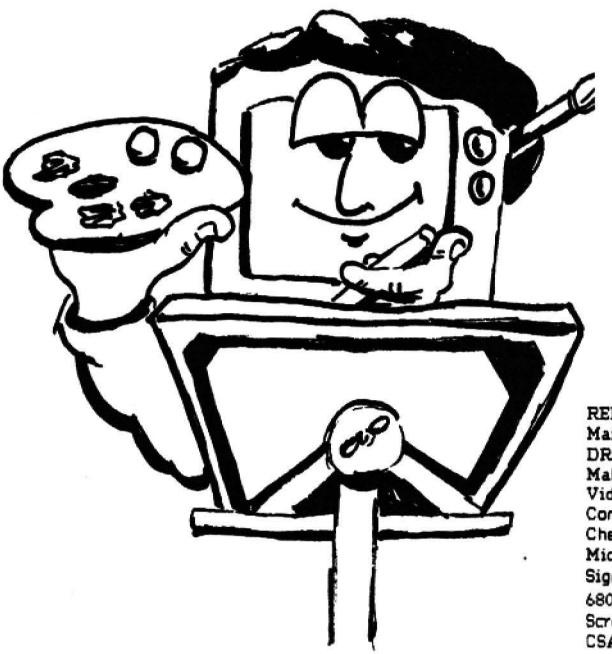
$2.50

Color Computer News

November/December 1981
Volume 1 Number 4



Color Computer News is published bi-monthly by REMarkable Software.

REMarks
by Bill Sias

The price increase in last issue's ad was in error. Someone got overly greedy and announced next years prices effective this October and even that price was wrong. My sincerely appology for any inconvience this may have caused anyone. Those folks that missed the "deadline" were credited with 8 issues instead of six. The actual price starting January 1, 1982 is $18.00 dollars for 12 monthly issues.

To clear up some apparient misunderstanding, Computer Plus includes a free copy of Color Computer News with every Color Computer that they sell. They do not, however, give away free subscriptions.

The first came from a fellow named Victor Andrews in the form of a phone call. It turns out that Soft Sector Marketing sells some CC software as well as their Models 1 and 3 programs. Anyway Andrew has donated his disassembler program to all of us as my usual Assembler article. Thank you Andrew!!

Transformation Technologies' C.C. Writer was reviewed in a syndicated newspaper column called breaker breaker by Fred Simon. The article appeared Sunday September 13 and talks about CB, Home computers and the "lack-at least on the part of Tandy Corp. (Radio Shack)" of "a good program that will do 'word processing' for the 16K or 32K TRS-80 Color Computer". Mr. Simon then talks specifically about Bill Dye and C.C. Writer and promises to keep everyone informed about new software. It's good to see more support for the Color Computer especially in a newspaper column.

We are continuing to grow at a rapid pace but I need your help. The vast majority of the articles are being written by a minority of the readers. We need your article NOW. In case you didn't know we do pay for articles and we have the easiest submission requirements of anyone in the industry. Acceptable formats are, in order of my preference: Color Computer tape or disk data file, TRS-80 Model 1 tape or disk, Flex disk, direct link via modem, double spaced typewritten, or hand written. Although all of the above formats are acceptable and I am not overly fussy about any of the above but the following will hinder yourchances of being published: hand written programs of more than a few lines or typewritten/printed programs of more than a page. All material submitted becomes our property and will not be returned unless submitted with a self addressed return mailer with sufficent postage attached. Acceptable topics are: programs of any sort, tutorials, hardware mods/descriptions, programming tricks/tips, software reviews, hardware reviews. The list is almost endless.

We have a few copies of Ole #1 back in stock. Please, if you would like one, send your order in under separate cover and include the word "Back Issue" between REMarkable Software and P.O. Box 1192. The cost is $2.50 and if you would like first class mail include an additional $.75. First class copies will be mailed every Friday, others will wait until we have enough orders to send bulk. They will be sold on a first come first served basis and when these are gone there will be no more. When we do run out I'll return your check, no charge cards. Number 2 is in stock, same deal except include $.75 for first classs postage. Number 3 is gone.

For several months now we've talked about changing into a monthly publication, well I have good news for you. Starting with the January issue we will be monthly! We have more than doubled the number of pages in the first issue now and I'm discovering that we're having a rough time keeping the information current so we've done it! I'm depending on you folks to remember that we need those articles more than ever.

Dear Bill,
The price was too good to be true, a 32K memory kit for $12.00 (less memory). This is just what we all have been looking for. An easy way to upgrade our Color Computers to 32K and for only $12.00.
Yes it is too good to be true, but Culpepper Computer Designs' ad says just that. "All parts and detailed instructions (less memory)". Now I'm one to try to save a little money when I can, so I ordered the kit. It took about four weeks to get here. That isn't too bad for only 600 miles, even with the Postal Service what it is. But what I received is the real story. I got a piece of wire 18" long and a small clip. Yes, that's it. Oh yes, the detailed instructions. The instructions are a real joke, no reference to any registers in the connection from Pin 4 to Pin 35 of the SAM. The test program POKEs the number 16 into memory, that's it.
I think that this type of advertising borders on illegal. This is another example of advertising ripoff. Anyone who wants to upgrade should follow the instructions in the July/August issue of CCN and save their $12.00.
The address of Culpepper Computer Designs is: 502 S. East Street, Culpepper, VA 22701.
But save your money this is a real ripoff.
Bob Vaughan
120 Eastpoint Dr.
Charleston, WV 25311

Dear Sir:
Take away a little heat.
With 16K of RAM I found that the temperature under the top cover was 49.5 degrees above the ambient air after running 2 hours. By painting the top cover, part # AZ5846, a dull black inside and outside, (being careful to not paint the fingers where contact is made with the top cover support), the temperature was 42.5 degrees above the ambient air temperature. The temperature rise will vary somewhat with the program being run.
How much does 32K increase the temperature?
I have also noticed that the front cover of the magazine says that "Color Computer is a trademark of the Tandy Corporation". I do not believe this is correct. TRS-80 Color Computer, yes, but not Color Computer. I have a device called the Color Computer which was made years ago, before Radio Shack even thought of their Color Computer.
Charles Worstell
36012 Military Rd S.
Auburn, WA 98002

* You are correct. When I typeset the trademark notice back in April I forgot about the Micro Chroma and the various other "Color Computers".

In the last two issues of the Microcomputer Newsletter, Tandy's official support for their computers, the statement has been made that it was impossible to recover the last graphic page when using Extended Basic. In other words, there is no possible way to execute a PCLEAR 0. It gives me great pleasure to again reinforce the fact that a group of hobbyists with a challenge can accomplish more that a corporation with paid professional programmers. Thanks to the challenge put forth in those newsletters hobbyists all over the country have proven that we know more about the Color Computer than Tandy does. To wit:

To PCLEAR 0 type:
POKE 1536,0: POKE 25,6: POKE 26,1: NEW
Thanks to Phil Beistel
Dave Bodnar

If you haven't heard about the "PCLEAR 0" yet, try this:
POKE 25,6: POKE 27,6: POKE 29,6: POKE 31,6 <ENTER>
PRINT MEM should give 14631 .. and it is useable for programs contrary to what RS says it's latest Microcomputer Newsletter.
R. Wayne Day
1779 Continental Dr.
Blue Mound, TX 76131

Greetings:
Since memory locations 25-32 are pointers, changing the point should make the "other" page of graphics memory open to programmers. On power up 25, 27, 29 and 31 hold 30 but after PCLEAR 1 those locations hold 12 therefore POKEing 6 into those locations moves the point back to the start of the graphics pages.

Pointer:
25, 26 = beginning of Basic program
27, 28 = beginning of simple variables
29, 30 = beginning of subscripted variables
31, 32 = beginning of free memory
H. D. Bassett

This is just a sampling of the letters I've received on this subject. The moral of the story is: Never tell a hobbyist that he can't.

I need help on the 4 pin (Din) I/O port for my Ham Radio Interface. I have been unable to find this info for serial interface. I don't need modem. I just want to talk to my HK terminal any help or reference will be appreciated.
Jesse F. Lee
W5GCJ
3105 Edgewood
San Angelo, TX 76903

For Mark Lockwood & Others
You can get Extended Basic for $90.00 from Sound Center Radio Shack, White Rock Shopping Center, Los Alamos, NM 87544 Phone (505) 672-9824
Roland C. Wong
West Covina, CA 91792

Dear Sir,
I have your Sept/Oct issue of Color Computer News. I tried programming "Spellit" pg 27. It works great but I can't get it to record the words and definitions on tape. I'm not very familar with programming yet and hope that you can tell me why I'm having this problem. I notice that line 460 immediately follows line 400. Are lines 410 - 450 missing? Please help!

Lillian V. Panagakos

* Oops! Here are the missing lines!
410 INPUT #-1,N
420 FOR I=1 TO N
430 INPUT #-1,W$(I): INPUT #-1,D$(I)
440 NEXT I
450 CLOSE
We print the program listings directly from a running copy of the program, but apparently when we cut it for layout those lines went on the floor instead of the layout sheet.

Gentlemen,
Great magazine! Has anyone devised a way to "DRAW" with angles other than the 45, 90 degrees etc given as options? I.E. DRAW at 37 degrees?
Sincerely
Ralph Coleman
2306 Griffin Rd
Churchville, NY 14428

Dear Bill;
Enclosed are two listings for a screen dump to a printer. One was assembled on Micro Works SDS80C editor assembler monitor. The other is in Basic which loads into RAM in the Exatron "Thing".
CCN is getting better every issue. Keep up the good work. I especially like the articles on machine language. They are very helpful to a novice like me.
Sincerely,
James C. Whitaker
2821 Reagan #102
Dallas, TX 75219

```
100 FOR I=1 TO 50
110 READ B
130 POKE 51500+I,B
140 NEXT I
150 DATA 16, 142, 4, 0, 198, 32, 166, 160,
129, 63, 34, 4, 139, 96, 32, 6, 129
160 DATA 95, 37, 2, 128, 64, 189, 162, 191,
90, 38, 234, 134, 13, 189, 162, 191
170 DATA 142, 0, 111, 18, 18, 48, 31, 38,
252, 16, 140, 6, 0, 37, 212, 57, 0
210 DEF USR0=51501
220 A=USR(0)
```

```
            NAM  SCRNDMP1 8/
25/81
*
**SCREEN DUMP TO PRINTER
**ASSEMBLED ON MICRO WORKS
**SDS80C ROMPAK
**BY JAMES WHITAKER
**DALLAS, TEXAS
*
KEY     EQU $A000 POLL KEYBOARD
VIDRAM  EQU $400  START OF VIDEO
PRINT   EQU $A2BF SEND A TO PRNTR
*
**PRINT OUT ROUTINE-SCREEN DUMP
*
START    LDY #VIDRAM ADDR 1ST CHAR
A@        LDB #$20    PRINT 32 CHARS
B@       LDA ,Y+     GET CHAR
            CMPA #$3F   IS IT BELOW 'A
            BHI C@      NO, TRY AGAIN
            ADDA #$40   CHANGE TO ASCII
            BRA D@      SEND IT
```

```
C@      CMPA #$5F   IS IT ABOVE 'Z
        BLO D@      NO, SEND IT
        SUBA #$40   CHANGE TO ASCII
D@    JSR PRINT   SEND IT TO BUFF
        DECB        IS IT LINE END
        BNE B@      NO, GET NEXT CHAR
        LDA #$0     CARRIAGE RETURN
        JSR PRINT   PRINT IT
        LDX #$111   1 MILLISEC DELAY
        NOP
        NOP
T@      LEAX -1,X
        BNE T@      END DELAY
        CMPY #$600  IS IT END VID
        BLO A@      NO, GET NEXT LINE
*
**GO TO BASIC
*
        RTS         RETURN TO BASIC PROG.
        END
```

Dear Bill,
Here is my check for an additional 6 issues. Bill, I will have to agree with Mr. Harrison (Mail Call Sept/Oct). CCN is a tool I use as Reference, it's more than a magazine. Bill, did you know that Radio Shack is now selling a 32K upgrade for the CC, the computer they said could not be upgraded more than 16K!! HA HA
Say Bill I thought we were going to be able to buy the programs on tape that are in CCN. Keep up the good work and let's go monthly.
A Happy Reader,
Robert Salyer

* The 32K upgrade uses 64K chips with the upper bank uncertified. You can buy the programs on either tape or Exatron format disks. The prices are $7.95 for cassette and $10.95 for the disks. They also include some extra programs that readers have submitted. I haven't said much about it because they don't contain many programs yet.

