COMMODORE COMPUTER CLUB **NEWS**

VOLUME 1, NUMBER 7 — APRIL, 1984 — VANCOUVER, B.C. • FREE • 24-HOUR CLUB ANSWER PHONE: PET-3311 (738-3311)

ALL THE NEWS THAT FITS

We're finally branching out, with our Richmond sub-group holding its first meeting Wednesday, April 4 at McNair Senior Secondary School in Richmond (No. 4 Road and Williams). The exact location is the school cafeteria, which hopefully will be reserved on an ongoing basis for the club, though other activities which have priority might necessitate the use of alternate space in the school.

Present and new members are welcome at this meeting along with their equipment. Whether the 64 and VIC libraries will be available at this Richmond location has yet to be determined, but the kind of response shown at the first couple of meetings will give an indication of the direction this group will take.

For information on upcoming Richmond meetings, check with our 24-hour answer phone -- 738-3311 (PET-3311).

***

Speaking of our answer phone, don't wait to call for information about the next meeting until about an hour before the meeting, since chances of your getting through are about as much as winning the top prize in the Lotto 649.

If all 600 of our members called it in succession, it would take nearly 10 hours for everyone to listen to the one-minute message (has anyone ever thought about a shorter message tape?). A good time to call it is late at night when you get up for a 1 a.m. snack or early in the morning.

***

In keeping with the club's anti-piracy resolution which was approved by members at the annual general meeting in January, a close eye is being kept on this activity at all meetings of the club. Our most recent workshop had a confrontation with one member who was allegedly pirating copyrighted software and he was told in no uncertain terms to clean up his act. It should be emphasized that the club has no interest in what goes on in the "basements of the nation", but ripping off software at club meetings, especially in the face of those who make or distribute it, can only bring the club a great deal of unnecessary grief (not to mention the would-be pirates). So brain up!

***

If you belong to the Commodore Computer Club, you are expected to take part in its activities, in other words, get your $20 membership worth. One of these club benefits is the newsletter, which is distributed free at all meetings, and recently in many of the local computer and software stores as well. We expect people to pick up the paper at the club, since to mail it out would cost almost $300 a month, which is more than the total cost of producing it. If you want to receive the paper at your home, this can be arranged, but at an additional expense. See our friendly editor (Mike Q.) for details.

On the other hand, if you live more than 150 km outside of Vancouver, you can rip us off! Membership in the club for someone so far away who won't likely attend our meetings costs only $10.00. Since it costs about 75 cents total to send a copy of the newsletter, and if we send 12 issues out during a year's membership (unlikely as that may seem), your membership ends up costing only $1.00. Since membership rates can't be adjusted until the next annual general meeting to be held in January, 1985, take advantage of this unique offer!

***

Another benefit of club membership is that bulk purchases of various items can be arranged for members, often at substantial savings below the normal retail cost. One such deal currently being investigated is a group purchase of the NEC 8023 printer through one of the larger local retailers. If enough club members join in, the total cost of the printer, including the Cardco G+ Interface would be around $600. Details of this are being announced at club meetings.

***

The Timex Sinclair Users Group and the Port Moody Parks and Recreation Department is sponsoring a mini-computer fair to be held from 1-5 p.m. on Sat., Apr. 28 and Sun., Apr. 29 in the Kyle Recreation Centre at 125 Kyle Street, Port Moody. For information on participating, call Marvin Steinway at 464-5818.

INSIDE THE 1541: 1

By LARRY PHILLIPS

In this series of articles, we will take a look at how a disk drive performs its function. We will start with the fundamentals of disk drives in general, explode a few myths, and finally, we'll examine some of the intricacies of Commodore drives.

First, let's define a few terms. If you own a Commodore machine, you think of the disk drive as 'everything inside that short wide machine plugged into your serial port'. Not quite so. The Commodore machine actually consists of a 'drive' and a 'controller'. The drive itself is capable of performing only three functions. It can seek, read and write. More about these later. The controller is the 'brains' of the operation, and is capable of anything the designers see fit to include. In Commodore's case, the controller is relatively sophisticated, and is generally known as a 'smart' controller. It handles the task of translating BASIC file commands (OPEN, PRINT#, LOAD, etc.) into seek, read and write commands for the drive.

Now back to the drive. Its purpose in life is to spin a diskette underneath a movable head. This head is very much like a head on an audio tape deck. When a current is passed through the head, it will magnetize the coating on the diskette. The direction in which the coating (an oxide of iron) is magnetized is determined by the direction of the current through the head. As the diskette spins, the head will lay down a circular path of magnetic transitions in the oxide. This is known as a 'track'. In order to be meaningful, the track will consist of many reversals of the direction of magnetization. These flux changes may then be read by the drive by passing the diskette under the head without passing current through the head. The flux changes will the induce currents in the head that are then amplified and fed to the controller.

Again, these two processes are similar to what happens in your audio tape deck. The big difference is that a tape must be read all the way from its current position to where the required song (information) is. A disk drive, on the other hand, also

(Continued on page 4)

CREATING MEDIUM RESOLUTION GRAPHICS ON THE COMMODORE-64

By DAVE WHITE

The reader will probably agree that one of the strongest features of the C-64 is graphics. Yet the C-64 does not have medium resolution graphics. The 80 by 50 graphics mode available on machines such as Atari, Apple II, and ZX81 just isn't there. Here is a program which gives us medium resolution graphics; it is written entirely in BASIC. Boolean arithmetic is used to access a two dimensional array. It is hoped that some of the programming concepts will be useful to the reader. The program does not run very quickly; you may wish to modify it to increase its speed.

DESIGNING THE PROGRAM:

80 by 50 is just double the normal or low resolution mode, in which one character represents one POINT, or CELL. If we wished to plot a pattern in LOW RES we would probably print <RVS-SPACE> for each cell to be plotted. There are some graphic characters which "break up" the <RVS-SPC> into four parts, or quarters. One of these characters is <C= B> (Commodore-B). If we were to fill a screen with <C= B>'s, an 80 by 50 checker-board would be the result. We have doubled the resolution by printing a character which has 4 "squares", or "quadrants" for each character.

There are eight graphic characters which use these "quadrants":

<C= B>, <C= C>, <C= V>, <C= D>, <C= F>, <C= U>, <C= I>, <space>.

With the use of reverse video we have a total of 16 such characters. All we have to do is figure out which ones to use, and when to use them! We can restate this problem to make it (hopefully) a little clearer. Suppose we have a value of X (from 0 to 79) and a value of Y (from 0 to 49). To plot this point on our screen we must:

1) Figure out which quadrant (i.e., which of the four corners) of the print character do we wish to be black? We will choose the graphic character which shows that quadrant black, leaving the other quadrants white.

