track 18 — Part 1

To provide a better understanding of the housekeeping functions in a 1541 disk drive, it is necessary to understand some of the things that go on in Track 18. Before starting, I will assume that you know a little about the structure of 8-bit bytes. If you don't, read pages 215-219 of the Programmer's Reference Guide (PRG) first. As far as possible I will use decimal notation in these notes.

To start with, let's look at the 'front page', Track 18 Sector 0. If you didn't know already, each sector (or block) on a 1541 disk can store 256 bytes. The usual convention is that the first two bytes on any sector hold the 'chain bytes' to the sector that follows the one being looked at. It happens (by the design of the disk operating system, DOS) that Track 18 Sector 1 follows Track 18 Sector 0, so the first two bytes on Track 18 Sector 0 will be '18' and '1'. That eliminates 2 bytes - let's go on to the other 254.

A picture would probably help, so let me put down in tabular form the sort of thing you will find in the first 144 bytes of Track 18 Sector 0. All values are in decimal.

Byte#	0	1	2	3	4	5	6	7
0	18	0	65	0	21	255	255	31
8	21	255	255	31	21	255	255	31
16	21	255	255	31	21	255	255	31
..								
..								
64	21	255	255	31	21	255	255	31
72	17	252	255	7	19	255	255	7
80	19	255	255	7	19	255	255	7
..								
..								
96	19	255	255	7	19	255	255	7
104	18	255	255	3	18	255	255	3
112	18	255	255	3	18	255	255	3
120	18	255	255	3	17	255	255	1
128	17	255	255	1	17	255	255	1
136	17	255	255	1	17	255	255	1

That odd assortment of figures is the Block Allocation Map (BAM), which is a critical part of disk management. The BAM is where DOS stores information about which sectors are in use (so can't be used right now for any fresh storage) and which are not (so are available for use).

When a disk is NEWed, there are 664 sectors available to the user for storage. So how can information about 664 sectors be stored in 142 (remember the chain bytes!) locations in the BAM?

The answer is that the information is encoded in a cunning way.

Any 8-bit byte in the Commodore 64 can store any number from 0 to 255, 256 numbers in all. The eight bits that make a byte take on a special meaning in the BAM. Starting from byte #4 (see above), each group of 4 bytes 'remembers' the contents of a track. Bytes 4-7 look after Track 1, 8-11 do the same for Track 2, and so on. Bytes 72-75 look after Track 18. See if you can work out which group looks after Track 26 (no prizes!). The bytes are used like this.

The first byte in any group of 4 is the big picture - how many sectors are free in the particular track. In this case, Track 1 has 21 (i.e., all) sectors free. The BAM above reflects a freshly NEWed disk, so all blocks (except in Track 18) are empty. As the disk has been NEWed, some of Track 18 is being used. This is reflected in bytes 72 and 73.

Notice too that the number of sectors decreases as we read from Track 1 (21 sectors) to Track 35 (17). Commodore has taken advantage of the longer tracks at the outer edge of the disk to increase the number of sectors in them.

Now the detailed picture. The next three bytes in each group of four mark precisely which sectors are in use. If we stretch out the 3 bytes as 24 bits, we have this:-

```
     Byte 5=255          Byte 6=255          Byte 7=31
   x x x x x x x x   x x x x x x x x   x x x x x x x x
   1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1   0 0 0 1 1 1 1 1

Sctr 7 6 5 4 3 2 1 0   15  13  11   9     spare  19  17
                         14  12  10   8            20  18  16
```

Each bit represents 1 sector, numbered as shown here. If the bit is 1 ('on') then the sector is empty. A '0' means it is full. The logic of '1' meaning empty is the inverse of what you might expect, but it makes DOS more efficient. This shorthand coding packs a lot of information into a small space. Just for fun, what would be in byte 5 if sectors 2, 3 and 4 were full?

The next part of Track 18 Sector 0 holds the disk name (which can be no longer than 16 characters). If you use a shorter name, DOS puts CHR$(160) in the unused locations. We now have:-

Byte#	0	1	2	3	4	5	6	7
144	T	E	S	T	D	I	S	K
152	1	9	8	4	+	+	+	+

For simplicity, I have shown the disk name in plain English, but in fact it would be stored in ASCII (see Appendix C of PRG if this is a new one on you), and I have used '+' so show the locations where CHR$(160) will be used to 'pad out' the name to 16 characters.

The next two bytes (160 and 161) are always CHR$(160) or $A0 if you are into hexadecimal notation. By way of explanation, CHR$(160) is usually termed 'shifted space' - not that it's gone anywhere, but it is the character you generate if you hold down the SHIFT key while pressing the SPACE bar.

Bytes 162 and 163 hold the disk ID that you selected when NEWing the disk. Byte 164 is another shifted space, while 165 and 166 hold the DOS version, which is 2A in the case of a 1541 drive. You see this at the right-hand end of the reversed 'header' line every time you read the disk directory. The last bytes used are 167 to 170, more shifted spaces. Bytes 171 to 255 are usually filled with zeroes, but from time to time you may find other odd bits in there which have no formal part to play in the operations of Sector 0.