Dear Sir;
I have two questions to ask you:
1) Is there any way to bypass the break key from BASIC?
2) Is it possible to divert error messages so they will not halt the execution of a BASIC program (i.e. ON ERROR GOTO)?
I have found a need for both of these in my every day programming.
Yours Truly,
Tom Markson
366 Newburn Drive
Pittsburgh, PA 15216

Dear Sirs,(CCN)
If CCN prints machine language or assembly language programs would you please explain to novices like me how to run them!! or what we need in order to run them. Be detailed. Yes, I have Extended Basic, but we are not experienced in changing programs to Extended BASIC from 4K. So please take the trouble to PRINT ALTERNATE LINES COMPLETELY in CCN.
With Regards
Joel Cohen
5 Terry Terrace
Livingston, NJ 07039

Dear Bill;
After our conversation the other day I felt that I should drop you a quick note to explain just how I came to purchase my Color Computer. My neighbor had just informed me that he had this new computer that I ought to see. I must admit the whole thing sounded rather boring to me, but I was going to be a nice guy and go see this silly computer of his. He showed me the usual things that one feels he must do to impress someone with their new "toy" and to this point I was not too impressed. I have to tell you Bill, that I have never been too taken with the arcade type games and sure enough this was the next item that he forced upon me. OK, so this little keyboard can play games. Big Deal!!!
What happen next, Bill, is quite incredible. out came this adventure game called "Black Sanctum". It was my turn to operate the computer, there I was standing out in the woods with a cabin in sight and it was starting to snow. The computer asked me what I should do. With a little prompting from my neighbor, I said, "go cabin", and much to my surprise the computer informed me that I was at the cabin door and "what should I do". A strange feeling began to overcome me and I felt that I was being transported to another medium. kind of a twilight zone that had taken control of me. This little game had me hooked for the next three hours and then I had to tear myself away. The following day at work, I found myself calling my neighbor to see if he had gotten any further in the adventure. I rushed home early from work and stopped at his house to take another look at this "game". Now we were in a maze of passages and corridors that ran either under or behind the cabin. In the ensuing days I spent hours in front of his computer working on

the "Sanctum" with a glazed look on my face. I dragged my wife over to see this machine that was taking her husband away from her. I secretly hoped that she too was going to be taken with this thing that by now had a tight hold on me, for, I knew that sooner or later I was going to have to talk to her about this computer that I just had to have.

There is no question, Bill, that at this point I was hooked and just had to have that computer. To further drive me crazy, my neighbor informed me that he had solved the mystery of the "Black Sanctum" but he would not tell me how. That did it. It was off to the computer dealers to find the best price and get that little bugger in my home so that I could go back to work on the "Sanctum". And I did. I too have solved the mystery of the "Sanctum" after several hours of talking first to my Color Computer and then myself. Now I wonder just who is this who is clever enough to write a program of this nature. I can live without ever knowing his name, but I have to warn him, My wife would like to have a few words with him...

Sincerely Yours
Thomas L. Mix
3424 College N.E.
Grand Rapids, MI 49505

```
1 REM THIS IS TIEFITE BY JML 4-4
-81
2 CLS:PRINT@165,"WELCOME TO TIE
FITE"
3 PRINT@480,"JML 4-4-81"
4 FOR D=1 TO 2000:NEXT D
5 A=0:SC=0
6 PRINT@320,"THERE ARE 10 TIE FI
GHTERS"
7 PRINT@362,"TO SHOT DOWN JEDI"
8 FOR D=1 TO 2000:NEXT D
10 CLS(0):GOSUB 500:FOR D=1 TO 1
000:NEXT D:GOTO 400
20 SET(H,V,0)
30 SET(H-1,V,0)
40 SET(H-1,V-1,0)
50 SET(H-1,V+1,0)
60 SET(H+1,V,0)
70 SET(H+1,V-1,0)
80 SET(H+1,V+1,0)
90 RETURN
100 SET(H-1,V,0)
110 SET(H,V-1,0)
120 SET(H,V+1,0)
130 SET(H-2,V,0)
140 SET(H-2,V-1,0)
150 SET(H-2,V-2,0)
160 SET(H-2,V+1,0)
170 SET(H-2,V+2,0)
180 SET(H+1,V,0)
190 SET(H+2,V,0)
200 SET(H+2,V-1,0)
210 SET(H+2,V-2,0)
220 SET(H+2,V+1,0)
230 SET(H+2,V+2,0)
240 RETURN
250 SET(H,V-1,0)
260 SET(H,V+1,0)
270 SET(H+1,V-1,0)
280 SET(H,V,0)
300 X=JOYSTK(0):Y=31
301 IF X>63 THEN X=63
302 IF X<1 THEN X=1
310 SET(X,Y,0)
315 RESET(X,Y)
320 P=PEEK(65280)
325 IF P=126 OR P=254 THEN 330 E
LSE RETURN
330 FOR M=30 TO 1 STEP-1
331 SET(X,M,0)
332 RESET(X,M)
333 IF X=H AND M=V THEN 340 ELSE
 334
334 IF X=H+1 AND M=V OR X=H-1 AN
D M=V THEN 340 ELSE 338
338 NEXT M
339 RETURN
340 SC=SC+1:IF SC=10 THEN 2000 E
LSE 341
341 GOSUB 800:GOTO 400
400 H=RND(53)+5:V=RND(25)+3
401 GOSUB 300
405 SET(H,V,0)
406 FOR D=1 TO 10:GOSUB 300:NEXT
 D
407 GOSUB 300
409 CLS(0)
410 GOSUB 20
420 GOSUB 300
430 FOR D=1 TO 10:GOSUB 300:NEXT
 D
435 CLS(0)
440 GOSUB 100
445 GOSUB 300
450 FOR D=1 TO 10:GOSUB 300:NEXT
 D
460 CLS(0)
464 SOUND 200,3:SOUND 185,1:SOUN
D 150,1
465 A=A+1
466 IF A=5 THEN 467 ELSE 470
467 CLS:PRINT@170,"TOO BAD JEDI!
"
```

continued on page 36

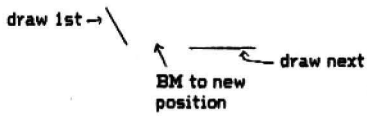
DRAW II

by Don Inman

This is a continuation of the discussion of the use of the DRAW statement begun in the September October issue of Color Computer News. In that issue, several options used with the Extended Color Basic DRAW statement were discussed:

B for Blank - don't draw
M for Move - move to new X,Y position
U,L,D,R,E,F,G,H - draw commands for various directions
Relative Motion - move to a new position relative to the last position
Catenating strings to form the DRAW statement

Draw statements are executed quite quickly and lines can be connected to create geometric shapes as illustrated in the September/October issue. We also showed how to "lift the drawing pen" between lines to a new starting position as:



The No Update Option

You can also draw several lines from some starting point but in different directions. The N option says "No update", which means "return to the origin of this line after drawing it."

Selecting a starting point near the center of the screen, you might use:

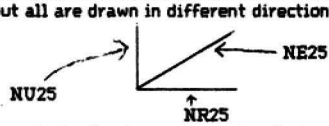
```
DRAW "BM128,96; NU25; NE25; NR25"
```

Draw a line Up 25 units and return to the starting point

Draw a line 45 degrees from Up 25 units and return to the starting point

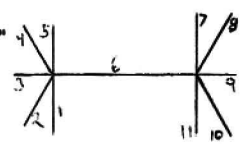
Draw a line Right 25 Units and return to the starting point

When this statement is executed, three lines are drawn. Each line originates at the point 128,96, but all are drawn in different directions.



Let's get a little fancier and draw the following figure using the N option.

```
100 'SET THE SCREEN
110 PMODE 4,1: PCLS: SCREEN 1,1
200 'DRAW FIRST 5
210 DRAW"BM100,96; ND25; NG25; NL25; NH25; NU25"
220 FOR X=1 TO 1000: NEXT X
300 'DRAW NUMBER 6
310 DRAW"R56"
320 FOR X=1 TO 1000: NEXT X
400 'DRAW LAST 5
410 DRAW"NU25; NE25; NR25; NF25; ND25"
420 GOTO 420
```



You could write a variation of the previous program that would draw a "snowflake" pattern.

DRAW II

Angle Option

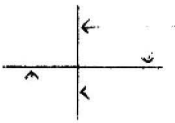
Lines can be rotated 0, 90, 180, and 270 degrees with the Angle command, A. The amount the line is rotated is designated by a subscript following the letter A:

A0 - draw at the direction specified
A1 - add 90 degrees to the direction specified
A2 - add 180 degrees to the direction specified
A3 - add 270 degrees to the direction specified

This option might be used in a FOR-NEXT loop to create 4 different lines.

```
210 A$(1)="A0": A$(2)="A1": A$(3)="A2": A$(4)="A3"
220 FOR X=1 TO 4
230   DRAW A$(X)+"BM128,96;U25"
240 NEXT X
```

The display would produce:

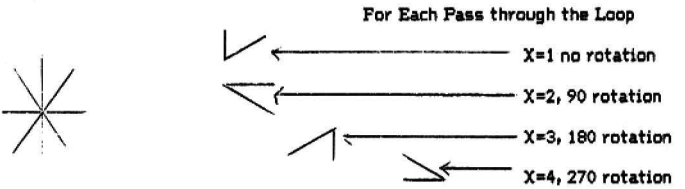


X=1, no rotation = Up
X=2, U+90 = right
X=3, U+180 = down
x=4, U+270 = left

Or, if you want 8 lines instead of four, change line 230 to:

```
230 DRAW A$(X)+"BM128,96; U25; BM128,96; E25"
```

From this change the display would be:



You might try drawing the "snowflake" using this method.

Drawing Nothing

You might think it useless to draw a blank line, but sometimes it might come in handy. Since the Color Computer doesn't like to mix text and graphics on the screen, you probably will want to DRAW some block letters in the graphics mode at times. Suppose you want to draw the letter, F.

By inserting a Blank move at the correct places, you can create the letter with a continuous DRAW statement:

```
DRAW"BM140,80; L25; D50; BU25; R25"
                              ↑— move up 25 but don't draw
```

This creates an "F" with segments shown by arrows (solid are drawn, dotted is not drawn).