2) Figure out where on our screen we should place this graphic character?

3) Find this character in an array (which we constructed earlier) and print it at the desired location.

Already we see a flicker of light at the end of the tunnel! Let's look at these steps, one at a time:

STEP 1:

We will do this in two parts. First we will find whether the quadrant lies in the TOP or BOTTOM of the character, then we will find whether it lies on the LEFT or RIGHT side of the character. Let us suppose we were plotting the medium res point (X=45,Y=0). This should fall at the extreme top of our screen, just to the right of center. Obviously it lies at the TOP of the character. (X=45,Y=1) is one half-character lower; it lies, therefore, in the BOTTOM of the character. (X=45,Y=3) would be TOP, and so on, alternating back and forth between TOP and BOTTOM. In fact, when Y is an EVEN number we use the TOP; when Y is ODD we use the bottom. Similarly, When X is EVEN we use the LEFT side of the character; when X is ODD, we use the RIGHT side. Thus all we need to do is find out whether the coordinates are EVEN or ODD. We do this by dividing by 2 and checking to see if there is a remainder.

STEP 2:

This is easier. The ROW is simply the integer of Y/2. The COLUMN is the integer of X/2.

We can use a calculation which combines steps 1 and 2:

```
COL=INT(X/2)
COL'REMAINDER= X - 2 * COL
ROW=INT(Y/2)
ROW'REMAINDER= Y - 2 * ROW
```

The remainders will have values of either zero or one. See lines 35-40 and 720-725. We use these remainders as subscripts of Q, the array which gives the subscript (1,2,4, or 8) to be used in referencing the array of graphic POKE codes (lines 42-43, 740-750). Let us see just how this is done.

Look at line 114. In the array element Q(1,0), the first subscript '1' is for "LEFT or RIGHT side". In this case, the '1' tells us it is the RIGHT side. The second subscript '0' is for "TOP or BOTTOM". The '0' informs us that it is TOP.

```
125 Q(1,0)=8
```

In line 125 we see that in array B, the subscript 8 references the POKE code 124. Your Commodore Users Guide shows that 124 is indeed the POKE code for the graphics character which shows the top right quadrant black, all else white. The reader may wish to check some of the other codes as well.

Now what we have said is all very fine. It would even appear that it just might work. But, so far, we have not allowed for those cases where there is ALREADY someting plotted in the character space we are about to poke. We will have to "add in" this information somehow. It turns out that this is not all that complicated. We just peek at the memory location to find the ASCII code of the character that is there. It will be one of our

(Continued on next page)

SPEEDSCRIPT TIPS

Since the article we published a couple of issues ago on Speedscript, the word processor in the January issue of *Compute's Gazette*, we've had several inquiries about how to make it work. Seems that copies of this issue are scarce as 364's, so we're reprinting the special commands below. We would recommend, though, that you endeavour to purchase the magazine from its Back Issues Dept., since there are many items of interest in the article accompanying Speedscript.

USE WITH "CONTROL"

A -- Change case
B -- Change background color
D -- Delete (Sentence, Word or Paragraph)
E -- Erase (Sentence, Word or Paragraph)
H -- Hunt for ...
I -- Insert Mode
K -- Clear Buffer
L -- Change letter color
P -- Print
R -- Recall buffer
V -- Verify
X -- Transpose Characters
Z -- End of document
4 -- Disk Directory
Up Arrow -- Send DOS command
English Pound -- Enter format key
Back Arrow -- Delete Character
= -- Amount of memory remaining

OTHER THINGS:

F1 -- Next word
F2 -- Previous word
F3 -- Next sentence
F4 -- Previous sentence
F5 -- Next paragraph
F6 -- Previous paragraph
F7 -- Load
F8 -- Save
Cursor Up -- Previous sentence
Cursor Down -- Next Sentence
Cursor Left/Right -- Does what it says
CLR -- Erase Everything
HOME -- Top of Screen
HOME (pushed twice) -- top of copy
Back Arrow -- Back Space
RUN/STOP -- Insert 5 spaces

FORMAT COMMANDS (use with CONTROL-English Pound)

l -- left margin (usually 5)
r -- right margin (usually 75)
t -- top margin (usually 5)
b -- bottom margin (usually 58)
h -- define header
f -- define footer
w -- wait for next sheet of paper
a -- true ASCII
u -- underline switch
c -- center line
e -- edge right
-- page number
s -- spacing (usually 2)
1-9 -- are redefinable characters

16 graphics. We could "walk through" our array A of ASCII's until we found it, but that might be time consuming. Instead we use another array which does just the reverse of array A.

Array A uses BIT-CODE as a subscript to reference ASCII code.

Array B uses ASCII code as a subscript to reference BIT-CODE.

The BIT-CODE is the number (from 0 to 15) of the graphic character. To "add" this BIT-CODE to the one we wish to use we use the Boolean OR (see line 740). We wisely chose values of BIT-CODES for the characters defining single points (ie, the four corners of the character) which were even powers of 2.

2 0=1
2 1=2
2 2=4
2 3=8

Exactly how this Boolean arithmetic works may be the subject of another article.

Here are some suggestions:

Type in and run the program.

Lines 200-299 may be replaced with any plotting routine of your choice.

Try modifying the program so that the two arrays A and B are loaded using READ and DATA. This requires less coding and less chance of error, since there need be no duplication of ASCIIs and bit-codes.