For the most part, DOS will do all the housekeeping needed in Sector 0. The only time you need to help it out is when a 'starred' file appears when you list a directory, something like:-

```
10  "GUNGADIN"     *PRG
```

This means the file GUNGADIN is somehow corrupted, and DOS needs help from you. Before doing ANYTHING else, VALIDATE the disk to correct the BAM and erase the corrupted file.

DOS also takes care of other matters. When you read or write a disk, DOS reads the BAM into its own memory. Each subsequent time you use the disk, DOS checks the ID again. If the ID has not changed from last time, DOS doesn't reread the BAM. It is for this reason that you are warned (and warned and warned...) against using duplicate ID's. If you use a mainly empty disk, then use a nearly full disk having the same ID, the DOS will remember the original BAM, and write to the second disk based on the free sector pattern of the first. The nearly full disk will probably not have free sectors where BAM thinks it does - disaster.

You may interfere with Track 18 Sector 0 to some extent. The most common feature is to change the disk name. This is done by using an advanced disk command to write a new name to bytes 144 to 159. There are utilities about that automate the process for you.

You may also alter the disk ID, but be warned. The ID is also written in many other places on the disk where you can't see it (or, for that matter, alter it). If you do change the ID in Sector 0, it will be only a cosmetic change. DOS will ignore this change when deciding the true ID of your disk, with possible disaster as discussed above. If you are in any doubt about the true ID of a disk, try this little routine, which reads the true disk ID stored in 1541 disk RAM.

```
100 OPEN 15,8,15,"I"
110 FOR X = 0 TO 1
120 PRINT#15,"M-R";CHR$(22+X);CHR$(0)
130 GET#15,A$ : A$ = A$ + CHR$(0)
140 IF X = 0 THEN PRINT "TRUE ID = ";
150 PRINT A$;
160 NEXT : PRINT : CLOSE 15
```

Track 18 - Directory

We now move on to the actual directory entries. Storage of these starts on Track 18 Sector 1. Commodore drives use an optimal search technique to save time during disk access. The selection algorithm used means that the sequence of sectors that follow Sector 1 goes something like Sectors 4, 7, 10, 13, 16, 19, 2, 5, 8....etc. In this way, the DOS doesn't have to wait a full disk rotation to find the next sector it wants. Quite nifty.

Sectors again hold 256 bytes, 254 when the link bytes are subtracted. Eight file entries are stored in each sector. The first entry uses 30 bytes, and the next seven each use 32. As before, a picture would help.

Byte#	0	1	2	3	4	5	6	7
0	18	4	130	17	0	T	E	S
8	T	F	I	L	E	1	9	8
16	4	+	+	+	+	0	0	0
24	0	0	0	0	0	0	5	0
32	0	0	129	19	0	T	E	S
40	T	D	A	T	A	1	9	8
48	4	A	+	+	+	0	0	0
56	0	0	0	0	0	0	10	0
64	0	0	132	22	5	R	E	L
72	A	T	I	V	E	F	I	L
80	E	+	+	+	+	12	14	100
88	0	0	0	0	0	0	60	0

Let's stop there, before we run out of page. The pattern of each entry repeats itself in blocks as shown between the dotted lines, so let's look at that pattern. Please note that my description of byte numbers will differ slightly from that found in Commodore documentation (eg, page 56 of the 1541 Manual) for reasons of clarity.

Skip over the first two bytes in each panel in the table. I take each of these panels as being a directory entry block, but Commodore don't seem to think the first two bytes are!!. Apart from the link bytes before the first entry, the two bytes (32, 33, 64, 65 and so on) are unused.

The next byte in each entry is the file type, which are numbered 0 to 4:-

File#	Type	Abrvn
0	Deleted	DEL
1	Sequential	SEQ
2	Program	PRG
3	User	USR
4	Relative	REL

When the file is being written to disk, the DOS uses values 0-4 for internal management. When the file is closed properly, the file# has 128 added to it to indicate a successful 'save'. Byte 2 shows a program file, byte 34 a sequential file and byte 66 a relative file. If any file is SCRATCHed, the type byte should revert to 0 (DEL) but it may not always, despite this being the intent of DOS. Provided the byte contains less than 128, DOS seems to ignore the error and take the file as being erased.

The next pattern pair (bytes 3, 4, 35, 36 etc) tell DOS where the actual file storage starts. The first sector of TESTFILE1984 is stored at Track 17 Sector 0, TESTDATA1984 at Track 19 Sector 0, and RELATIVEFILE at Track 22 Sector 5. Using this information, DOS can go to the file that has been chosen. From there, it reads the first 2 bytes of the first sector to find the second sector, and so on.

Bytes 5-20, 37-52, 69-84 and so on are the filenames given to the specific files. As with the name of the disk on Track 18 Sector 0, the names are 'padded' out to 16 characters with CHR$(160). SCRATCH does not erase these characters.

For most files, there is nothing more until bytes 30-31, 62-63, etc. These bytes store the number of blocks or sectors used to store the file. The bytes are in low byte-high byte order, so byte 30 holds the low byte and 31 the high byte. The total number of blocks is thus (byte 31 * 256 + byte 30), or the correct byte numbers for the file in question.