Try the following program on your Color Computer to see what it produces.

```
100 'SET SCREEN
110 PMODE 4,1: PCLS: SCREEN 1,1

200 'DRAW A WORD
210 DRAW"BM100,100;L10U20R10;BR15;L10D20R10U20;
BR5;D20R10;BR15;U20L10D20R10;BR5;U20R10D10L10F10"
220 GOTO 220
```

Hint: What kind of a computer is this?

DRAW II

Color

Since this is a color computer, don't you think it's time we put some color into the DRAW statement? This can be done with the command:

C for color —Cx— x is a color code (0-8) selected from those possible for the mode and color set in use

This time we'll select PMODE 3 which provides 4 colors from one of the two color sets:

Color Set 0	Color Set 1
green	buff
yellow	cyan
blue	magenta
red	orange

Insert the Color command at the beginning of the DRAW string:

```
PMODE 3,1: PCLS: SCREEN 1,0
DRAW"C3; BM100,100; L10 U20 R20"
```

This would display:

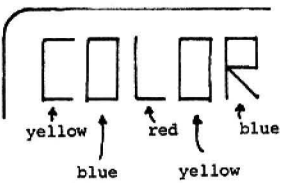
C ←a blue C on a green background

Several C commands may be inserted in a DRAW statement to change colors as in the following program.

```
100 'SET SCREEN
110 PMODE 3,1: PCLS: SCREEN 1,0

200 'DRAW COLORFUL WORD
210 DRAW"C2;BM100,100;L10U20R10BR15;
      C3;L10D20R10U20BR5;
      C4;D20R10BR15"
220 DRAW"C2;U20L10D20R10BR5;C3;U20R10D10L10F10"
230 GOTO 230
```

Now, you see on the display:



Many variations could be made such as:

a) randomly selecting the color codes
b) changing to color set 1
c) changing colors for segments of letters etc.

Scale

The SCALE command lets you enlarge or reduce the size of a display created by a DRAW statement. After a SCALE command has been executed, all absolute and relative motion commands will be reduced or enlarged according to the specified SCALE factor. In other words, the SCALE factor stays in effect (within a given program) until a new scale is specified. The SCALE command is:

S for scale —Sx— x is an integer (1-62)

If no scale factor is specified, the computer uses S4, Scale factors for various values of x are:

x	Scale Factor
1	1/4
2	2/4 or 1/2
3	3/4
4	4/4 or full scale
5	5/4
.	.
.	.
.	.
62	62/4 or 15 1/2 times full scale

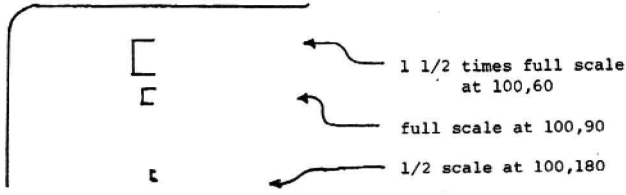
The following program prints the letter C in three different sizes specified by the scale commands:

```
S2 = half scale
S4 = full scale
S6 = 1 1/2 times full scale

100 'SET SCREEN
110 PMODE 3,1: PCLS: SCREEN 1,1

200 'DRAW & CHANGE SCALE
210 FOR X=2 TO 6 STEP 2
220  S$="S"+STR$(X): Y$=STR$(360/X)
230  DRAW$$+"BM100,"+Y$+":L10U20R10"
240 NEXT X
250 GOTO 250
```

Display:



That's all for this time. Next issue, we'll discuss using substrings within a DRAW statement. We'll also show some applications to try to tie all the DRAW options together (tighten the DRAW strings).

Making Education More Colorful.

by David Bodnar

When I was given the opportunity to write an educational column for COLOR COMPUTER NEWS, I had to make several assumptions about the readers of this column and what they would expect to find in it. My first assumption was that they either owned, worked with or planned to purchase a COLOR COMPUTER.

The second assumption was a bit tougher. Why would a reader be interested in a column dedicated to education? As I thought about it I realized that virtually every owner of a Color Computer needed information about the computer and it's operation. I also realized that each of us has a strong desire to educate our family, friends and colleagues about the potential that the COLOR COMPUTER has for us.

This interest is not confined to learning about computers but it also includes using the computer as a tool to teach ourselves and others about a myriad of topics.

My own interest can be traced back nearly 15 years, but it really intensified last year when I purchased a Color Computer for use in my classroom. Since then my school district has funded a research project designed to test the feasability of teaching 5th and 6th grade students with and about computers. I will keep you informed about its progress so that we all may benefit from this experience.

As you may know there is very little educational software available for the COLOR COMPUTER. If we are to make wise purchases we should have some idea of a program's potential and limitations before buying it. I plan on evaluating software by using it myself and, where appropriate, by having my students take a long, HARD look at it.

This column will also contain suggestions for making programs more "AMICABLE" (I hate "friendly") to your audience, regardless of its age. We will try to end with a short program that may be of interest to classroom teachers, students and parents alike. The first of these appears below.

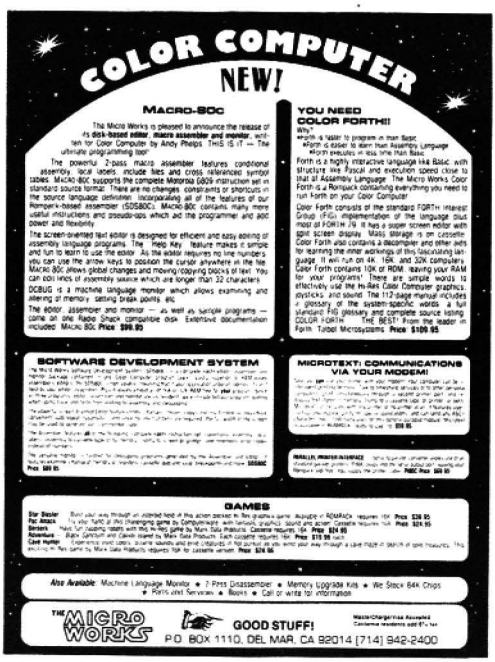
This program began as a tool to help solve letter substitution puzzles that appear in GAMES MAGAZINE. These puzzles consist of a simple letter substition code that turns a common phrase or riddle into a meaningless jumble of letters. For example, the phrase "COMPUTERS ARE LOTS OF FUN!!" could become "DEYNKSLFA MFL ZESA EO OKQ" if all of the C's became D's and all of the U's became K's, etc.

When you run the program it asks you to type a phrase. (The length is limited to 255 characters including spaces and punctuation.) The program then pauses for several seconds to create a random substitution table that will turn all A's to J's, all B's to C's or whatever. It then displays the coded phrase and asks for a letter to change. If you wish to change A's to M's, key A in response to "CHANGE FROM" and M in response to "CHANGE TO". The phrase will appear below the original with all A's replaced by lower case M's. This helps you to keep track of what has been changed. You may not take back a change after it has been entered, but if you get hopelessly lost type "XX" and you will restart with the same phrase.

This program is by no means complete. You may wish to add a feature that will tell you if you have correctly solved the puzzle, although it is usually quite obvious when you are done. You might also add a feature that would allow you to move back 1, 2 or more moves to save starting over if you make errors. It has lots of possibilities ... let me know what you do with it.

```
100 CLEAR 1000: DIM C(26), F1(26), FL(26)
200 CLS: PRINT" TYPE A PHRASE": LINE INPUT B$: CLS: L=LEN(B$)
300 A$=STRING$(L," ")' CREATE SECOND STRING TO BE CHANGED TO CODE
399 '(400-600)-SELECT RANDOM CODE - F1 INSURES AGAINST DUPLICATION OF
LETTERS
400 FOR X=1 TO 26: PRINT@32*6+3,"GETTING RANDOM LETTERS":
PRINT32*7+10,USING"##/26";X
500 C(X)=RND(26): IF F1(C(X))=1 OR C(X)=X THEN 500: ELSE F1(C(X))=1
600 NEXT X: CLS
699 ' (700-1100) CHANGE LETTERS TO CODED EQUIVALENT
700 FOR X=1 TO L
800 A$=ASC(MID$(B$,X,1))-64
```

```
900 IF AS<1 OR AS>26 THEN MID$(A$,X,1)=MID$(B$,X,1): GOTO 1100'IF NOT A
LETTER LET IT ALONE
1000 MID$(A$,X,1)=CHR$(C(AS)+64)
1100 NEXT X
1200 'DECODE BY PLAYER
1250 C$=A$
1300 CLS: PRINT@32*2,A$: FOR X=1 TO 6: FL(X)=0: NEXT X
1400 PRINT@32*13,"CHANGE FROM";: INPUT F$: IF F$="XX" THEN A$=C$: GOTO
1200 ELSE: INPUT"TO";T$: IF T$=F$ THEN 1400
1500 FOR X=1 TO 26: IF FL(X)=1 AND ASC(F$)-64-X THE SOUND 150,10: GOTO 1400
ELSE NEXT X 'IF YOU ALREADY CHANGED THE LETTER THEN IGNORE
1600 FL(ASC(F$)-64)=1 'FLAG TO SHOW LETTER WAS CHANGED
1699 '(1700-1900) CHANGE ALL SELECTED LETTERS TO NEW VALUE
1700 FOR X=1 TO LEN(A$)
1800 IF MID$(A$,X,1)=F$ THEN MID$(A$,X,1)=CHR$(ASC(T$)+32)' +32 MAKES IT A
LOWER CASE LETTER
1900 NEXT X
2000 PRINT@32*8,A$: SOUND 100,1: GOTO 1400
```




```
 1 PRINT@33,"THIS PROGRAM WILL
   BUILD A TREE":PRINT@167,"BY
   THOMAS L. MIX":PRINT@199,
   "3424 COLLEGE N,E,":PRINT@231,
    "GRAND RAPIDS,MICH. 49505"
2 FOR R=1TO460*4
3 NEXT R
5 CLS(5)
10 CLS(0):FORH=0TO63:SET(H,29,5):
   NEXT H
15 FOR V=10TO29:SET(31,V,5):NEXT V
18 H=RND(29):H=H+34
20 FOR V=15TO35 STEP-1:SET(31,V,5)
22 NEXT V
30 H=RND(62):V=RND(20)
40 IF POINT(H,V) THEN 30
50 Q=RND(8):SET(H,V,Q)
60 IF POINT(H,V+1) THEN 30
70 IF POINT(H,V-1) THEN 30
80 IF POINT(H+1,V) THEN 30
85 IF POINT(H-1,V) THEN 30
90 RESET(H,V)
99 GOTO30
```

Videotex
by Gregory H. Cegielski
2029A South 14th Street
Milwaukee, WI 53204

All of the following addresses are shown as Decimal ($HEX).

VIDEOTEX™, from Radio Shack, works this way: It loads with the CLOADM command starting at address 1536 ($0600) with the last location at 3839 ($0EFF); the EXEC pointer is loaded with 1728 ($06C0), and that is where program execution begins. (the EXEC pointer is at location 157 ($009D)).