Use color in the program. You will notice some rather strange things happen when two different colored points find themselves in the same character cell.

```
4 REM  *** 80 BY 50 GRAPHICS ***
5 REM  *** BY DAVE WHITE   224 3
082 ***
15 :
20 X=0:Y=0:C%=0:R%=0:CR=0:RR=0:M
=0:C=0
30 REM X,Y ARE COORDS IN 80 * 50
 GRID
35 REM C%,R% ARE COORDS IN 40*25
 GRID
40 REM CR,RR ARE REMAINDERS,  US
ED TO FIND WHICH QUARTER OF THE
BYTE IS ON.
42 REM M IS THE ADDRESS OF SCREE
N MEMORY   BEING POKED
43 REM C IS THE BINARY CODE FOR
QUARTER    OF BYTE (RANGES FROM
0-15)
44 :
45 DIM Q(1,1)    :REM    BIT COD
ES OF QUARTER-BYTES, WITH SUBSCR
IPT= REMAINDER
46 :
47 DIM A(15)     :REM TABLE OF AS
CII POKE CODES, WITH SUBSCRIPT =
 'BIT' CODE
48 :
50 DIM B(255):   :REM     TABLE
OF 'BIT' CODES, WITH SUBSCRIPT=
ASCII POKE CODE
51 :
56 :
110 REM      LOAD ALL ARRAYS (TA
BLES):
111 :
113 Q(0,0)=4 :REM TOP LEFT QUART
 OF BYT
114 Q(1,0)=8 :REM TOP RT   QUART
 OF BYT
115 Q(0,1)=1 :REM BOT LEFT QUART
 OF BYT
116 Q(1,1)=2 :REM BOT RT   QUART
 OF BYT
118 :
120 A(0)=096: A(1)=123: A(2)=108
122 A(3)=098: A(4)=126: A(5)=097
124 A(6)=127: A(7)=252: A(8)=124
125 A(6)=127: A(7)=252: A(8)=124
126 A(9)=255: A(10)=225:A(11)=25
4
128 A(12)=226:A(13)=236:A(14)=25
1
130 A(15)=224
135 :
139 B(32)=0
140 B(96)=0:  B(123)=1: B(108)=2
142 B(98)=3:  B(126)=4: B(97)=5
144 B(127)=6: B(252)=7: B(124)=8
146 B(255)=9: B(225)=10:B(254)=1
1
148 B(226)=12:B(236)=13:B(251)=1
4
150 B(224)=15
169 :
170 REM FILL UP ALL OF COLOR MEM
ORY NOW
175 PRINT CHR$(147)
180 FOR M=55296 TO 56295:POKE M,
12:NEXT
185 POKE 53281,0:POKE 646,15
200 :
205 REM MAINN LOOP TO DEFINE PLO
TTINNG:
210 :
215 FOR D= 2 TO 8 STEP 2
220 FOR X=  5 TO 75
230 Y=INT(X/2)+D:GOSUB720
240 NEXTX
250 NEXTD
299 END
699 :
700 REM PLOT SUBROUTINE
710 :
720 C%=X/2:CR=X-2*C%
725 R%=Y/2:RR=Y-2*R%
735 M=1024+R%*40+C%
740 C=B(PEEK(M))OR Q(CR,RR)
750 POKE M,A(C)
795 RETURN
799 :
```

A COMPLEAT GUIDE TO MACHINE LANGUAGE PROGRAMMING ON THE PET

By HAROLD BROCHMANN

CHAPTER 5 -- ASSEMBLY LANGUAGE

INCREMENTING AND THE CONDITIONAL BRANCH [5-3]

In BASIC we frequently make use of FOR...NEXT loops, for example:

```
10 FOR X=1 TO 100
20 PRINT X
30 NEXT
```

Here is another program which is also a classfied as a loop:

```
10 X=X+1
20 PRINT X
30 IF X<> 100 THEN 10
```

The second program accomplishes its loop by INCREMENTING (line 10) and by use of a CONDITIONAL BRANCH (line 30).

In ML we can increment the X register with INX, the Y register with INY and the accumulator with INA. Similarly we can branch with BEQ (branch if the last result was equal to zero), BNE (branch if result not equal to zero), BPL (branch if result is positive) and BMI (branch if result is negative).

BPL and BMI are rather difficult to use and we shall not deal with them now. The other two, BEQ and BNE are very convenient to have available and will be introduced at this time.

In 6502 machine language, conditional branching is never done to an absolute address. Rather, such branching is RELATIVE. This means that we branch so many addresses forward or so many addresses back from where we are at the moment.

Let us write a loop in ML:

```
033A A2 00    LDX #$00
033C A9 05    LDA #$05
033E 9D 00 80 STA $8000,X
0341 E8       INX
0342 D0 FA    BNE $033E
0344 60       RTS
```

The exact BASIC equivalent of this ML program would be:

```
10 X=0
20 A=5
30 POKE 32768+X,A
40 X=X+1
50 IF X<>0 THEN 30
60 RETURN
```

Of course, this BASIC program would never stop because X would never get to zero! In ML, however, the largest value that we can refer to is $FF (255). If a byte containing this number is incremented, it "rolls over" to zero. For this reason the BASIC program line 50 really ought to read: 50 IF X<>256 THEN 30

As far as our ML program is concerned, then, BNE allows execution to procede to RTS after $FF (255) increments to zero.

Let us examine the contents of $0342 and $0343. $D0 is the BNE instruction, but how does $FA refer to $033E?

Conditional branches have relative addressing. They can branch forward 127 addresses and backwards 128 addresses. Forward addressing is accomplished with relative addressing $01 - $80, while backward branching is accomplished with relative addressing $FF - $81. $FF - $FA = $05, so that the address to branch to is 5 back from $0343. This places us at $033E.

Calculating these relative branch addresses are a real nightmare and if we don't find a handier way to do it all sorts of mistakes are inevitable. Fortunately this problem as well as many others will be made very much simpler to deal with when we introduce MICROMON in the next chapter.

For now, enter the last program as follows using MLM.

```
.: 033A A2 00 A9 05 9D 00 80 E8
.: 0342 D0 FA 60 .. .. .. .. ..
```

SYS 826 should fill the first 256 screen locations with the letter E. As you can see, this happens incredibly quickly, and should give you some idea of how fast machine language routines execute.

SUMMARY [5-4]

To summarize this section we provide a list of the assembler instructions we have discussed so far together with their various addressing modes and corresponding codes:

MNEMONIC	IMPLIED	IMMED.	ABS.	ABS.,X	ABS.,Y	REL.
LDA	-	A9	AD	BD	B9	
STA	-	-	8D	9D	99	
LDY	-	A0	AC	-	-	
STY	-	-	8C	-	-	
LDX	-	A2	AE	-	-	
STX	-	-	8E	-	-	
INY	C8	-	-	-	-	
INX	E8	-	-	-	-	
RTS	60	-	-	-	-	
BRK	00	-	-	-	-	
BEQ	-	-	-	-	-	F0
BNE	-	-	-	-	-	D0
BPL	-	-	-	-	-	10
BMI	-	-	-	-	-	30

For a complete list of the 6502 instruction set, we refer you to a more advanced text.

INSIDE THE 1541

(From page one)

has the ability to seek, or to move its head to another track, without having to read over all that unwanted stuff in between. The little ticking noises you hear in you drive when you load or save are the sound of a stepper motor moving the head to the track you are interested in.

Disk drives come in all sorts of configurations and capacities, from the old standard 80k bytes or so, to hard disks that hold 1.27 gigabytes (1,270 megabytes, or 1,270,000 k bytes, or about the amount of data contained on 7,500 Commodore floppies). Regardless of capacity, The principles outlined above hold true.