Relative files use 3 bytes to store information peculiar to the file type. If all the files in the example were REL type, bytes 23, 55, 87 etc would hold the length of record selected when the file was set up. Byte 87 above indicates a length of 100 bytes per record. REL files also use a sequence of sectors ('side sectors') to assist fast access. Bytes 85 and 86 indicate that the side sectors for this file start at Track 12 Sector 14.

Two other bytes are used for 'save with replace', such as when you use SAVE"@0:MYFILE",8. This syntax has a mixed reputation, but if you do use it, it works like this. DOS firstly writes a new file in some free disk location, remembering the track and sector of the first block it uses in bytes 28-29, 60-61 etc. Then, if the SAVE was successful, the information in bytes 28-29 is transferred to bytes 3-4 to be the pointer to the new file, and bytes 28-29 are made zero again. Remember this if for some reason the syntax causes problems. If you know where the old or new file is stored, you may be able to manipulate the directory entry to recover the situation.

There are some useful routines to handle the directory - probably more than you will ever need. Keep an eye out for them in journals and magazines.

Paul Blair

This has been the last article in a series of articles on the 1541 Disk Drive by Paul Blair of Canberra.
These articles are required reading for those users who have never used disk drives before, as Paul has managed to clarify those points which were never properly explained in the 1541 manual. If you have missed any in this series you will be pleased to know that early in 1985 the complete series of articles will be published in booklet form.

The Editor

THE COMMITTEE OF THE

C=OMMODORE C=OMPUTER

USERS GROUP [QLD]

EXTEND

SEASONS GREETINGS

TO ALL OUR MEMBERS

Commodore Screen Handling

One of the factors which often distinguish a good professional program from one written by the average user is the effective use of good screen formatting. Many programmers write brilliant programs and then spoil them by displaying the results in a matter-of-fact manner. In this article, we shall look at just some of the more advanced methods of setting up and displaying information on the screen of the VIC and C64, not forgetting the Commodore 4000/8000 series. I am not talking about sprites, fancy graphics, or multicolour displays, but simply methods of displaying information on the screen in an ordered manner that is both pleasing to the eye and clear to the user. Don't be put off by the word advanced; the concepts are easy to learn and use in everyday programming.

First, allow me to suggest a few general rules for formatting the screen output. Here we have a small problem in that every programmer has their own ideas, however, for the sake of this article, let us agree on the following

1. Reserve the top 1-5 lines for screen headings. Use these to display the program name, main menu heading and any sub-headings as required.
2. Line 5 is for column titles maybe.
3. Reserve the bottom three lines (23-25) for error or other messages. The user then always knows where to look for information.
4. Never allow the screen to scroll under its own accord. (Maybe use smooth scrolling if it is required?) When printing a long list of information, display it one screen at a time, in sections, and preferably with an option between screens which allows the user to move backward or forward or abort the procedure.
5. Always display ALL the title or field names on the screen before inputting any information. This shows the user what is expected for each entry and minimises confusion.
6. Don't use too many colour changes, two, maybe three are enough. More than this rapidly becomes annoying. Use colour to highlight special features not to show off how many different colours you can program. Pick suitable colour matches for screen and border (often the same), and character colours that are easy on the eye and STICK TO THEM except for special occasions.

Ok, now let's look at how to implement some of these ideas with particular reference to screen positioning.

Many versions of the BASIC languages used on micros have included a PRINT AT facility where the line and column for input or output of screen information can be specified. With most of the Commodore machines so far, such a direct command has not been part of the BASIC language. However, since the operating system prints the next character at the current cursor position, if we set this to the desired position first, we can print anywhere on the screen, in any order, at any time. This is the key to good screen handling.

There a several methods we can use. Let's look through the three most common ones and you can chose the one which suits you best.

The oldest and most universal method, which works on all machines without alteration (except for minor changes for the screen width of the VIC or the 8000 series), is to use two strings of cursor movements such as

```
V$="[DOWN25] : H$="[RIGHT40]
```

and then use the LEFT$ function to select the required number of cursor movements. For example to print "HELLO" on line 10, starting at column 10, we could use

```
PRINT "[HOME]"; LEFT$(V$,10); LEFT$(H$,10);"HELLO"
```

For the horizontal positioning to a specific column, it often may be easier to use the the old faithful TAB(10) instead. Only problem is that with the TAB command we cannot go backwards along a line.

The second method also works on all machines, but requires different values for the different versions of BASIC between the PET 2000, 4000/8000 and VIC/C64 machines. Since the computer itself has to keep track of the current position of the cursor, all one has to do is fool the operating system into putting it where we want. If we know where to look that is!

There a number of locations in the lower memory of all Commodore machines which control the cursor. (Take a look at a good memory map.) These not only control the current screen line and column, but also control whether the cursor should flash (useful for getting the cursor to flash on GET statements), the current character under the cursor, the current character colour, and many others. For now let's stick to the point. The cursor line and column are contained in the locations

CBM	2000‡	4000/8000	VIC/C64
Line	245	216	214
Column	226	198	211

‡‡ Does anyone out there still use an early PET with 'Rev. 2' BASIC 1.0 ?)