The first instruction shuts off the IRQ (timer) and FIRQ (cartridge) interrupts; nothing can interfere with VIDEOTEX. A branch is then made around the body of VIDEOTEX, and a routine is entered which rewrites VIDEOTEX at 268 ($010C) and continues for 2080 ($0820) bytes, ending just short of said rewriting routine, at 2348 ($092C). The effective entry address into VIDEOTEX, during the rewriting, lands at 609 ($0261). After the rewriting is complete processor control is transfered to 609 ($0261), which is a NOP ($12), and the actual VIDEOTEX begins. The RESET FLAG, located at 113 ($0071) is then loaded with a HEX ($55), and the RESET VECTOR at 114 ($0072) and 115 ($0073) is loaded with 609 ($0261). For this last reason, pressing the RESET button causes the computer to jump back to 609 ($0261) and reenter VIDEOTEX, until power to the computer is shut off. This can be changed with a poke to address 2103 ($0837) with any value other than 85 ($0055). Without the HEX ($55) reset flag the computer will reenter BASIC; however this will reconfigure about the first half of VIDEOTEX, and you won't be able to use VIDEOTEX again without reloading.

To get around this, I used the same rewriting sequence to first duplicate VIDEOTEX at 30208 ($7600) to 32319 ($7E3F), and have that copy then make a second copy to configure the area below as the original would have. (I have 32K, although this could also be done with any size RAM). At machine speed, there is, of course, no detectable time difference going through an extra copy.

The actual code to do this is as follows:

ADDRESS	MACH. CODE	WHAT HAPPENS
1723 ($06BB)	1A 50	Shuts off IRQ & FIRQ
1725 ($06BD)	16 08 40	Branch to end of VIDEOTEX plus 1
1728 ($06C0) to 3839 ($0EFF)	NORMAL VIDEOTEX - bring in with CLOADM	
3840 ($0F00)	30 8D F7 BC	Puts address of start into X-register
3844 ($0F04)	10 8E 08 3F	Puts length into Y-register
3848 ($0F08)	CE 76 00	New location into U-register
3851 ($0F0B)	A7 C0	Move A's contents into U's address, increase U by 1
3855 ($0F0F)	31 3F	Decrease Y by 1
3857 ($0F13)	7E 76 00	Jump to the copy you just made
3862 ($0F16)	00	To keep BASIC happy

Actually, using just PEEK and POKE you could move your VIDEOTEX up to 30208 ($7600); there were two other factors in my choice of using this routine.

First, because the normal screen is located at 1024 ($0400) to 1535 ($05FF) I wanted to include just before any program coming in from tape a screen of instructions and phone numbers of various CBBS's - - after all, why just look at a blank green screen while waiting for the program to load, may as well have some instructions and the bright colors sure impress visitors.

Using the Extended BASIC CSAVEM command to:

CSAVEM "VIDEOTEX",&H0400,&H0F16,&H06BB

will also write to tape, and reload with CLOADM, whatever is on the screen. Since CSAVEM can be used as a program line, all that is necessary is to first have your program fill in the screen as you desire, and as a near final step save the screen contents and your VIDEOTEX which closely follows. Had you put VIDEOTEX way up in memory to start there could be a lot of extra loading inbetween. When using CSAVEM you can also put a GOTO to loop back to CSAVEM and make enough copies so that you don't have to always rewind. Put a timer, i.e. FOR ZZ=0 TO 500: NEXT between the steps, and you'll have a slight pause between your copies.

The other reason is that there may be times when it may be desirable to offset load your VIDEOTEX (if some memory contained a program that should not be altered, the Micro Works CBUG tape monitor at 1536 ($0600) for example) (And let me add that everybody, everybody, who is serious about their Color Computer should have the Micro Works ROM CBUG MONITOR, which has allowed me more insight into the machine than anything else).

Since I have 32K and VIDEOTEX only stores 26 pages, I ran the Micro Works disassembler on it and found out that what it does is check for the hard wire jumper on PIA $FF22 to determine if the machine is 4K or 16K.

If the jumper is on 16K it stores in register B the value 26 ($1A), otherwise it stores just 2. If you have 32K, you can increase the "off line" storage to 58 pages by a POKE 2112,58:EXEC; although you will want to make this change permanent if you decide to go with the reprogramming of VIDEOTEX already mentioned. In that case you will have to limit your "offline" storage to 53 pages, because the copy of VIDEOTEX you will make up at 30207 ($7600) will need some room, and BASIC's stack can't be allowed to fall into your VIDEOTEX copy if and when you reset to BASIC.

For 32K and Extended BASIC, here is a short program which will at least give you a working copy and 53 pages of storage, but will not fill your screen automatically from tape:

```
10 A=30208
20 POKE 2103,255 REM PERMITS RESET
30 POKE 2112,53  REM STORES 53 PAGES
40 FOR X=1728 TO 3839
50 POKE A, PEEK(X)
55 PRINT CHR$(PEEK(X)); REM (optional)
60 A=A+1: NEXT X
70 CSAVEM "ANY NAME",30208,32319,30208
80 FOR Z=1 TO 500: NEXT Z
90 GOTO 70
```

You few guys with 64K can store 122 pages; POKE 2112,122; if you've got the memory it'll goto 250.

We all should write personal letters of appreciation to the capable engineers at Motorola who gave us this machine.

```
10 'RACECAR' BY PEGGY SCHUBERT
20 'RS COLOR COMPUTER, 4K
30 CLS
40 PRINT"RACECAR
          BY PEGGY SCHUBERT
50 PRINT"PRESS LEFT ARROW TO MOV
E LEFT
60 PRINT"PRESS RIGHT ARROW TO MO
VE RIGHT
70 PRINT"PRESS <ENTER> TO START
80 INPUT X
90 CLS
100 C=4
110 D=RND(3)-2:L=RND(12)
120 FOR Y=1 TO L
130 R=R+D
140 IF R>22THEN R=22:D=D-2
150 IF R<1THEN R=1:D=D+2
160 PRINT@480+R,CHR$(149);CHR$(2
07);CHR$(207);CHR$(207);CHR$(207
```

continued on page 44

COMMENT CORNER

by Andrew Phelps
The Micro Works

The following is a list of comments which could be added to a disassembly listing of the Color Computer ROM. Two sections are given here. The first is the serial output driver in the Color BASIC ROM which is used to send a character to the printer. The second is a serial Input/Output driver from Extended BASIC which allows use of the RS232 port in both directions.

These routines may be called from Assembly Language programs in order to use the Color Computer with a variety of serial peripherals or communication links. New routines for special applications may be written using these as models.

Variables, areas, and routines -

Addr	Comments
0095	BAUD RATE
0097	PRINTER RETURN DELAY
009B	PRINTER WIDTH
009C	PRINT HEAD POSITION
A2BF	START OF PRINTER OUT
A2FB	SEND MARK BIT
A302	DELAY HALF BIT TIME
A7D3	DELAY ROUTINE

Line-by-line comments -

Addr	Comments
A2BF	SAVE CCR,A,B,X
A2C1	INHIBIT INTERRUPTS
A2C3	SEND A STOP BIT
A2C5	REMOVE MSB, ADD START BIT
A2C6	INITIALIZE BIT COUNTER
A2C8	START LOOP; SAVE COUNTER
A2CA	CLEAR B
A2CB	GET BIT TO SEND
A2CC	MOVE BIT INTO B
A2CD	MOVE TO BIT 1 IN B
A2CE	OUTPUT TO $FF20
A2D1	DELAY HALF BIT TIME
A2D3	3 NOPS; PATCH IN CODE
A2D6	OTHER HALF BIT DELAY
A2D8	RESTORE COUNTER
A2DA	COUNT DOWN BITS
A2DB	LOOP FOR EACH BIT
A2DD	SEND STOP BIT
A2DF	RESTORE INTERRUPTS & A
A2E1	WAS IT A CARRIAGE RETURN?
A2E3	IF IT WAS, GO DELAY
A2E5	INCREMENT HEAD POSITION
A2E7	GET HEAD POSITION
A2E9	SAME AS PRINTER WIDTH?
A2EB	IF LOWER, DON'T DELAY
A2ED	RESET HEAD POSITION
A2EF	DELAY FOR C/R
A2F1	DELAY SOME MORE
A2F3	GET RS232 INPUT
A2F6	MOVE INPUT BIT TO CARRY
A2F7	LOOP TIL PRINTER NOT BUSY
A2F9	RESTORE B,X; RETURN
A2FB	GET A "2" (OUTPUT HIGH)
A2FD	STORE TO OUTPUT
A300	CALL DELAY HALF BIT
A302	GET BAUD RATE CONSTANT
A304	SKIP TWO BYTES
A305	GET C/R DELAY CONSTANT
A307	JUMP TO DELAY ROUTINE
A7D3	DECREMENT X REGISTER
A7D5	LOOP TIL ZERO
A7D7	RETURN

EXTENDED BASIC SERIAL I/O

Variables, Areas, and Routines -

Addr	Comments
00E6	BAUD RATE CONSTANT
00E7	INPUT TIMOUT CONSTANT
008A	ALWAYS CONTAINS ZERO
8DBC	INPUT RS232 CHARACTER
8DE6	GET BIT OR TIME OUT
8DF7	DELAY ONE BIT TIME
8E0C	SEND RS232 CHARACTER

Line-by-line Comments -

Addr	Comments
8DBC	SET CONDITION CODE "ZERO"
8DBD	SAVE CCR, B, X
8DBF	INHIBIT INTERRUPTS
8DC1	GET TIMOUT CONSTANT
8DC3	GET A ZERO TO X
8DC5	GET A BIT
8DC7	WAIT UNTIL STOP BIT
8DC9	GET A BIT
8DCB	WAIT UNTIL START BIT
8DCD	WAIT HALF BIT TIME
8DCF	INITIALIZE BIT MASK (=1)
8DD1	SAVE MASK ON STACK
8DD3	CLEAR INPUT BYTE
8DD4	WAIT BIT TIME
8DD6	READ INPUT PORT

```
8DD9  MOVE INPUT BIT TO CARRY
8DDA  BRANCH IF INPUT BIT ZERO
8DDC  OR BIT MASK INTO A
8DDE  SHIFT BIT MASK
8DE0  LOOP FOR NEXT BIT
8DE2  BUMP MASK OFF STACK
8DE4  RESTORE & RETURN
8DE6  READ INPUT PORT
8DE9  MOVE INPUT BIT INTO CARRY
8DEA  INCREMENT X
8DEC  IF NONZERO, RETURN OK
8DEE  COUNT DOWN A
8DEF  IF NONZERO, RETURN OK
8DF1  TIMOUT; REMOVE RET ADDR
8DF3  RESTORE REGISTERS
8DF5  MAKE CONDITION CODES <>0
8DF6  RETURN TO CALLING PROGRAM
8DF7  CALL FOR HALF BIT DELAY
8DF9  SAVE A REG
8DFB  GET BAUD RATE CONSTANT
8DFD  DELAY THREE CYCLES
8DFF  COUNT DOWN
8E00  LOOP FOR DELAY
8E02  RESTORE A AND RETURN
8E0C  SAVE REGISTERS
8E0E  INHIBIT INTERRUPTS
8E10  DELAY 1 BIT (STOP BIT)
8E12  DELAY (2ND STOP BIT)
8E14  SEND A ZERO FOR START BIT
8E17  DELAY FOR START BIT
8E19  START WITH BIT ZERO
8E1B  PUT BIT MASK ON STACK
8E1D  GET DATA BYTE
8E1F  MASK FOR CORRECT BIT
8E21  IF ZERO, GO SEND ZERO
8E23  IF NOT ZERO USE A "2"
8E25  SEND THE BIT
8E28  DELAY FOR ONE BIT TIME
8E2A  MOVE BIT MASK OVER
8E2C  IF MORE BITS, LOOP
8E2E  USE A "2" FOR STOP BITS
8E30  START THE STOP BIT
8E33  CLEAN MASK OFF STACK
8E35  RESTORE AND RETURN
```

QUESTION: What is RS232?