Before we leave the head, I should explode a myth. In a floppy disk drive, the head does not 'fly', but actually rubs the surface of the diskette. Only on hard disks does the head fly above the surface at a very low altitude. This does not mean that you should worry any less about cleanliness, though. You can still lose data, although it will not be in quite as spectacular a fashion.

We'll finish off for now with a look at diskettes. They come single or double sided, single or double density, and hard or soft sectored.

Density refers to the amount of data that may be reliably retained on a floppy.

Sectoring refers to the way a drive determines its rotational position on a track. Some drives have a sensor that detects 'sector holes' in the floppy. Others detect an Index mark only, and then write special sector marks on the diskette which are later read and sent to the controller for decoding into position information. In Commodore's case, the process is carried one step further. There are no positional sensors in the drive at all. Every sector is identified with a 'sector header' that is written on the track and contains information telling the controller about track, sector, and disk ID.

A double sided drive is one that has two heads, one on each side of the diskette.

When buying diskettes, your best bet is to buy double density, as you get statistically better data retention. The sectoring, hard or soft, does not matter in the least, and is ignored by your drive. As for single or double sided, I leave the choice to you. The diskette manufacturers will tell you that you should not use both sides of a floppy in a single sided drive. The reason, they say is that in doing so, you are causing the disk to spin in the opposite direction, and therefore dragging dirt back out of the sleeve. Personally, I get excellent results using both sides of single sided, double density diskettes.

NEXT ISSUE: Where are the 'brains' anyway? and What happens when you NEW a diskette?

Published by The Commodore Computer Club, P.O. Box 91164, West Vancouver, B.C. V7V 3N6. Editor: Mike Quigley.

Club meetings are normally held: *Workshop*: first Tuesday of the month, 7:00 p.m., Thompson Secondary School cafeteria, 1755 E. 55th Ave. (near Victoria Drive); *Business*: third Tuesday of the month -- 7:00 p.m., King Edward Campus, 1155 East Broadway, 2nd floor auditorium. These dates and locations are subject to change. For up-to-date information on any changes, please call the club's 24-hour answer phone:

PET-3311 (738-3311)

Club Executive: President -- Jim Bauerle; Vice-President -- Sigmund Steiner; Secretary -- Marvin Steinway; Treasurer -- Hu Reijne; Directors -- Robert de Boer, Guenter Hake, Jim Jorgenson, Terry Juuti, Murray Kopit, Mike Quigley, Elmer Roy, Philip Seligman, Nick Shevchenko, Gerri Sinclair, Tony Smith, Arthur Taser, Al Townsend, Jim Wilcox.

THE COMAL CORNER

By LARRY PHILLIPS

For years, I have been complaining about overpriced software. For a company to charge fifty or sixty dollars for a rather poorly written game is a complete outrage, and tends to victimize those who are just starting out in the hobby. Beginners often find programming a difficult task, and buy commercially available software only.

Commodore Canada has recently helped this situation in two ways. They have made a software package available that is priced far less than it's worth. With the same release, they have given beginners an easier route to travel in learning to program. I am referring, of course, to COMAL for the 64. COMAL stands for COMmon Algorithmic Language. In comparison to BASIC, it is more powerful, allows better structuring of a program, is easier to read, and easier to learn. The price? Ahh, that's easiest of all, it's in the public domain. Yes! Free! I could hardly believe it. Your only expense will be the COMAL HANDBOOK, by Len Lindsay. ($18.95 U.S. - about $25.00 here, when you can find it).

This text contains a complete reference guide to the language, sample procedures and functions, and some comparisons to BASIC (for those of us who have many bad programming habits to break). If you aren't sure if you want to use it in place of BASIC, you really don't have to buy the HANDBOOK, because Commodore has seen fit to include quite a few programs and some limited documentation on the COMAL disk. Though some of the programs have minor bugs and flaws, none are too serious, and they will serve to show you how the language is used.

Now, what is this language I rave about?

COMAL has very little in common with BASIC. There are quite a few commands and statements that mean the same thing in both languages, and some BASIC statements that COMAL will translate to its own syntax. This is where the similarity ends (Thank the Great Programmer).

First and most important, COMAL ignores line numbers while running, but has the good sense to use them when you are editing. This means that you no longer have to remember what line you put that routine or subroutine on. Instead, you will use a label, function name, or procedure name to do various things in your program. Imagine! No more GOSUB 1450, hoping it's right, in order to sort an array. COMAL allows you to say something like SORT (last'name$).

In using line numbers during the editing process, COMAL avoids the problems associated with editing in such languages as PASCAL, FORTRAN and LISP. The problems in these languages stem from the difficulty in finding the place in the program that you want to change. COMAL acts somewhat like BASIC, with full-screen editing. LIST to, LIST from, LIST range are implemented. When you enter a line number followed by a statement, the line will be replaced in the program. There are two differences, however. The line is checked for proper syntax when it is entered, and COMAL will not enter the line if it is bad. The other difference is in the deleting of a line. If you type the line number by itself, and hit RETURN, the line will not be deleted. You must type 'DEL 500' to delete line 500. You can specify ranges just as in LIST. This feature may seem a bit awkward at first, but sure stops you from accidentaly erasing a line.

COMAL relies heavily on what are referred to as structures. A structure is a pair of statements that form a logical unit. Everything in between the two statements will be executed in a manner prescribed by the structure. Sound complicated? Well it isn't, and a few examples will serve to illustrate the principle.

The structured statements in COMAL are:

```
Repeat ... Until
While  ... Endwhile
Case   ... When ... Endcase
If     ... Then ... Elseif ... Else
   ... Endif
For    ... Endfor (like Next).
Func   ... Endfunc
Proc   ... Endproc
```

When you LIST a COMAL program, all structures are shown by indentation, i.e.:

```
repeat
 input "enter y or n ":yesno$
until yesno$ in "Y y N n"
```

Notice the indentation of the second line that serves to show the structure of the routine. This little quickie will accept only 'Y' or 'N' (shifted or unshifted) as an input. Notice also the variable name 'yesno$'. Variable names may be up to 16 characters long, and may be real, integer, or string, and all types may be used in arrays of up to 32 dimensions. Lengths of string variables are limited only by the amount of available memory.

All structures are multi-line, that is, you may include more than one line of code within a structure. You may also nest structures. Procedures and functions are recursive. This means that a function or procedure can call itself from within itself. Handy if you can figure out what will happen when you do.

All in all, this is an extremely versatile and powerful language that fills a gap (to be precise, an 8k ROM gap called BASIC). The price is right, the language is excellent. Ask your club librarian or a friend who has it about getting a copy.