All we now have to do is set these locations with the required values (with POKE statements) right? Unfortunately, its not quite that simple. Try the following. Clear the screen and enter the following line. (NOTE: for other than the VIC/C64 change the locations according to the table)

```
POKE 214,10: PRINT "HELLO": PRINT "PAUL"
```

What happens?. HELLO is on the second line and PAUL is on the twelfth!

The problem lies in location 214. This contains not the current line, but the next line instead. One other problem, screen lines are numbered from the top as 0,1,2... etc. Therefore use as follows:

```
POKE 214,L-1: PRINT : POKE 211,C
```

where L is the required line and C the column. For example,

```
POKE 214,9: PRINT : POKE 211, 10: PRINT "HOW'S THE WEATHER"
```

The blank PRINT after the POKE of the screen line is VITAL. You cannot use PRINT "HELLO" or the like. Also, since it is not possible to POKE a negative value, this method will not allow one to position to the top screen line - that's what PRINT "[HOME]" is for anyway! There are several advantages with this method as we shall see later.

The third method only works on the VIC/C64 as far as I know. The KERNAL (the set of operating system instructions in higher memory) has a built in routine called PLOT which can be readily accessed through machine code to set the cursor position or read the current cursor position. (Maybe this was originally to provide an a BASIC 'PRINT AT' command?) It can be used to set the cursor as follows

```
POKE 781, Line
POKE 782, Column
SYS 65520
```

For example,

```
POKE 781,5: POKE 782,10: SYS 65520
PRINT "BIT COLD IN CANBERRA EH PAUL?"
```

WARNING: With both methods 2 and 3, NEVER set the line number greater than 24 or column greater than 79 or the computer may crash. Try it and see. (The computer actually keeps a list of the addresses (memory locations) of the start of each screen line therefore setting the line to greater than 24 upsets this sequence with strange results.)

From here on, let us use method 2 (since it's my favourite!) to demonstrate some examples of screen positioning. Unfortunately, I'm running out of space in this article, therefore, you should enter the following, RUN them to see what happens, and if you do not understand any points drop me a line.

Example 1: With the TAB command it is impossible to print leftwards of the current cursor position. With location 211, this is easy, allowing one to do some things that are quite difficult to do by other methods.

```
100 PRINT "[CLR]USING NOTHING"
110 FOR C=39 TO 0 STEP -1: PRINT C;: NEXT
130 POKE 214,5: PRINT : PRINT "USING TAB"
140 FOR C=39 TO 0 STEP -1: PRINT TAB(C)C;: NEXT
150 POKE 214, 15: PRINT : PRINT "USING POKE 211"
160 FOR C=39 TO 0 STEP -1: POKE 211,C: PRINT I;: NEXT
```

Example 2: General screen positioning

```
 10 FOR I=1 TO 1000
 20 L= RND(1)*22:C=RND(1)*35
 30 GOSUB 100
 40 PRINT "HERE"
 50 FOR J=1 TO 500: NEXT
 60 GOSUB 100
 70 PRINT "    ": REM ERASE MESSAGE
 80 NEXT : END
100 REM POSITION CURSOR
110 POKE 214,L: POKE 211,C: RETURN
```

Example 3: One can use this method effectively for positioning the cursor for INPUTs. Only now, to allow for the '?' and space printed by the INPUT statement, the cursor should be positioned 3 characters to the left of the actually desired position.

```
100 LN$="[CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC]"
110 REM PRINT SCREEN HEADING
120 PRINT "[CLR,SPACE10]ENTER PERSONAL DETAILS"
130 PRINT LN$
140 REM POSITION TO 6TH LINE FOR MAIN INFO
150 POKE 214,5: PRINT
160 REM PRINT FIELD TITLES
170 PRINT "NAME      : ";A$(1)
180 PRINT "ADDRESS 1 : ";A$(2)
190 PRINT "ADDRESS 2 : ";A$(3)
200 PRINT "POSTCODE  : ";A$(4)
210 PRINT "PHONE     : ";A$(5)
220 REM RE-POSITION CURSOR FOR INPUTS
230 POKE 214,5: PRINT
240 FOR I=1 TO 5: POKE 211,10: INPUT A$(I)
250 NEXT
260 REM CHECK IF ALL CORRECT ON LINE 23
270 POKE 214,22: PRINT
280 INPUT "ALL CORRECT[SPACE3]N[LEFT3]";A$
290 IF A$<>"Y" THEN 130 :REM NO ? THEM TRY AGAIN
300 END
```

Notice how efficient this method is when you have to re-enter any fields. The cursor is repositioned to the beginning of the info on the screen. The field can be edited with the INST/DEL keys and (or) RETURN pressed to enter the correct field.