RS232 is a method of sending bytes of data along a single wire. It is used for talking to printers, modems, and CRT terminals.

How can data be sent with just one signal wire?

RS232 data is sent serially; that is, one bit at a time. Bit zero is sent first, then bit one, and so on. Each bit is sent for a predetermined time.

How do you send a bit?

In the Color Computer, a bit is sent by writing it to bit one of location $FF20. Writing $02 to $FF20 will send a "one" by putting -12 volts on the RS232 output line. Writing $00 to $FF20 will send a "zero" by putting +12 volts on the RS232 output line.

How long is the output left at the value for each bit?

That depends on the baud rate. The standard rate for the Color Computer is 600 baud, or 600 bits per second. Therefore there is a programmed delay of 1/600th of a second after each bit is shifted out.

How can a receiving device know when the first bit starts?

When no data is being sent, a "one" value is left on the RS232 line. Before each character is sent, a "zero" is sent for one bit time. This is called a start bit and warns the receiving device that data bits follow. After all data bits are sent, a "one" is sent for at least one bit time to allow the receiving device to process the character before the next start bit is sent.

Does the stop bit always come after the data bits?
The stop bit comes in between bytes, so it doesn't matter if it is thought of as coming before or after the other bits. The Radio Shack printer routine puts one stop bit before and another after the other bits.

How do I receive a bit?

The RS232 input line is connected to bit 0 of location $FF22. LDA $FF22 / ANDA #1 will put the zero or one into the A register.

Then how do I receive a byte?

First, wait for the input to go to a zero. This is the start bit. Now, wait for 1 1/2 bit time so that you can sample in the middle of each bit. Then, do the following eight times: Shift in the next bit, then wait one bit time. When all bits are in, you are done.

How many bits are sent?

Although eight bits can be sent, ASCII characters only need seven bits. The Radio Shack printer routine only sends seven bits, but it sends two stop bits so that the byte is long enough even if the the printer expects eight bits. The output routine at $8E0C sends eight bits.

How is the RS232 port usually hooked up in the Color Computer?

The serial out line is normally assumed to be hooked to a printer. The serial in line is not used for serial in (since printers don't send anything anyway) but rather for a printer busy line: one means busy, zero means ready. The remaining line is called carrier detect and is not used for printers.

What is carrier detect for?

Carrier detect is used by modems to tell the computer that a carrier tone is present. It is edge-triggered; that is, it can can be used to see if the line has gone from one to zero, or from zero to one. It can be used to generate an interrupt.

What are the "Printer width" and "Head position" variables in Basic's printer routine?

The only purpose of these is to provide a short time delay whenever the computer thinks a carriage return is being done. This may make it unnecessary to use the printer busy input.

If I am hooked up to a two-way device like a modem, can I send and receive at the same time?

Sure. A program like that is just a little harder to write.

So what is RS232, anyway?

It's a way of communicating, not just to a printer but to mainframes and other computers, to time bases or appliance controllers, to plotters or local networks. It's your computer's link to the outside world.

CLOAD "CHEKBOOK"

by Richard White

Why write a program that can do what a calculator can do? The best answer may be that it is a way to develope programming skills and reading this and keying in the program won't hurt either. Programs develope in funny ways. This one started to grow in my mind when I couldn't balance the checkbook against the bank statement. It turned out that my wife had pulled out one check to get a reimbursement before I got around to checking it off in the checkbook. The program helped me discover that. The program runs in 16K of memory and requires Extended Color Basic. It allows you to enter deposits, checks, adjustments including void checks, deletions, additions and service charges. Other features are; review and edit all entries, save cassette files, mark tax deductable checks and add notes such as payee, what the check was for and the like. At tax time next year, I will simply load in the monthly files from the cassette and review the entries listing the deductions as I go. The whole year will be on a single cassette.

When you first start, you enter the previous balance and first check number if you have not entered a cassette file. After that, the program will automatically present the next check number in order unless you type a different one in. It will automatically reuse the date and amount of the last check if you do not enter these. You may omit the decimal and zeros if an even dollar amount. The program reprints the amount, formatted with the PRINT USING command, and the new balance. Type a T in the last column to indicate a tax deductable item. Finally you are given the note option. A menu of choices and their letter codes is included in the program and can be recalled by typing M <ENTER> at the check entry point of the entry sequence. The menu choices are selected at the same time by typing their key <ENTER>.

Type R <ENTER> to get into the review and edit mode. When the program asks what check number to start with enter one that is before the one you want that you are sure is in the file. The program displays each number as it searches backwards for the target number. You may now review the file pressing <ENTER> for each new entry, or E for Edit, A for Add or D for Delete. In Edit the <ENTER> key is used to step across the line allowing you to enter new data only where needed. A inserts a new entry and D deletes an entire entry. The program will recalculate all the new balances with each change and presents the next entry for review.

Typing F <ENTER> will get you into the File mode. When outputting to tape you are asked for the first and last months in the file and the year, it will then make up a file name, prints it, writes two copies to tape and offers the option to make a backup on a second tape.

The other menu choices are "D" for deposit, "A" for adjustment and "V" for voided check. For a deposit, "DEP" is printed instead of a check number and the amount is added to the last balance rather than subtracted. The adjustment allows for all other transactions like service charges, check charges and perhaps interest in a NOW account. The void check entry permits keeping the check number on file, but automatically enters a $0.00 amount.

I thought I would really get organized before starting to type the program in. I generated a flow chart, defined subroutines, blocked out sets of line numbers for each and put a lot of programs on paper before entering it into the computer. Table 1 outlines the program structure as it finally evolved. The approach worked well up to the edit and review section which grew out of the alloted space and had to be continued above the cassette subroutines. Lines 1-99 are allocated to start-up functions and frequently used general purpose subroutines.

A word about the general utility subroutines. These include such things as the INKEY$ sub, the timer, centering, cassette positioning and others. When I start a program I load a tape with these subroutines and go from there. They become a part of each program and always have the same line numbers, low ones, that are easily remembered and save a byte of two of memory versus numbers over 100 or 1000 each time used.

The menu starts at line 100, a key number to remember. If you get an error or break the program and want to pick up where you left off type GOTO 100and the program will resume without losing any variables. RUN would initialize the program

losing all data previously entered. Data entry subroutines run from line 200 to 370. Since I have tried to use subroutines as much as possible, some lines are a collection of GOSUBs. This makes the program harder to follow but sure saves memory.

I find formatting the screen, particularly when data lines are to be scrolled while maintaining a heading one of programming's harshest punishments. The screen formatting subroutines in lines 410 to 440 are worth some study for ideas you can incorporate into your own programs. Line 410 starts with three PRINT commands that scroll whatever is on the screen up three lines, leaving three blank lines at the bottom. PRINT@0 is then used to print the column headings at the top of the screen. Another PRINT command clears the line under the headings. Then a PRINT@383,"" command positions the cursor at the start of the third line up from the bottom for entry of a check number or menu selection.

Depending of what the next transaction is, CK$ in line 415 may be check number, DEP for deposit or ADJ for adjustment. This is printed with a few blank spaces to assure there is nothing else in the space. PRINT@390,"" moves the cursor for the start of date entry at location 391.

The PRINT@396,"" in line 420 positions the cursor for amount entry. The amount is entered as you would into a calculator with the numerals printing from left to right. Obvious the right end will vary with the size of the number. PRINT USING corrects this, reprinting the entry right justified with two figures after the decimal. If you have entered an even dollar amount without the decimal point and two following zeros, PRINT USING supplies these as well as the dollar sign. The amount itself is contained in a string variable QN$ for filing and retrieval purposes and must be converted to a number using VAL(QN$).

After I had written and debugged the program, I felt that these screen formatting subroutines provided a fairly economical and effective solution for an effective screen display. Now I see some things that are either unnecessary or could be done differently to save memory and to make the program easier to follow. The blank spaces after the string variables are probably unnecessary. More wasteful is line 440 which could have been included in line 435, saving GOSUB statements elsewhere in the program. Debugging a program has two parts, getting the program to work the way you want and then cleaning up the garbage. Lines 415 and 416 are an example. They are so similar that one is unnecessary. 416 was added to solve some problem during debugging, but there must be a better solution.

Another form of program garbage are lines that are not used at all. These arise when you solve a problem by writing some new lines elsewhere and sending the program to them while forgetting to remove the old lines. Not only do these waste memory, they cause confusion when you come back later to improve the program. Try to reserve some time at the end of a debugging session to clean things up.

String variables provide an efficient way to store, file and retrieve both numeric and alphabetic information. Remember that check number, date, amount and balance were in string variables rather than in numeric variables. We see why in line 630 where these variables are simply added with $ separators to form a single variable SN$(K) that holds all the information for a single entry. The variables CK$, DA$, BA$, QN$, TD$ and PY$ are available to be reused for the next entry. There is considerable memory savings versus putting the data into a multiple variable array. The cost is the memory used for the lines to make the string SN$(k) and to later break it up. In this case, I estimate the savings to appear when there are more that 10-15 entries.

Line 650 to 695 are the string disassembly subroutine, the IF - THEN statement in line 650 catches a string containing no data before an FC error occurs in line 675. This was added during debugging and there probably is a better solution. In line 675 the program finds the location in the string of each of the $ data separators. Lines 680 and 690 use LEFT$ and MID$ statements to put the data into it's respective string variables for display. The first time through string disassembly tries one's patience. Once you get the hang of it you will find it rather elegant and satisfying.

At this point in our adventure, we have successfully waded the marsh of string disassembly and find ourselves facing the odorous swamp of review and edit. The primary problem was to get the balance changes in subsequent strings right when

there was an amount changed added or deleted. Use the Program Structure Summary in Table 1 to help guide you through a detailed study.

There are a number of features of the cassette file saving that deserve comment. Lines 870 and 880 input the start month, SM end month EM and the year for the file. In line 880 a file name is made from these entries which line 885 displays. The variable JK is set to 0 here also. In line 890 the file is opened, KJ is set to 0, JK is set to 1 indictating this is the first file save. In line 900, the program starts looking at the date of each SN$(KJ) until it finds the first one for the start month. When the desired SN(KJ) is found, the program goes to 930 to print it to the cassette. The program returns to 900 to get the next SN$(KJ). It tests the date to see that it is not greater than the end month and checks that the last entry with date has not been printed before printing the entry to cassette. If either of the above conditions are met the program goes to 910 where the file is closed and JK is checked to see if this was the first save. If so then a FOR TO NEXT loop is used to time about two seconds of blank tape between saves and the program returns to 890 to do it again. If JK=2, the option to record the file on a backup tape is offered in line 920.

It is always good practice to save a file twice and critical data should be on a backup tape. Borrow the techniques in 885 to 940 to do this in your programs. The file input routine, lines 950-980, are usual using "N"as a count variable and EOF(-1) to find the end of the file.