ERRATA DEPARTMENT

The following are corrections for the Compute! book Machine Language for Beginners:

The following are corrections for the Compute! book Machine Language for Beginners:

***

Program 1-3, VIC Version: page ix of the Introduction.

Line 800 should read as follows:

```
800 FOR AD=864 TO 885: READ A:
    POKE AD,DA: NEXT AD
```

Lines 810-840 need to be changed to the following:

```
810 DATA 160,0,169,1,153,0
820 DATA 30,153,0,31,169,6
830 DATA 153,0,150,153,0,151
840 DATA 200,208,237,96
```

***

Program 1-4, 64 Version: page x of the Introduction.

On the newer, more recent models of Commodore 64s the color registers must be set or only half the screen will be filled.

***

Program C-2, Simple Assembler (VIC, PET, Apple, 64 Version) on page 227.

In order for this program to work properly on a VIC-20 or a Commodore 64 the following change needs to be made: In line 200, on page 227, the semicolon needs to be removed.

***

Supermon 64 (page 319).

Before entering the hex numbers with the use of the Tiny PEEKer/POKEr, type in the memory partitioning POKES: POKE 8192,0 and POKE 44,32 then type NEW. Then, after you've finished entering all the hex numbers type: POKE 44,8: POKE 45,232: POKE 46,17: CLR. You can then SAVE it in the ordinary, BASIC way, to tape or disk. It's ready now to RUN or LOAD. (Note also that the checksum program on page 333 checks 129 bytes at a time. This can have the effect of attributing a typing error to the wrong block if the error occurs near the beginning or the end of a block.)

***

Program 5-1, page 60, will fail under one condition. Here is a foolproof double compare:

```
SEC
LDA TESTEd
SBC SECOND
STA TEMP (store result in any
temporary location)
LDA TESTED+1
SBC SECOND+1
ORA TEMP
BEQ EQUAL
BCC LOWER
BCS HIGHER
```

SO WHAT'S A "WEDGE"?

BY H. JIM BAUERLE

The WEDGE program is usually supplied with the demo diskette of any Commodore disk drive you purchase.

Once you seriously start using the Wedge, you will get very dependent on it. I put it as the first program on every disk under the name "++UNIV DOS". This permits me to load it by simply typing LOAD"+*",8. If you follow this command with a colon and push SHIFT RUN/STOP, the program will run automatically. Once the program is RUN, it tucks itself away in high memory, protected from BASIC. Loading some machine language programs may destroy it, so watch for this. [Ed. Note: Using programmable characters on the VIC interferes with the Wedge.]

Why is this program called a "Wedge", you ask? Well, on the PET, the ">" character is called the Wedge by a lot of people. Most commands in the program are proceeded by this character. You can also use the "at" sign (@), located between the "P" and the asterisk.

Here is a summary of some of the major commands. We assume that you push RETURN after each of them and that all Wedge commands start in the leftmost column of your screen.

> [RETURN] -- by itself will return the disk status (error messages, etc.). This is the command you use after a disk error has occurred and the red error LED on your drive flashes.

@ -- may be used interchangeably with the > character in all following examples.

/BLAT -- will LOAD the program "BLAT" into memory without running it. Note that quotation marks are not necessary. To run it, you must type RUN.

/BL* -- A shortened way to do the above. This will load a program called "BLASTED" if it precedes "BLAT" in the directory.

>$ -- this will show a list of programs on the disk. The directory will scroll from the bottom of the screen after it is filled. This means the topmost programs will disappear. You may stop this scrolling action any time by pushing the SPACE bar. To start the scrolling again (to see more programs in the directory), simply press the SPACE bar again. To get out of the Directory, push RUN/STOP.

>$0 and >$1 -- directories for Drive 0 and Drive 1 respectively (for those with dual drives).

>$0:*=SEQ -- will list only sequential files

>$0:*=S -- same as the above (for lazy typists like myself)

>$0:*=PRG -- will list only program files

>$0:*=P -- same as above

>$0:BLAT -- will find the file called "BLAT" and only that file (no matter what kind of a file it is)

>$0:B*=PRG -- will list all files beginning with the letter "B" which are program files

>$0:B* -- will list all files beginning with the letter "B" regardless of type

>$0:???B* -- will list files with a "B" as the fourth letter in their titles (example -- FUMBLE)

R0:NEW BLAT=0:OLD BLAT -- RENAMES a file called "OLD BLAT" and calls it "NEW BLAT"

S0:BLAT -- will scratch (remove from the disk) the program named "BLAT".

S0:B* -- will scratch all files which begin with the letter "B"

S0:* -- This command will scratch every file off the disk, leaving you with no programs. *USE THIS COMMAND WITH EXTREME CAUTION!*

V0 -- Will Validate or Verify the disk. This reads through all the programs on the disk and updates the BAM or Block Availability Map so the Disk Operating System (DOS) knows which blocks are not in use and can allocate new files to them.

I0 -- Initializes the disk. This brings the read/write head to Track 18, which is in the middle of the disk. Doing this often makes it easier to read files on disks which have misaligned tracks. Many old-time programmers religiously Initialize before Writing or Reading anything to the disk.

N0:NEW DISK,ID -- This command is a very powerful one -- *it will erase all the information from your disk, so use it with great caution!*. You must use this command with new disks which you have just bought in order to format them -- in other words, prepare them so they can be read and written to with your particular drive. "NEW DISK" in this case is the name which we have given the disk, and ID is a 2-character code which denotes a unique identifier for that disk. Any two letters or numbers here are acceptable, i.e. 14, AA, Q1. Take care not to use the same ID on two different disks. Under certain circumstances you can take out one such disk and put another in the drive, and really mess up everything!

C0:NEW FILE=OLD FILE -- This seldom-used command copies one program "OLD FILE" onto the disk under another file name ("NEW FILE").

AD RATES FOR THE COMMODORE COMPUTER CLUB NEWS:

(All dimensions in inches)

FULL PAGE	10-1/4x15	$150.00
1/2 PAGE	10-1/4x7-1/2	$ 90.00
1/3 PAGE	6-3/4x7-1/2	$ 67.50
1/6 PAGE	3-1/4x7-1/2	$ 37.50
1/12 PAGE	3-1/4x3-3/4	$ 22.50
BUS. CARD	3-1/4x2-1/2	$ 20.00

Copy must be camera ready and to the exact size specifications above. All printing and artwork must be black ink on white background. All pictures must be screened.

Any deviation from the above will result in additional charges at the prevailing rates. As an example, a half page ad was made up for one issue and the cost of producing the ad was equal to the cost of the space alone; therefore, the total cost of the ad was double the above rates.