Example 4: Both the screen line and cursor positions can be PEEKed to see where one actually is on the screen so as to avoid scrolling the screen or other problems which will affect our nice screen format. For example, printing a long list of random numbers by 'screenful'

```
 10 H$="[CLR,SPACE10]EXAMPLE 1"
 20 LN$="[CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC]
 30 GOSUB 130
 40 FOR N=1 TO 200
 45 REM CHECK IF CURSOR PAST LINE 19
 50 IF PEEK (214)>20 THEN GOSUB 100
 60 PRINT N, RND(1)
 70 NEXT : END
100 REM POSITION CURSOR FOR MESSAGE
110 POKE 214,22: PRINT
120 INPUT "PRESS RETURN FOR NEXT PAGE";A$
130 REM PRINT SCREEN HEADING AND PAGE NUMBER
140 P=P+1
150 PRINT H$: PRINT LN$: PRINT "PAGE :" P: PRINT LN$
160 PRINT "NO","RANDOM": RETURN
```

These examples cover the main applications but I'm sure you can thing of many more. One last one which I include in my programs is a routine I call 'Idiots' - a subroutine to print a message on line 22, only accept a 'y' or 'n' answer, then clears the line and returns from whence it came. (Can also include sound or colour changes to good effect.) It needs to have a string, for example SP$ set to be 39 spaces before it is used. It also turns the cursor flash on for the GET statement. This is done in location 204 (167 for 4000/8000 machines) but is a bit suspect for the VIC/C64s depending on the screen colours.

```
     10 SP$="[SPACE39]
* other lines of program eg
     20 GOTO 200
*
     50 REM IDIOTS
     55 REM POSITION MESSAGE, C FLASH ON
     60 POKE 214,22 : PRINT : PRINT M$;" Y/N "; POKE 204,0
     70 GET A$: IF A$ <>"Y" AND A$ <> "N" THEN 70
     80 REM C FLASH OFF, RE-POSITION AND CLEAR
     90 POKE 204,1: POKE 214,22: PRINT :PRINT SP$: RETURN
*
    200 REM SET UP MESSAGE AND CALL IDIOTS
    210 M$="IS IT REALLY COLD IN CANBERRA"
    220 GOSUB 50
    230 POKE 214,10 : PRINT
    240 IF A$="N" THEN PRINT "I BET IT IS!"
```

Greg Perry

MEMBER'S ADVERTISEMENTS

For Sale - at bargain prices !!!

CALCRESULT - EASY (Spread Sheet on cartridge for C-64) -	New Price $100.00 - Sacrifice	$30.00
MAGPIE - (Database on cartridge for C-64) -	New Price $150.00 - Bargain at	$30.00
JUMPMAN Jnr. - (Game on cartridge for C-64)	New Price $ 45.00 - Gift at	$15.00

Contact Ralph De Vries at 303477

For Sale: Commodore 1526 Printer $325.00

Contact Mike Levine at 370 9598, after hours.

For Sale: Commodore 801 Printer $250.00

Contact Col Ramsay at 075/463 494, after hours.

The following three pages contain articles by Mike Tod, which we have taken over from the Sept. 1984 issue of the I.C.P.U.G. magazine.

THE DREADED BUMPS

There can be no disk user who hasn't heard the dreaded grating noise produced by Commodore disk drives. The cause was mentioned in the previous DISK FILE (p238), and its occurrence is a serious hazard to the health of a drive, especially the 1541.

This head-banging routine occurs when certain read errors are encountered on the disk. However, many protection schemes designed to prevent (or rather make more difficult) illicit copying, create such errors and then check for their presence during loading. If they are absent, then the load will usually abort.

The result is a lot of head-banging, and drives slowly (but noisily) going out of alignment.

It is certainly possible to check for these corrupted sections of the disk without invoking the auto-mis-alignment feature but authors of protection schemes usually don't bother (or care?).

I don't want to get involved here with the arguments for and against protection, but I would say to all software authors that protection techniques that induce the bumps at any time are to be abhorred.

Indeed, I wonder if there may be a case for legal action against the software houses for selling a product that will cause damage to the disk drive. I guess if a garage sold you a tyre that caused your front axle to break, it would be legally liable.

Anyway, those who studied the 1541 memory map published last time may have spotted location $6A (REVCNT). This location has a flag which prevents the automatic invocation of the bumps.

By setting bit 7 to a 1, the bumps can be turned off, and this is performed as follows:

```
OPEN 15,8,15
PRINT#15, "M-W"; CHR$(106); CHR$(0); CHR$(1); CHR$(133)
CLOSE 15
```

The last CHR$ value is a combination of the number of attempts the drive should have if an error is encountered (in the 1541 this is normally 5) - bit 6 (a value of 64) turns the head offset routine off and bit 7 (=128) turns off the bumps.

I haven't altered the offset in the example as it causes no problems (the offset is the gentle tick-tick-tick-tick which you hear immediately before the drive goes 'crunch'). It is quite alright to alter the number of retries if desired - this can be of value if trying to read a particularly stubborn disk.

Users of other disk drives may like to know that there's a solution to the problem for the 4040 and 8050:

```
4040PRINT#15, "M-W"; CHR$(92); CHR$(67); CHR$(1); CHR$(138)

8050PRINT#15, "M-W"; CHR$(245); CHR$(16); CHR$(1); CHR$(138)
```

It appears that some 8050's may not respond to the modification because of internal differences in the memory maps.

Of course, it goes without saying that this modification is reset to normal whenever the drives are reset.

@ REPLACE - DOES IT OR DOESN'T IT?