The last item I want to mention is the PCLEAR 1 in line 10000. From line 1 the program goes to the very last line for PCLEAR 1 and then returns to line 10. This avoids an SN ERROR after the first RUN command after loading and was discussed in a previous COLOR COMPUTER NEWS. It does not avoid an FC ERROR of the computer has not been turned off after running a machine language program and before loading CHECKBK.

TABLE 1

LINES	FUNCTION
1-10	Clear and dimensions
11-22	Utility Subroutines
90-95	File Input Choice & Initial Balance
100-180	Menu & Choice Selection
200-240	Check Input Routine
250-270	Deposit Entry Routine
300-310	Adjustment Routine
350-370	Void check routine
400-440	Screen Format Routines
530-550	Date Subroutine
560-580	Amount Subroutine
590-610	Balance Subroutine
620-630	String Assembly Subroutine
650-695	String Disassembly Subroutine
700	Start Review & Edit Routine
705-715	Find starting Check
720-740	Review Entries Loop
750-770	Edit, Addor Delete Selection
770-795	Start Delete Routine- Continues 1060
800-817	Add line Routine- Continues at 1060
820-830	Edit Mode Balance Adjustment- Continues at 1000
850-860	File Input or Output Selection
870-940	File Output Routine
950-980	File Input Routine
1000-1050	Edit Routine Concluded
1060-1090	String Balance Correction
10000	PCLEAR 1

```
1 GOTO10000
10 CLEAR5000:DIMSN$(100):GOTO90
11 Z=(32-LEN(ZT$))/2:PRINTTAB(Z)
   ZT$:RETURN
17 IFZS*60)TIMER THEN 17 ELSE RE
   TURN
18 PRINT"****TO PROCEED TOUCH AN
   Y KEY****"
19 Z$=INKEY$:IFZ$()""THEN RETURN
   ELSE19
20 PRINT"TO SET TAPE RECORDER AN
   D POSITION TAPE TO SAVE OR LOAD,
   PRESS ANY KEY FOR MOTORON ON AN
  DTHEN ANY KEY FOR MOTOROFF"
21 Z1$=Z$:GOSUB19
22 AUDIOON:MOTORON:GOSUB19:MOTOR
   OFF:Z$=Z1$:RETURN
30 CLS:ZT$="MENU":GOSUB11:ZT$="P
   RESS (KEY) INDICATED":GOSUB11:PR
  INT:PRINT"(1ST DIGIT OF CHECK#)
  THEN REMAINING DIGITS AFTER '?'
  TO START NEW CHECK SEQUENCE":PRI
  NT"(ENTER) FOR NEXT CHECK# IN SE
  QUENCE AUTOMATICALLY"
35 PRINT"(D) FOR DEPOSIT":PRINT"
   (A) FOR ADJUSTMENT":PRINT"(V) FO
  R VOIDED CHECK":PRINT"(R) FOR RE
  VIEW AND EDIT":PRINT"(F) FOR LOA
  D OR SAVE FILE":PRINT"(M) FOR ME
  NU";:RETURN
```

```
90 CLS:PRINT:PRINT"PRESS (F) IF
   TAPE FILE ELSE ANY KEY":GOSUB19:
  IFZ$="F" THEN Z$="I":GOSUB20:PRI
  NT:PRINT"SET RECORDER TO PLAY &
  PRESS ANYKEY":GOSUB19:GOTO950
95 GOSUB410:PRINT@352,"ENTER INI
   TIAL DATE AND BALANCE":CK$="BAL"
  :GOSUB415:GOSUB530:PRINT@404,"";
  :LINEINPUTBA$:GOSUB430:GOSUB435:
  GOSUB440:N=N+1:GOSUB620:GOSUB30:
  GOSUB19:CK=VAL(Z$):CLS:BA=VAL(BA
  $):CLS:GOSUB410:GOTO120
100 POKE65494,0:GOSUB30:GOSUB19:
   CLS:FG=0:CK=VAL(Z$):GOSUB410:GOT
  O120
110 IFE=1 THEN 720
115 GOSUB410:LINEINPUTZ$:CK=VAL(
   Z$)
120 IFZ$="" THEN 200ELSEZ=ASC(Z$
   ):IFZ=13 THEN 200
125 CK=VAL(Z$):IFCK)0 THEN 200
130 IFZ$="D" THEN 250
140 IFZ$="A" THEN 300
150 IFZ$="V" THEN 350
160 IFZ$="R" THEN 700
170 IFZ$="M" THEN 100
180 IFZ$="F" THEN 850ELSE110
200 'ENTER CHECK
205 IFCK=0 THEN 240ELSECK$=Z$:PR
   INT@384,Z$;
210 LINEINPUTZ$:CK$=CK$+Z$:CK=VA
   L(CK$)
215 CH$=CK$:GOSUB415
220 GOSUB530:GOSUB560:GOSUB590:L
   INEINPUTZ$:IFZ$="T" THEN TD$=Z$E
  LSETD$=""
230 GOSUB435:GOSUB440:N=N+1:K=N:
   GOSUB620:GOTO110
240 CK$=STR$(VAL(CH$)+1):CH$=CK$
   :CK=VAL(CK$):GOSUB416:GOTO220
250 'ENTER DEPOSIT
260 CK$="DEP":GOSUB415:GOSUB530:
   GOSUB560:GOSUB270:GOSUB590:GOSUB
  270:TD$="   ":GOTO230
270 QN$=STR$(-VAL(QN$)):RETURN
300 'ENTER ADJUSTMENT
310 CK$="ADJ":GOSUB415:GOSUB530:
   GOSUB560:GOSUB270:GOSUB590:GOSUB
  270:TD$="   ":GOTO230
350 'VOID CHECK
360 LINEINPUTZ$:IFZ$="" THEN CK$
   =STR$(VAL(CH$)+1)ELSECK$=Z$
370 GOSUB415:GOSUB530:QN$="VOID"
   :GOSUB425:TD$="   ":CH$=CK$:GOTO2
  30
400 'SCREEN FORMAT SUBS
410 PRINT:PRINT:PRINT:PRINT@0,"C
   HECK# DATE AMOUNT  BALANCE TAXD"
  :PRINT:PRINT@383,"":RETURN
415 PRINT@384,CK$+"  ":PRINT@390
   ,"";:RETURN
416 PRINT@383,CK$+"  ":PRINT@390
   ,"";:RETURN
420 PRINT@390,DA$+"  ":PRINT@396
   ,"";:RETURN
425 PRINT@395,"";:PRINTUSING"$$#
   ###.##";VAL(QN$):PRINT@405,"";:R
  ETURN
430 PRINT@404,"";:PRINTUSING"$$#
   ###.##";VAL(BA$):PRINT@414,"";:R
  ETURN
435 PRINT@414,TD$+"  ":PRINT@416
   ,"NOTE: ";:RETURN
440 PY$="":LINEINPUTPY$:RETURN
530 'DATE SUB
540 LINEINPUTZ$:IFZ$="" THEN GOS
   UB420:RETURN
550 DA$=Z$:GOSUB420:RETURN
560 'AMT SUB
570 LINEINPUTZ$:IFZ$="" THEN GOS
   UB425:RETURN
580 QN$=Z$:GOSUB425:RETURN
590 'BALANCE SUB
600 BA=BA-VAL(QN$)
610 BA$=STR$(BA):GOSUB430:RETURN
620 'STRING ASSEMBLY
630 SN$(K)=CK$+"$"+DA$+"$"+QN$+"
   $"+BA$+"$"+TD$+"$"+PY$:RETURN
650 IFSN$(K)="" THEN RETURN:'STR
   ING DISASSEMBLY*********
660 LS=LEN(SN$(K)):K1=0:L(0)=0:P
   Y$=""
675 K1=K1+1:L(K1)=INSTR(L(K1-1)+
   1,SN$(K),"$"):IFL(K1))0 THEN 675

680 CK$=LEFT$(SN$(K),L(1)-1):DA$
   =MID$(SN$(K),L(1)+1,L(2)-L(1)-1)
  :QN$=MID$(SN$(K),L(2)+1,L(3)-L(2
  )-1)
690 BA$=MID$(SN$(K),L(3)+1,L(4)-
   L(3)-1):TD$=MID$(SN$(K),L(4)+1,L
  (5)-L(4)-1):IFLS)L(5) THEN PY$=M
  ID$(SN$(K),L(5)+1,LS-L(5))ELSERE
  TURN
695 RETURN
700 'EDIT SUB
705 CLS:INPUT"STARTING CHECK #";
   C8:K=N:E=1:IFC8=0 THEN705:POKE65
  495,0
710 K=K-1:IFK=0 THEN PRINT"CHECK
    # "C8" NOT IN FILE":E=0:POKE654
  94,0:GOTO100
715 GOSUB650:GOSUB415:CK=VAL(CK$
   ):IFC8()CK THEN 710
720 GOSUB410:PRINT@32," PRESS (E
   ) TO EDIT-":PRINT@64,"  PRESS (A
  ) TO ADD":PRINT@96,"   PRESS (D)
   TO DELETE":PRINT"ANY OTHER KEY
  TO PROCEED":PRINT
722 GOSUB415:GOSUB420:GOSUB425:G
   OSUB430:GOSUB435:PRINTPY$:POKE65
  494,0:GOSUB19:IFZ$="E"ORZ$="A"OR
  Z$="D" THEN 750
```

```
725 IFVAL(CK$))0 THEN CH$=CK$
730 IFK=N THEN PRINT"THIS IS LAS
   T ENTRY IN FILE":BA=VAL(BA$):GOS
  UB18:E=0:GOTO100
740 K=K+1:GOSUB650:GOTO720
750 GOSUB410:CK=VAL(Z$):IFCK)00R
   Z$="E" THEN 820
760 POKE65495,0:IFZ$="A" THEN 800
770 IFZ$()"D" THEN PRINT"INVALID
    ENTRY":GOTO720
780 JK=K:N=N-1:K=K-1:GOSUB650:BA
   =VAL(BA$):QN$=""
790 SN$(JK)=SN$(JK+1):JK=JK+1:IF
   JK(N THEN 790ELSEJ=K:GOTO1060
795 K=K-1:J=K:GOTO1060
800 N=N+1:K=K+1:JK=N:BA=VAL(BA$)
   :CK=VAL(CK$)
810 SN$(JK)=SN$(JK-1):JK=JK-1:IF
   JK)K THEN 810ELSEGOSUB410:POKE65
  494,0:LINEINPUTCK$:GOSUB415:GOSU
  B530:GOSUB560:CK=VAL(CK$):IFCK)0
   THEN BA=(BA-VAL(QN$))ELSEBA=(BA
  +VAL(QN$))
815 BA$=STR$(BA):GOSUB430:LINEIN
   PUTZ$:IFZ$="T" THEN TD$="T"
817 GOSUB435:LINEINPUTPY$:GOSUB6
   20:GOSUB660:BA=VAL(BA$):J=K:GOTO
  1060
820 IFVAL(CK$))0 THEN BA=(VAL(BA
   $)+VAL(QN$)):GOTO1020
830 BA=VAL(BA$)-VAL(QN$):GOTO1020
850 'FILE SUB
855 CLS:PRINT:PRINT"KEY IN (I) T
   O INPUT FROM TAPE          (O) TO
   OUTPUT TO TAPE":GOSUB19:GOSUB20
  :CLS:PRINT:IFZ$="I" THEN PRINT"S
  ET RECORDER TO PLAY & PRESS ANYK
  EY":GOSUB19:GOTO950
860 IFZ$="O" THEN PRINT"SET RECO
   RDER TO RECORD & PRESS  ANY KEY"
  :GOSUB19
870 CLS:PRINT:PRINT"ENTER START-
   MONTH AND END-MONTH":PRINT:PRINT
  "ALL TRANSACTIONS IN THESE MONTH
  SWILL BE RECORDED"
880 INPUT"START-MONTH NUMBER ";S
   M$:INPUT"END-MONTH NUMBER ";EM$:
  INPUT"YEAR ";YR$:IFEM$=SM$ THEN
  NF$=SM$+"/"+YR$ELSENF$=SM$+"-"+E
  M$+"/"+YR$
885 PRINT:PRINT:PRINT"PLEASE WRI
   TE DOWN FILE NAME":PRINT:PRINT"
      "NF$:GOSUB18:EM=VAL(EM$):SM=V
  AL(SM$):JK=0
890 OPEN"O",-1,NF$:KJ=0:JK=JK+1
900 KJ=KJ+1:GOSUB650:DA=VAL(LEFT
   $(DA$,2)):IFDA(SM THEN 900
904 IFKJ)N THEN 910
906 IFDA)EM THEN 910ELSE930
910 CLOSE-1:IFJK=1 THEN MOTORON:
   FORKJ=1TO1000:NEXT:GOTO890
920 IFJK=2 THEN CLS:CLOSE-1:PRIN
   T"PRESS (Y) TO SAVE FILE TO BACK
  UPTAPE":GOSUB19:JK=0:IFZ$="Y" TH
  EN 940ELSE100
930 PRINT#-1,SN$(KJ):GOTO900
940 GOSUB20:CLS:PRINT"SET RECORD
   ER TO RECORD & PRESS  ANY KEY":G
  OSUB19:GOTO890
950 PRINT:INPUT"ENTER FILE NAME"
    ;NF$:OPEN"I",-1,NF$:N=0
960 IFEOF(-1) THEN 980ELSEN=N+1
970 INPUT#-1,SN$(N):GOTO960
980 CLOSE-1:PRINT"PRESS (R) TO R
   EVIEW, ANY OTHER KEY TO ADD ENTR
  IES":GOSUB19:IFZ$="R" THEN CLS:K
  =1:E=1:GOTO720
1000 'ENTRY EDIT**********
1010 IFCK)0 THEN CK$=Z$
1015 CK=VAL(CK$)
1020 GOSUB410:PRINT@33,"IN EDIT
   MODE":GOSUB415:GOSUB530:GOSUB560
  :QN=VAL(QN$):IFVAL(CK$))0 THEN B
  A=(BA-QN)ELSEBA=(BA+QN)
1030 BA$=STR$(BA):GOSUB430:LINEI
   NPUTZ$:IFZ$="T" THEN TD$="T"
1040 GOSUB435:LINEINPUTZ$:IFZ$()
   "" THEN PY$=Z$
1050 GOSUB620:GOSUB660:IFQO$=QN$
    THEN 740
1055 BA=VAL(BA$):J=K
1060 POKE65495,0'STRINGS BALANCE
    CORRECTION**********
1070 K=K+1:GOSUB650:QN=VAL(QN$):
   IFVAL(CK$))0 THEN BA=(BA-QN)ELSE
  BA=(BA+QN):POKE65494,0
1080 BA$=STR$(BA):GOSUB620:IF K(
   N THEN1070
1090 K=J:GOSUB650:GOTO720
10000 PCLEAR1:GOTO10
```