Color and other services are also available, at additional charge.

All ads are payable in advance except where a purchase order accompanies the insertion.

For further information, see any member of the executive or the editor at any of the club's regular meetings.

LIBRARY ADDITIONS

NEW 64 DISK:

```
"C64 COMAL 0.14  " 01 2A
  5  "BOOTCOMAL"          PRG     16  "RECURSIONS"          PRG
131  "COMAL80.CAN"        PRG     10  "FORMATTER"           PRG
  1  "----------------"   SEQ      1  "FILE'TO'PRINT"      PRG
  1  "}--------------<"   SEQ      1  "FILE'TO'SCREEN"     PRG
  1  ">ERROR MESSAGES{<"  SEQ      7  "DISK COMMANDS"      PRG
  1  ">  FILE      <"     SEQ     11  "C64 COMAL INFO"     PRG
  1  "}------------ -<"   SEQ      3  "REMOVE COMMENTS"    PRG
  5  "COMALERRORS"        SEQ      6  "SEE'ROLL/DEMO"      PRG
  1  "}------------ --<"  SEQ      5  "SEE'PAGE/DEMO"      PRG
  1  ">FILE GENERATOR{<"  SEQ      3  "CURSOR/DEMO"        PRG
  1  "}---------- ----<"  SEQ      2  "VALUE/DEMO"         PRG
 10  "GENERROR.E.L"       SEQ      2  "SHIFT/DEMO"         PRG
  1  "}---------- -----<"  SEQ      2  "JIFFY/DEMO"         PRG
  1  ">AUTO BOOT PROG{<"  SEQ     10  "QUICKSORT/DEMO"     PRG
  1  "}---------- ------<"  SEQ     4  "JOYSTICK/DEMO"      PRG
  6  "HI"                 PRG      3  "PADDLE/DEMO"        PRG
  1  "}-------- -------<"  SEQ      7  "DISK'GET/DEMO"      PRG
  1  ">SEQ DATA FILES{<"  SEQ      4  "LOGICAL'OPS/DEMO"   PRG
  1  "}---------- -------<"  SEQ    1  "}-- ------------<"  SEQ
 21  "INFORMATIONS3NOV"   SEQ      1  ">  SPRITE DATA  <"  SEQ
 42  "INSTRUCTIONS0.14"   SEQ      1  ">   FILES      <"  SEQ
  1  "}------ ---------<"  SEQ      1  "}-- ------------<"  SEQ
  1  ">COMAL PROGRAMS{<"  SEQ      4  "MOB2"               SEQ
  1  "}---------- -------<"  SEQ    1  ">SANTA"            SEQ
  6  "SEE'INFORMATION"    PRG      1  "}--- ------------<"  SEQ
  4  "SEE'INSTRUCTIONS"   PRG      1  ">END OF COMAL  <"   SEQ
 20  "LOGO'BOOK'SAMPLE"   PRG      1  ">DEMO PROGRAMS <"   SEQ
  4  "SNOWFLAKE"          PRG      1  ">              <"   SEQ
  6  "SPRITE/TURTLE"      PRG      1  ">THE FOLLOWING <"   SEQ
 34  "LOGO EMULATOR"      PRG      1  ">TWO PROGRAMS  <"   SEQ
 10  "LANDER"             PRG      1  ">ARE WRITTEN IN<"   SEQ
 21  "MUSIC"              PRG      1  ">   BASIC      <"   SEQ
 11  "SANTA'GAME"         PRG      1  ">              <"   SEQ
  3  "BOUNCE"             PRG      1  ">DO NOT LOAD  <"   SEQ
 21  "SPRITE'DESIGNER"    PRG      1  ">THEM INTO    <"   SEQ
  7  "READ'DIRECTORY"     PRG      1  ">COMAL.       <"   SEQ
  8  "PRINT'DIRECTORY"    PRG      1  "}------------ <"   SEQ
 15  "EXPRESSION"         PRG     17  "1541BACKUP(FREE)"    PRG
  8  "UTILITIES"          PRG     16  "SINGLE FILE COPY"    PRG
667 BLOCKS FREE.
```

NEW VIC DISK:

```
"CCC VIC AJ      " MJ 2A
  4  "VIC MEDGE"          PRG      4  "BAR GRAPH          D" PRG
  4  "DIRECTORY"          PRG     11  "SX-DOS FIGHT       G" PRG
  9  "STOCK MARKET      G" PRG     7  "VIC MOZART         M" PRG
  7  "SKIING            G" PRG     11  "ROCKS             G" PRG
 15  "CUTOFF            G" PRG     10  "RAIN WALKER        G" PRG
  5  "NUMERIC KEYPAD    U" PRG     30  "TYPE-TUTOR 8K      E" PRG
 44  "SEM MO. CAL 8K    P" PRG      7  "TYPING TEACHER     E" PRG
 15  "SEM YR. CAL 8K    P" PRG     11  "GOTCHA! JS          G" PRG
  8  "SEM APPT. CAL     P" PRG     43  "FINANCES 8K        B" PRG
  8  "SNAKES + LADRS    G" PRG     16  "BUDGETING 8K       B" PRG
  7  "TARGETSHOOT JS    G" PRG      9  "CIRCUS JS          G" PRG
  4  "COMPUTE PRFRDR    U" PRG     11  "CIRCUS PART 2      " PRG
 32  "DISKMASTER 16K    U" PRG      7  "BAJA 1000          G" PRG
  4  "FUGUE             M" PRG      8  "BA2"              " PRG
  4  "VIC PIANO         M" PRG     11  "NEVETS JS          G" PRG
  1  "LADDERS 8K        G" PRG      7  "TANK MANIA 2JS     G" PRG
 19  "L2"                 PRG      9  "TM2"              " PRG
 19  "MONTHS 8K         P" PRG      3  "VIC-PET CONV.     U" PRG
 31  "ECONOMICS 8K      B" PRG      9  "TUNNEL RUN JS      G" PRG
  9  "MEMORY UTILITY    U" PRG     15  "SGL DRV COPY"          PRG
  5  "ADDITION DRILL    E" PRG
1698BLOCKS FREE.
```