I hate to bring up again the vexed question of the bug in the replace feature of SAVE and OPEN. However, when I was in Canada and the USA in June there was, as always at Commodore gatherings, a great deal of discussion on the problem, or rather apparent problem.

In their early disk days (1978-ish), Commodore announced that there was a bug related to using the @-replace feature. No firm evidence for it existed, but many problems appeared to be @-replace related. These generally took the form of files becoming interlinked.

Indeed, Commodore took the problem so seriously that the source code for the DOS of all Commodore drives has a comment in the area of code responsible for @-replace which reads 'Check for bug here'.

Many people, including Jim Butterfield, Harry Broomhall and myself, have gone through this code with a fine toothcomb and found nothing that can possibly cause a specific problem.

Some years ago, Jim and Harry both declared in public that the @-replace bug does NOT exist - not now or ever! While they were doing this, I remained more cautious but still accepted the fact that there was absolutely no evidence for its existence.

At the time, Harry threw down the gauntlet and offered a crate of beer to the first person to prove that a problem on their disk was caused by using @-replace. This obviously requires evidence, and the only valid evidence is either a disk on which an @-replace could be performed in the absolute knowledge that the act of @-replacing would cause corruption of the disk, or a precise statement of reproducible conditions that would cause @-replace to foul up. As yet, Harry has not had to pay up.

@-replace works by first of all writing the new version of the file to the disk and only when this has been succesfully completed is the original version scratched.

The fact that a scratch operation takes place means that the same precautions must be taken with @-replace as with the SCRATCH command. That is that no unclosed files should be @-replaced.

The disk also needs to have sufficient space to allow the complete file to be written, before the old one is scratched, and this could lead to a DISK FULL error - even if the new version is smaller than the old.

On the 1541, the DISK FULL error is reported earlier than on the 4040 (that is, with 3 blocks left) and the file is CLOSEd automatically by the drive when this occurs. If the operation was a SAVE, it could mean that the new version of the program on disk is actually shorter than it should be and this fact should be borne in mind. Note that a VERIFY will show all is well! The solution is to always make sure that the disk has space for the new file or program.

This problem is one which is not actually a bug with @-replace, but rather a bug with the DISK FULL situation.

Many people have been using @-replace all over the world for many years with no problems at all. Those who do experience problems appear to attribute other problems (some caused by their own errors) to the so-called bug in @-replace. There is no such bug!

RESETTING THE 64

The normal way of resetting the 64 back to its 'normal' configuration is to press the RESTORE key (with the RUN/STOP key pressed). This clears the screen and resets it to normal colours and mode, turns the sound and sprites off and resets some of the pointers but without upsetting any of the BASIC variables.

It is extremely useful when playing around with machine code or some of the more advanced features of the 64.

However, it is also possible to perform this reset using a BASIC command and Fred Mellings of Horley in Surrey has found that SYS1 also resets the 64.

The processor in the 64 has a special machine code instruction known as the BREAK instruction which is simply represented by a byte containing the value 0. When this is encountered, the processor starts to execute a special routine in the ROMs which is, unless deliberately altered by some other program, the same routine used by the RESTORE key.

The SYS1 command forces the processor to sequentially execute machine code starting from location 1 - but this area of memory actually contains data and not machine code, although the proceesor knows no difference. Eventually, a byte containing the value 0 is encountered, the processor enters the special reset routine, and the 64 is reset.

The contents of many of these locations (including 1) are not always predictable and so the effects of SYS1 could be unpredictable. Fortunately, location 2 normally contains 0 so a more reliable method would be SYS 2.

Random Numbers on the C-64

Want to calculate your Pools or Lotto entries, or write a simulation program (or game) that mimics statistical results? You need random numbers! Unfortunately, many users seem to get confused when trying to use random numbers on the VIC/C64. (Especially those who have grown up with the earlier PETs!) In this article, we will take a look at the RND function and how to use it on the VIC/C64.

The first principle to establish is that random numbers are not in fact truly "random", if indeed such can exist. The RND(X) function calculates a determinate floating-point number between 0.0 and 1.0 by a mathematical formula. Depending on the sign of the argument fed to the equation (X can be +ve, -ve or 0) a particular pseudo-random number will be generated. The "seed" for each calculation is stored in memory locations 139-143. These are set to a constant when the computer is switched on (on my C64 (all?) to 128,79,199,82,88 to give the first RND(1) of .185564016), and every subsequent call of RND resets (re-seeds) this value.

RND(+ve)

Any positive argument generates the next pseudo-random number from the current seed, then resets the seed for the next calculation. The actual value of the argument is irrelevant, RND(1) and RND(1256) do the same job. This process generates numbers in an apparently infinite sequence. However, West in "Programming the PET/CBM" suggests that after about 45000 repetitions the numbers lose their "randomness". This is the argument that should be used in most cases.

RND(0)

A zero argument takes its seed from the system hardware clock which changes every millionth of a second over a range of 0-16383. (From the CIA timer at locations 56324 and 56325.) It DOES NOT seed from the TI variable as some books say. Although RND(0) probably can be considered to generate the most "random" numbers, the problem is that the seed can only be one of 16384 possible values at best. And, since every 60th of a second the SAME seed value is used, as well as the fact that most programs perform a sequence of repetitive instructions, there is a fair chance of generating a very similar "random" number, for example 0.308594167 and 0.308594823. This causes difficulty when we wish to generate numbers over a small range, say 0-30.