Micro Works Disassembler
by Mark Rothstein

I recently received a copy of the Micro Works disassembler. I was so impressed with it that I decided to write this review article. In this review I'm going to discuss:

What Disasm is,
My impressions of the program,
Reasons why I find it useful and how you may use it,
Some of it's varied output features,
A difficulty I had with DISASM and how I solved it.

What is it?

DISASM, a cassette program for the Color Computer written by Andy Phelps, disassembles computer programs that are in RAM or ROM. It requires 16K of RAM and will output a readable listing of a machine language program to either the TV screen or to a printer.

My first impression.

When I first ran the disassembler, I was really surprised. I had written a disassembler for the 8080 a few years ago and I remembered all the features I had either put in, or wanted to put in. DISASM has them all. It will output a listing similar to the source listing of an assembler ... and with cross references. Or it can output a version that can be reassembled with another program.

This 6809 disassembler is a good finished product. It worked as advertised, the first time I ran it. The version I got came with a good set of instructions with examples of different operating modes. These instructions are clear, but I didn't completely understand them at first because there are so many different output modes. Anticipating this, Micro Works built in a set of defaults. With these defaults, when the program asks a question, an "I don't know" answer is acceptable. In their ads and articles in Color Computer News, Micro Works has been suggesting that Color Computer owners disassemble their copy of the BASIC ROMs. "I don't know" answers to all questions automatically cause the program to disassemble the BASIC ROM at addresses $A000 to $BFFF (where the "$" indicates that the value which follows is in hexadecimal). The disassembly listing scrolls up on the TV display. While the program is running, the scrolling can be adjusted or switched to a single step mode; the listing can be stopped and restarted at any location within the BASIC address range.

Reasons why DISASM is a very useful tool.

Now have you ever wondered how Microsoft (they are the people who wrote Color Computer BASIC) programmed the Color Computer to store your program in BASIC, to read the joysticks, to play music or to paint colors? Well DISASM will HELP you figure out how they've done it. Even if you are a seasoned programmer, this is occasionally a rewarding task -- once in a while you'll learn a new trick. When I find a new project which is related to something already done in BASIC, a question I ask is "How does BASIC do it?" and "Are there any idiosyncrasies in the Color Computer that I'd better take note of?".

A specific case in point is my current project; the design of an inexpensive but reliable bar code reader. This will allow Color Computer owners to load programs directly from Color Computer News. (See the whole series of articles in Byte from November 1976 to May 1978, and the Bar Code Loader from Byte publications.) The easiest place to attach a bar code scanner is at the joystick connector. BASIC can scan the black and white bars, but it does this much too slowly; consequently a machine language program must be devised to do the scanning. What does the BASIC joystick code look like then? The answer is to look at addresses $A9A2 thru $AA19. (Incidentally, the Micro Works documentation includes a list of the addresses of many useful function areas in BASIC.) This time an examination of BASIC didn't turn up anything useful, but at other times, it does.

Micro Works Disassembler

Other Example Uses.

The disassembler can even be made to operate on itself. This isn't necessary, though. A complete source listing of the program comes with the documentation.

There's also a very practical use for DISASM. When I generate a machine language program, in testing it I sometimes find that it doesn't work the first time. As I isolate errors, I fix them with patches -- tacked on sections of code. Then I test the patch and move on to other errors. You're supposed to immediately correct the source code and reassemble. I find this a chore, and practically I only reassemble when I have to fix a serious error or I've got a lot of patches and I'm at the end of a work session. Sometimes accidentally a patch doesn't make it into the next assembled version. This can make the next debugging session quite tedious. But now with DISASM, this doesn't happen anymore. I keep a copy of DISASM co-resident in memory with my machine language program and when I make a patch, I document it quickly and easily -- I disassemble it. This technique has been very effective. It's also helped me find fatal flaws in my hand assembled patches. (More on this next time when I review Micro Works Assembler Editor.)

The many output formats.

Now on to the DISASM output formats which deserve comment. DISASM is designed for both narrow (32 columns like the TV and some printers) and wide (64 columns or more) output devices. In order to output all the important information in the program listing, 64 columns are required. In wide output case, the listing may appear as shown in listing 1.

This is an excerpt from a BASIC disassembly. The first block of lines are external references: Note $C000, the address of the program and game cartridges, the addresses of the PIA's, etc. Next come the addresses of the RAM variables (indicated by the "V" label). After that, if you've read Getting Started with Color Basic, you'll recognize the addresses of some of the BASIC routines (with the typos removed) -- Get keyboard data, Output a character, Start the cassette, etc. See pages 269 and 270 in the Getting Started manual.

Now for the columns: Take for example, te instruction line $A012.

```
A012  8637   --  --  .7  -- LDA #37 "7"
```

The first column is the instruction addresses, then comes the object code (8637), then two blank columns where reference information is listed. Column five has the ASCII equivilent of the operand if it is text and is an immediate value (as represented by the "#" symbol). Now for the columns three and four, examine the BASIC addresses $A01B and $A023:

```
A01B  2651  A06E  .... &Q --- BNE PA06E
A023  2649  A06E  A01B &I --- BNE PA06E
```

Column three has the address referenced by this instruction (if any). (Normally, in assembler output, the label PA06E would have a symbolic name and the address reference would not be obvious.) To explain column four I must say that the 64-column output produces a listing which contains all addresses explicitly referenced together with the address of the highest reference. (This list is sorted numerically.) Column four of the highest address contains the address of the next higher location, and so on. The last reference has a "...." in column four. This is shown at address $A01B.

The narrow format.

If the 32 column display field is used, this output will take up two lines -- this is much less readable. One alternative is to suffer. The other is to omit some of the fields. Five other variations are available, and the user may switch between them "on-the-fly".

Also while the program is running, the user may vary the output speed on the TV display from unreadably fast, to fast to single step. These are conviently selected by the keyboard.

Micro Works Disassembler

My problem with DISASM.

The only problem I had with DISASM concerns the output routine. The ease with which I was able to fix the problem is indicative of the thought that went into DISASM design. Micro Works claims that DISASM is compatable with any printer that will run with BASIC. This is correct -- they use the BASIC printer output routine! My printer, (TI Silenttype) however, is not quite compatable -- it doesn't linefeed automatically after a carriage return. Radio Shack has told me how to patch BASIC, but DISASM requires a different patch. Here are the key ideas in the solution: First DISASM is written in position independent code; located anywhere in memory; it will execute. Not like any of the Radio Shack ROM Paks which are intentionally only a little bit position-dependent. (This is something like being a little bit pregnant.) Second, DISASM make only one direct call to the BASIC printer driver. This is in the beginning of the program where it is easy to find and modify. Therefore a patch in only one place will solve the problem. The solution is shown in Listing 2. DISASM has been relocated to $4020 and the patch resides at locations from $4000 to $401F. The patch also sets The baud rate to 300 Baud. Not that the patch is also position independent.

In Summary.

In summary, I believe that a serious user of the Color Computer will find the Micro Works disassembler to be a useful and efficient tool to have around -- from disassembling BASIC to maintaining documentation of his own programs.