NEW VIC TAPES:

```
"CCC VIC TAPE 13 " 13 2A
 12  "DIRECTORY"          PRG      4  "COMPUTE PRFRDR    U" PRG
  7  "PROG CLASS VIC"     PRG     32  "DISKMASTER 16K    U" PRG
  9  "STOCK MARKET      G" PRG     4  "VIC PIANO         M" PRG
  7  "SKIING            G" PRG      1  "LADDERS 8K        G" PRG
 15  "CUTOFF            G" PRG     10  "L2"                 PRG
  5  "NUMERIC KEYPAD    U" PRG      8  "SNAKES + LADRS     G" PRG
 44  "SEM MO. CAL 8K    P" PRG      6  "FUGUE             M" PRG
 15  "SEM YR. CAL 8K    P" PRG     19  "MONTHS 8K         P" PRG
  8  "SEM APPT. CAL     P" PRG     31  "ECONOMICS 8K      B" PRG
  7  "TARGETSHOOT JS    G" PRG     10  "VIC DT"             PRG
402BLOCKS FREE.

CCC VIC TAPE 16 " 16 2A
 12  "DIRECTORY"          PRG     16  "BUDGETING 8K       B" PRG
  7  "PROG CLASS VIC"     PRG      9  "CIRCUS JS          G" PRG
  9  "MEMORY UTILITY    U" PRG     11  "CIRCUS PART 2      " PRG
  5  "ADDITION DRILL    E" PRG      7  "BAJA 1000          G" PRG
  4  "BAR GRAPH         D" PRG      8  "BA2"              " PRG
 11  "SX-DOS FIGHT      G" PRG     11  "NEVETS JS          G" PRG
  7  "VIC MOZART        M" PRG      7  "TANK MANIA 2JS     G" PRG
 11  "ROCKS             G" PRG      9  "TM2"              " PRG
 10  "RAIN WALKER       G" PRG      3  "VIC-PET CONV.     U" PRG
 30  "TYPE-TUTOR 8K     E" PRG      9  "TUNNEL RUN JS      G" PRG
  7  "TYPING TEACHER    E" PRG      4  "FAST FIND-TAPE    U" PRG
 11  "GOTCHA! JS        G" PRG     10  "VIC DT"             PRG
 43  "FINANCES 8K       B" PRG
3938BLOCKS FREE.
```

THE COMPUTING CYNIC

By MARK JACQUES

I can't really understand *Compute!* Magazine's approach to the programs which they publish. Recently there have been threatening dictums in both *Compute!* and the *Gazette* saying that people who type in these programs are forbidden from giving them to anyone else unless that person also owns a copy of the magazine in which they appeared.

Although this makes sense in the fact that many of their programs are dependent upon instructions in the magazine for running them properly, such an approach may come as a shock to old-time computerists who assume that anything published in a magazine is "public domain." Jim Butterfield doesn't go along with *Compute's* policy, despite the fact that he is one of the magazine's associate editors. Butterfield regards programs as like "recipes" -- which means not only can you do them, but improve on them.

Compute's attitude is unreasonable, since many of their programs contain bits and pieces from programs which have previously been "copyrighted", including the simplest joystick routines. Obviously one thing is certain here -- some programs are more "public domain" than others.

Other magazines, such as *Power Play*, published by Commodore, have a more intelligent attitude to their programs, placing them in the category of public domain with no strings attached.

Similar problems have arisen with the *Gazette's* soon-to-be-released disks. An editorial by editor-in-chief Robert Lock in the April *Gazette* issue suggests that people who order disks and distribute them to others who don't have the magazine (such as users' groups) are not only guilty of "theft," but are depriving the people at *Compute!* of income.

While I agree that the magazine is hardly a charity, one wonders why the disks are being released at all if they will create so much hassle. Are they designed to be a convenience to readers who are weary of typing in programs themselves, or are they strictly a money-making proposition? Is *Compute!* going to run all over the country prosecuting people for such a crime?

I would rather the magazine stuck to going after people who are *selling* their programs, and leave the business of *sharing* programs alone. After all, most people that I know read both *Compute!* and the *Gazette*. Such a benificent move on the part of the magazine would certainly be a great gesture of goodwill to its readers, and would probably *encourage* rather than *discourage* sales of the magazine and *Compute's* other products.

Editor Lock asked for reader feedback in his editorial, claiming that "we're fundamentally opposed to such protection. We had decided we would be able to approach it all differently." The address, in case you're interested in taking him up, is P.O. Box 5406, Greensboro, N.C. 27403, U.S.A.

***

Still speaking of *Compute!*, one of our members got a big shock recently when he received a letter from them. He had written to their "Bug-Swatter" column pointing out that one of the games they published didn't work correctly. So he suggested a few minor changes.

The letter he received back from the magazine said there was nothing wrong with the program (quite true -- it *would* run without the changes) and that he should spend some time reading some enclosed Xerox pages on "how to type in programs"!

Compute!, it seems, is particularly sensitive about criticism for typographical or any other kind of errors, despite the fact that these occur frequently. Some errors in programs printed in magazines eventually make it uncorrected to their books as well. I can recall one program with a high score routine which answered the prompt "Play Again? (Y/N)" with a RUN, which promptly wiped out all variables, including the high score!

AN AMUSING ALLEGORY

By PHILLIP S. LAWRENCE

The scene opens in a company cafeteria. John is seated at one of the tables eating his lunch, and browsing through a stereo equipment magazine. Bill and Dave enter, seating themselves at the same table.

***

BILL: Oh! Is that the new Farkle Mark IV digital tweeter system?

JOHN: Yes, I'm thinking of getting one to go with my Instrumental Bass Systems Model six horizontally polarized mid range augmenter.

BILL: Personally, I wouldn't have any IBS equipment. I really think the Common Door equipment is just as good, and costs a lot less.

JOHN: Sure, but there aren't any high band wobbulators available for it. It's just not a serious stereo system.

DAVE: You guys and your stereos! How much money have you spent on them?

JOHN: About 7 or 8 thousand, but it's really good stuff. Well worth it.

BILL: Well I only spent half that much, and mine is pretty good.

DAVE: Eight thousand? Wow! I couldn't justify a tenth of that. I mean what could it do for me? How could it make money for me? Could I use it to do practical things around the house?

JOHN: You don't understand, Bill, it's fun to play with. It's relaxing to listen to. Besides, you could make money with good equipment by setting up a recording studio, or by becoming a stereo consultant.

BILL: Sure, and what's more practical than providing yourself with pleasure and entertainment.

DAVE: All right, assume that I could justify a purchase on those grounds, which system should I buy? There are so many to choose from, and the cheap ones seem to be just for playing Disco. I would probably want one that could play more serious music.

JOHN: Well the IBS system would be perfect for you. It will play all the serious stuff, and besides, they use CP/M, you know, Compatible Playing for Music. That means that there is an enormous record and tape base available for the IBS.