RND(-ve)

A negative argument is quite different. It seeds the random number generator with a specific value depending on the value of the argument. This is always the same for the same argument. A RND(-ve) is used specifically to initialize the random seed to a set value so that we may generate a specific repeatable sequence of random number with subsequent calls to RND(+ve).

The best way to generate a "random" seed for a non-repeatable sequence is with the jiffy clock variable TI. That is, use X=RND(-TI) at the beginning of the program, then get the random numbers with RND(+ve).

The best way to see the difference between RND(0) and RND(+ve) is by a small program which POKEs the screen in a supposedly random fashion.

```
100 REM RANDOM SCREEN POKE
110 FOR I=0 TO 10000
120 N=RND(1)*1000
130 POKE 1024+N, 1 :REM SCREEN RAM
140 POKE 55296+N,1 :REM ** COLOUR RAM
```

** Line 140 is only needed on C64s made before approximately January 1984.

RUN the program and see what happens. The program will fill the screen with the 'A' character in a fairly random pattern until, at the end, if we let it run for long enough, the whole screen will be filled.

Now, change line 120 to use RND(0), clear the screen and re-RUN the program. Notice the difference? This time the 'A' character fills the screen in a definite pattern. No matter how long we let the program continue, the whole screen will never be filled! In this case, RND(0) will generate only 256 different numbers.

Using random numbers

Since the RND function generates a number between 0.0 and 1.0, it is not of much direct use for most cases. What is normally required is a random whole number (integer) between limits, say between 1 and 36 for the pools. This is done by

```
N=INT (RND(1)*(E-S+1)+S)
```

where S=1 and E=36 then N=INT(RND(1)*36+1). This will also generate the special case of numbers between -1 to +1 by

```
N=INT(RND(1)*3-1)
```

Let's finish with a full "6 from 36" pools program (or lotto, or whatever your addiction).

```
100 DIM N(6)
110 X=RND(-TI)
120 X=INT (RND(1)*36+1)
130 FOR I=1 TO N:IF X=N(I) THEN 120
140 NEXT :N=N+1:N(N)=X
150 IF N<6 THEN 120
160 PRINT "YOUR WINNERS (!) ARE"
170 FOR I=1 TO 6: PRINT N(I);: NEXT
```

..ps the C.C.U.G. (and I!) will expect a donation if you win!

Greg Perry

THE GROUP MODEM

For many months now our Technical Officer Roger Haigh has been slaving away over a hot soldering iron, busily designing a Modem, suitable for use with the VIC-20 and CBM-64.
It has now been tested, and yes (!), it is working. The group is now in the process of producing the first batch of 50 modems, and this is where our members who have expressed a desire to purchase one of these modems come into the picture.
The Pre-Release price of the modem (group members only) is $120.00. For this amount you will receive a fully assembled and tested unit, complete with a 90-day guarantee. Delivery is expected towards the end of February 1985, pending delivery of certain vital components.

As the group has to outlay a very substantial amount of capital to purchase components and parts we have to ask those members who wish to order at the pre-release price for a $100.00 deposit.

Those members who are not prepared to order at present will be able to purchase the modem after February at the regular group price of $150.00.

Roger Haigh will be pleased to answer your questions at our December meeting.

LIBRARY NOTES

A new disk (U6) has been added to the Public Domain Library:

C.C.U.G.D. U6

LOADME	Auto loader
PRINT DIRECTORY	Printout of Disk Directory.
TAPE-DISK SAVER	
DIR.SORT&PRINT"	Sorts and prints multiple disk directories
PROTECTOR	Protect files
FCOPY	Disk copier
8MONAD.A	Machine Language Monitor by Paul Blair.Instr.from Ken Charters
CMONAD.A	Same as above - different memory location.
FILE PARAMETERS	Displays all disk file parameters. From "Anatomy of 1541"
SUPERSPRITE	Game from Computes Gazette.
SOUND SCULPTOR 1	Sound program from Computes Gazette.
2	Part of above program
FAST ADD	Maths drill program.
MYSTERY MANOR	Game from Computes Gazette.
TREASURE HUNT	" " " "
S/SCRIPT STORY	A "spicy" Speedscript File.
PROPS	Computes Gazette game.
MIND BOGGLE	" " "
SKI PHYSICS	Relationships between speed, distance and time.
3-D TIC TAC TOE	Computes Gazette game.
CASTLE DUNGEON	" " "
FRANTICFISHERMAN	" " "
WORD SCRAMBLE	" " "
THERAPY	Computer Psychotherapist.
SPELLING CRITTER	Spelling quiz.
SHAPE MATCH	Pattern recognition for young children.
BONK BARRELS	Computes Gazette game.
SPACE PATROL	" " "
BEEKEEPER"	" " "
ROBOT MATH	Arithmetic tutor.
SNO CAT	Computes gazette game.
BALLOON BLITZ	" " "

Have you written an interesting program? Maybe you have modified a public domain program to make it work better. Possibly you have obtained a public domain program that is not in our library. We want to hear from you. Please submit programs on disk to Cliff Pottinger (You will get your disk back). Our Public Domain Library can only grow if you, the members, provide the material.