BILL: Just a minute John, the Common Door system may not have quite as much soundware available, but it has everything you need. On top of that, the Common Door has a lot of nice features built in that you don't have on your system. Things like full song editing and less restrictive song titles.

DAVE: Listen, all this sounds like APL to me. Which is the best system?

JOHN: Just don't go with the Common Door stuff, it's only good for playing Disco. Get yourself an IBS, you'll never regret it.

BILL: Don't listen to that garbage, Dave. Of course there is serious stuff available for the Common Door. The thing he hasn't told you is that there isn't ANY Disco available for the IBS.

***

The discussion grows into an argument, and the only thing that all agree on is that one should never, under any circumstances, buy the Audio Stack system. Dave leaves the cafeteria with the hint of a smile on his face. He enjoys getting these two started. He also secretly owns a system marketed by a well known watch manufacturer, and wouldn't dream of admitting it.

DEADLINE FOR NEXT ISSUE'S ADS AND ARTICLES: SUNDAY, APRIL 22. NO EXCEPTIONS, PLEASE!

WORD PROCESSING

By MIKE QUIGLEY

(Note: While this review is based on the VIC-20 version of HesWriter, the C-64 version, released later, is reportedly highly similar.)

Two important features to look for with a word processor are the ability to access text and the ability to manipulate this text easily. In both these areas, HesWriter is disappointing.

The maximum length of a HesWriter line is 18 characters. There is word wrap, which moves words which won't fit at the end of one line, unbroken, to the beginning of the next line. Copy is entered a line at a time, and it is not possible to edit lines above the one you are currently working on without entering the Edit Mode, where alterations can be made only with some difficulty.

For example, if you want to insert a lot of material in the middle of a previously typed line, you have to first determine the number of this line by entering the Number Mode, retain the number (how? with a pencil?), then create the new text elsewhere, insert it *before* the line in which you want to insert it, and then delete any extraneous words. If this sounds confusing, believe me, it is. Furthermore, when inserts and edits are made in this way, extra spaces are sometimes created which are printed out along with the text. It's also possible to find yourself in editing situations where copy is frozen and nothing further can be inserted without going through variants on the above procedure. Deleting lines or copying lines from one location to another are both handled in a similar manner, and returning to the main Entry Mode sometimes leaves you with no copy on screen to refer to.

HesWriter *does* work quite well with simple projects which don't require any fancy formattin. It is also well suited to use with low-cost printers like the VIC-1525, since it has no provision to use features like italics, condensed print, superscript, and so forth. In fact, it won't even access the expanded print on the VIC printer, created with CHR$(14)! Printing with HesWriter proceeds at a slow but acceptable pace highly reminiscent of some BASIC word processors, a similarity which is occasionally enforced by BASIC error messages.

There are various format commands in HesWriter designed to determine the four margins of the printed page, number of spaces between lines, position of headers (titles), and the use of centred or justified copy (the latter with aligned margins). Not all of these worked with a Gemini-10 printer and Cardco interface combination and I couldn't determine if this was because of the printer or because I wasn't using acceptable values in these commands. There are usually no minimum values established in the manual. A value of "0" for the left margin produced erratic results in printout, for example.

HesWriter saves to tape or disk and allows you to change screen and lettering colors. There is a "search" mode, which does not work very well -- some instances of a given sequence were bypassed. Among the things you can *not* do is access the disk directory or other functions like Scratch and Validate.

The shifted space in HesWriter (CHR$(160), which is used as a "fixed space", causes problems with non-Commodore printers where this is a printing graphic character. As well, HesWriter places a double space after each sentence when printing out. This is the procedure we were all taught to do in high school typing class, but it may not always be desirable, and it is not possible to disable this feature.

HesWriter is a world unto itself as far as compatibility with other word processors is concerned. It saves programs on disk as sequential files, which means it's not possible to load in material from most major word processors, which save to disk with program files. Even loading from word processors which do save sequentially proved fruitless, since HesWriter's files are created in a different manner. Tape files were also incompatible.

In short, at a price roughly equivalent to that of Cardco's Write Now!, which offers many more features and considerable versatility in manipulating copy, HesWriter does not have a great deal to offer. One would hope, in fact, that HesWare would issue a revised version of HesWriter for the VIC-20 incorporating features such as found in more recent word processors and which was the equal of their high-quality games and other products.

***

Another word processor I'm not too crazy about is Totl.Text, which exists in versions for the VIC and 64. It's just too darn slow, being written in BASIC. Although it has piles of interesting features, I find the general slowness of it to be a real pain in the neck. In its advanced version for the VIC, Totl.Text is in two parts. You create the copy with the first part, and run it with the second. If you see an error while you're running off your copy, you have to abort the whole thing, reload the "creating" part (which takes considerable time when you're using tape), fix it, and then rerun it using part two. This comes close to being "user stupid" in my eyes.

I've been quite surprised to see rave reviews of Totl.Text in some computer magazines. These reviewers sound like a primitive from Borneo coming to North America who discovers the bicycle to be a miraculous form of transportation, while unaware of faster methods of travel like the car and airplane. Why bother with Totl.Text when there are so many more efficient word processors in machine language?

• USENET NEWS •

Ed. Note: The following comes to us courtesy of club member Keith Mosher, who obtained it from USENET. It is reprinted for your interest only -- we can't vouch for the accuracy or reliability of the "tips" suggested.

***

I recently found a solution to the "sparkle" problem of the C-64. Apparently there are levels of severity and, as a result, two fixes.

The first is to place a 500pF cap from the AEC pin of the VIC chip to ground. The AEC lead is accessible from a test pad just to the left of the outside of the RF shield of the VIC and clock circuits. Then the ground side was just next to C34, an electrolytic cap right next to the SID chip.

I started with a 470pF cap and found that it cured most of the sparkle. I added another 150pF cap in parallel (physically, not electrically) and improved the performance even more. I added another 47pF cap (total 667pF) and the problem is almost gone. Now I only occasionally see a one or two bit error instead of the whole character being constantly garbled.

The other fix is to replace the character generator ROM, a $22 expense just for the chip. I've also heard of these being combined.

Mike Nelson
Illinois

***

Here's another difference between V1 and V2 C-64s. When a V1 powers up it sets a bit in the D000's somewhere that apparently enables collision detection between sprites. This memory location is one of the VIC II chip's control registers. If the right bit isn't set then sprites can bang into each other all day and nobody knows. Well, needless to say V2s don't have this set when they are powered on. I found this out the hard way when I brought my new game home and found out I couldn't die because no matter what hit me, I just smiled and kept on going. Racked up a pretty high score, but wasn't much of a challenge.

And you thought it was only blank screens!

Paul Maiorello
New Jersey

***