The Library Committee.

BOOK REVIEW

C 64 P D U B O I A H

This rather cryptic title stands for "Commodore 64 Public Domain Utilities Book Of Instructions And Hints".
The compiler of this book is our valued committee member Terry Steer, who spent many weeks gathering the material for this book and which is now available to group members for the sum of $5.00.
Like the other committee members Terry often receives phone calls for assistance from members. A lot of these calls relate to public domain programs which the group makes available to our members, but which are often insufficiently documented. To overcome this problem Terry decided to collate information on those programs and utilities which have proved most popular with our members.
With a collection such as this there will always be omissions of course, but, given the fact that this has been largely a labour of love, I think that our compiler has done a very good job indeed.

Some of the topics covered include:

Speedscript	Character Editor	C-64 Wedge
Backup Programs	Copy Files	Sprite Editor
Tape Copier	Joysticks	Pokes

and many more.

All in all, I feel that Commodore users who struggle with the above topics can save themselves a good deal of frustration by investing in this booklet. Recommended.

Ralph De Vries

From Other User Groups

VIC-UPS W.A.: This group, which covers some 6 user groups in and around Perth has around 1500 members! Their newsletter is properly printed (type-set that is), and attracts a lot of advertising support from local dealers. They certainly seem to have their act together in the west.

Commodore User Group A.C.T.: Their Oct. Newsletter runs an interesting program to modify Speed Script to work with the P/D centronics parallel printer wedge. If you are interested in this article see our librarian for a photocopy of same.

Commodore 64 User's Group VIC: In the September issue of their newsletter the editor is begging for articles. Let's face it, if it was not for Greg Perry and some of our regular software reviewers we would be in the same boat.

Rockhampton Commodore Users Group: As mentioned elsewhere this group is now up and running, and already they produce a newsletter! We also had the pleasure of meeting some of their members recently. We look forward to further contacts.

S.A. Commodore Computer Users Group: A very active group. We have been receiving their newsletter for quite some time now, but regrettably have not had any personal contact. We must remedy that in the new year.

Strange as it may seem, we have never received a newsletter from NSW user groups, although we send copies of "CURSOR" to several groups in that state. Could it be political?

Is there a Commodore Users Group in Ipswich? We don't know of one, but rumours persist that there is one. Who can enlighten us?

During the Computer Expo we had a chat with a Toowoomba computer dealer who would like to form a Commodore group in that city. We wish him well in his endeavours, and we have offered him any assistance we may be able to render him.

Most of our members are probably aware of the fact that local newsagents now stock two Commodore magazines which are published by Commodore in the USA. They are called "Commodore Microcomputers" and "Power/Play". Although published by Commodore they contain sufficient independent articles to make them worthwhile.
In the Sep/Oct issue of "Commodore Microcomputers" there is an interesting article by Jim Strasma on Database programs, as well as other articles on Spreadsheets and Wordprocessors.
As the undersigned has recently spent quite a few hours reviewing several DB programs it was good to see my own conclusions on "The Consultant" and "The Manager" confirmed by Jim Strasma.
I can thoroughly recommend this issue to those members who are interested in business applications on the C-64.

Ralph De Vries

COMMODORE COMPUTER USERS GROUP (QLD) DIRECTORY

President:	GREG PERRY	Ph. 38 3295
Secretary:	NORM CHAMBERS	Ph.341 5651
Minute Secretary:	ROGER HAIGH	Ph.399 8037
Treasurer:	LESTER BENNETT	Ph.200 1243
Education Officer:	DEREK FARRELL	Ph.359 8559
Technical Officer:	ROGER HAIGH	Ph.399 8037
Chief Librarian:	CLIFF POTTINGER	Ph.277 4520
Librarian - Books:	CLIFF YULE	Ph.356 7571
Librarians - Disks:	Ken Charters	Ph.341 7222
	Max Bean	Ph.208 1225
Librarian - VIC:	JULIANNE FALLEN	Ph. 30 2982
Newsletter Editor:	RALPH DE VRIES	Ph. 30 3477

VIC-20 Sub-Committee Chairman:	BARRY WILSON	Ph.399 6204
CBM-64 Sub-Committee Chairman	TERRY STEER	Ph.200 5926
PET Sub-Committee Chairman	GREG PERRY	Ph. 38 3295

For specific computer problems contact members of the relevant Sub-Committee.
Please enclose a stamped self-addressed envelope, when contacting committee members by mail.

Please address all editorial matter to:
24 Kaloma Rd. The Gap, Brisbane, QLD 4061. (Not to P.O. Box 274, Springwood please!)
Deadline for any particular month is the second Tuesday of that month.

The opinions expressed herein are those of the Author(s), and not necessarily those of the C.C.U.G.(Q) or the Editor.

Published by C.C.U.G.(Q), P.O.BOX 274, SPRINGWOOD Q 4127
Printed by GAP PRINTING, Lahore Street, THE GAP Q 4061