Listing 1. An excerpt from a BASIC Disassembly.

```
                                    NAM   DISASM
                                    ORG   $0000
                               XC608 EQU   $C608
                               XC60E EQU   $C60E
                               XFFC9 EQU   $FFC9
                               XFFFF EQU   $FFFF
                               V0000 RMB   1
                               V0001 RMB   1
A00    A1C1      A1C1 A000   !A      AA000 FDB   PA1C1
002    A282      A282 A002   ".            FDB   PA282
004    A77C      A77C A004   '|            FDB   PA77C
006    A70B      A70B A006   '.            FDB   PA70B
008    A7F4      A7F4 A008   't            FDB   PA7F4
00A    A9DE      A9DE A00A   )^            FDB   PA9DE
00C    A7D8      A7D8 A00C   'X            FDB   PA7D8
```

Listing 2. Patch to print CR with LF.

```
001 0600                X4024   EQU $4024
002 0600                XA2BF   EQU $A2BF
003 0600                V0000   RMB $0096
004 0696                V0096   RMB $3F6E
005 4604                        ORG $4000
006 4000 86B4           P4004   LDA #$B4        SET 300 BAUD
007 4002 9796                   STA <V0096
008 4004 201E                   BRA X4024       JUMP TO DISASM
009 4006 810D           CRLF    CMPA #$0D       CHECK FOR CR
010 4008 2605                   BNE P4013       NO? PRINT & RTS
011 400A BDA2BF                  JSR XA2BF       ELSE PRINT CR
012 400D 860A                   LDA #$0A        THEN PRINT LF
013 400F 7EA2BF         P4013   JMP XA2BF       GO TO BASIC
014 4012                        END   *         DRIVER
```

SIGMON

by Kenneth Kalish
85 Cooper St.
Pringle, PA 18704

If you're interested in writing some graphics subroutines or some short utilities, or if you'd just like to learn about 6809 assembly language, then the most economical answer is available from Datasoft, Inc. under the name of "Sigmon". It is billed as a combination monitor, mini-assembler, and debugger. Sigmon comes with a price tag of only $29.95, and Datasoft doesn't take all year to get it out to you, either. Everything goes in at once, and takes up about 6K or RAM.

Sigmon works, and it works well; but as with everything, there's good news and then there's bad news. The most serious drawback lies in the fact that you don't get a full fledged Editor/Assembler. You do get a mini-assembler, as advertised. That means that there is no source code, therefore no editing and no labels. With Sigmon, you assemble one line at a time, directly to memory. Having no labeling or editing capability can be quite a serious handicap when writing very long programs or routines with complex internal interactions. For example, if you want to insert an instruction in the middle of your assembled code, you can use the block MOVE command to make room for it, but you'll end up throwing off all of your relative addressing. You'll then have to go back and correct each one by hand. It also makes it pretty rough when you're branching ahead in memory to a spot you haven't written yet, You'll usually have to write a guess or a dummy branch, then come back later and fix it with the correct destination (or you'll find yourself structuring your programs to include mostly backward branches). It can all get to be a little aggravating, and a little confusing.

However, since you are programming closer to the ground, so to speak, you are liable to end up learning a lot more about some of the finer details of the language than you would otherwise. You also don't have to wait for assembly runs, which is natural enough since there is no source code to assemble. This is especially helpful when you only want to make one or two quick changes to a program. In addition, you get instant notice of assembly errors; and since there is no source code storage space, you'll have more RAM to work with.

The usual procedure would be to assemble your program to memory, and next use the disassembler to check the code that you use put in. (The disassembler also come in handy in exploring Color Basic or game pak ROMs.) You can then debug the program using break points or the built in single stepper. Sigmon's stepper works by determining the expected address after execution of the current displayed line, and then placing a temporary SWI instruction at the address of that expected destination. It therefore cannot be used through ROM areas.

The single stepper does work beautifully, displaying the register contents as well as the current address, memory contents and disassembled nmemonic for each instruction. However, as a single stepper, it does suffer from a serious malady. That is, after the twentieth time through a simple loop, you begin to feel more like a telegraph operator than a computer programmer, what with having to press a key for each line executed.

Fortunately, a short patch can be applied which provides a good solution. Load Sigmon into it's normal residence and say; "Physician, heal thyself". If that doesn't work, then assemble the following code at address $0FE7 (4071 dec.):

```
$0FE7 LDA 341
$0FEA ORA #8
$0FEC STA 341
$0FEF BRA $2528    Back to Sigmon
```

Then, connect the patch by assembling at $1A11 (6673 dec.);

```
$1A11 BSR $0FE7    To patch
```

and save it to tape with WRITE "SIG",$0FE7,$27F7,$0FF2
That's it. The patch provides a repeat key function for the up arrow key, by suppressing the action of the keyboard buffer for that particular key. (The keyboard buffer is the same one which is used by the keyboard scanning routine in Color Basic. The subroutine is in turn called by Sigmon.) You can now step along by holding down

the up arrow key; and you can vary the step rate by using Sigmon's own "SPEED" command (which imposes a variable delay whenever a carriage return is printed, and is intended to control listing speed).

So there you have it. Sigmon is a no-frills but very functional approach to assembly language programming, with the tradeoffs lying mainly in the advantages of cost and unity, versus a marked lack of ease and convenience in certain programming tasks. You ultimately could write anything with it, such as your own version of Space Invaders, but it isn't very well suited for writing long or complex programs. Still, the final judgement has to be that there is an awful lot of good stuff packed in there for only $29.95.

continued from page 8

```
468 PRINT@202,"DARTH VADER'S"
469 PRINT@234,"TIE FIGHTERS GOT
YOU!!";END
470 GOTO 400
500 SOUND 89,3:SOUND 147,3:SOUND
 147,3
510 FOR D=1 TO 100:NEXT D
520 SOUND 133,3:SOUND 125,3:SOUN
D 108,3
530 SOUND 175,3
540 FOR D=1 TO 100:NEXT D
550 SOUND 147,3:FOR D=1 TO 100:N
EXT D
560 SOUND 147,3:RETURN
800 CLS(0):SET(H,V,0)
805 SOUND 230,1:SOUND 220,1:SOUN
D 200,1:SOUND 180,1
810 FOR D=1 TO 40:NEXT D:CLS(0)
820 SET(H+2,V,0):SET(H-2,V,0):SE
T(H-1,V-1,0)
830 SET(H+1,V-1,0):SET(H-1,V+1,0
):SET(H+1,V+1,0)
840 FOR D=1 TO 40:NEXT D:CLS(0)
850 SET(H-4,V,0):SET(H+4,V,0):SE
T(H-2,V-2,0)
860 SET(H,V-3,0):SET(H+2,V-2,0):
SET(H-2,V+2,0)
870 SET(H,V+3,0):SET(H+2,V+2,0)
880 FOR D=1 TO 40:NEXT D:CLS(0)
890 RETURN
2000 CLS:PRINT@170,"GOOD SHOOTIN
G JEDI!!"
2010 PRINT@200,"THE FORCE WAS WI
TH YOU!
2020 END
```

METEOR

```
10 CLS:INPUT"INSTRUCTIONS";A$:IF
LEFT$(A$,1)="Y"THEN CLS:GOTO 210

20 PO=0:G=0:S$=CHR$(134)+CHR$(13
7):E$=CHR$(139)
25 L=1024
30 PRINT@G," ";:PRINT@RND(31)+48
0,"*":PRINT@0,PO;
35 R=JOYSTK(0):H=R/2
50 IF PEEK(L+H)=106 THEN 100
60 PRINT@H,S$;
70 IF PEEK(65280)=126 THEN 79 EL
SE IF PEEK(65280)=254 THEN 79 EL
SE 90
79 SOUND235,1
80 FOR V=1 TO 16:IF PEEK(1056+H+
(32*V))=106 THEN 130 ELSE IF V<1
4 THEN POKE1024+H+(32*V),139:IF
V<14 THEN POKE1024+H+(32*V),143:
NEXT V:IF V<16 THEN 80 ELSE 30
90 PO=PO+1:GOTO 30
100 PLAY"L255;V31;O1ADCFBAGED;V1
6;ACEGAD;V4EABCAEDB"
110 DK=DK+1:IFDK=3THEN170ELSE30
120 FOR I=1 TO 100:NEXT I:GOTO 1
70
130 PO=PO+100:SOUND50,1:POKE1056
+H+(32*V),255:SOUND50,1:POKE1056
+H+(32*V),143:GOTO 30
170 CLS:IF PO>HP THEN 260 ELSE P
RINT@32,HP$":";HP;:PRINT@64,"YOU
 EARNED "PO" POINTS ON THAT MISS
ION";
180 PRINT@224,;:INPUT"   DO YOU
WANT  TO  TRY ANOTHER MISSION";Y
$:IF LEFT$(Y$,1)="N"THEN END ELS
E 20
210 FOR I=1 TO 20:PRINT@RND(31)+
480,"*":NEXT:PRINTTAB(15);"BY HO
LLIS HOLCOMB                MOOR
E,OK."
220 PRINT:PRINT"YOUR MISSION IS
TO BLAST A PATH THROUGH A METEOR
 STORM.YOU STEERWITH THE RIGHT J
OYSTK.YOU GET   100 POINTS FOR E
ACH METEOR YOU  SHOOT.SHOOT WITH
 RIGHT JOYSTK   BUTTON."
230 INPUT"PRESS ENTER TO BEGIN";
A$:GOTO 20
260 HP=PO:PRINT@160,CHR$(128);"Y
OU GOT HIGH SCORE OF"HP:INPUT"EN
TER YOUR NAME";HP$:CLS:GOTO 180
```

6809 Machine Code
by Bill Sias

This issue we're again going to move away from writing any machine language and look instead at the disassembler written by Larry Ashmun and marketed by Soft Sector Marketing. SSM has donated this program for you to type in or, for the cover dates of this issue, you may purchase it directly from them for $9.95 (regular price is $14.95). The listing is long so I'll quit here and let you get right into it.

```
1 '        DISASSEMBLER 6809
2 '        COPYRIGHT (C) 1981
3 '   SOFT SECTOR MARKETING INC.
4 '
5 '       WRITTEN BY L. ASHMUN
6 '
7 '            VERSION  1.4
9 CLS
10 GOSUB9000
20 GOSUB9300
48 GOTO1100
49 ' special dec. to hex.
50 D5=INT(D4/16):D6=D4-D5*16
55 D$=H$(D5)+H$(D6)
60 RETURN
99 ' 1 byte dec. to hex.
100 D=PEEK(A):A=A+1
105 DH=INT(D/16):DL=D-DH*16
110 D$=H$(DH)+H$(DL):OP$=OP$+D$
120 IFD)32AND D(91THENDA$=DA$+CH
    R$(D)
130 RETURN
149 ' 2 byte dec. to hex.
150 IFAD(0ORAD)65535THENAD$="err
    or":RETURN
155 A1=INT(AD/4096):AI=AD-A1*4096
160 A2=INT(AI/256):AI=AI-A2*256
170 A3=INT(AI/16):A4=AI-A3*16
180 AD$=H$(A1)+H$(A2)+H$(A3)+H$(
    A4)
190 RETURN
199 ' immediate mode
200 PR$="#":FORZ=1TOBT
210 GOSUB100:PR$=PR$+D$:NEXTZ
220 RETURN
249 ' direct mode
250 GOSUB100:PR$=D$
260 RETURN
299 ' extended mode
300 FORZ=1TO2
310 GOSUB100:PR$=PR$+D$:NEXTZ
320 RETURN
349 ' 1 byte relative mode
350 GOSUB100
360 IFD)127THEND=D-256
370 AD=D+A:GOSUB150:PR$=AD$
380 RETURN
399 ' 2 byte relative mode
400 GOSUB100:D1=D:GOSUB100:D2=D
410 D=D1*256+D2
420 IFD)32767THEND=D-65536
430 AD=D+A:GOSUB150:PR$=AD$
440 RETURN
499 ' indexed modes
500 R$="":R1$="":GOSUB100:XZ=D A
    ND16
505 ON(D AND96)/32GOTO515,520,525
510 R$="X":GOTO530
515 R$="Y":GOTO530
520 R$="U":GOTO530
525 R$="S"
530 IF(D AND128))0THEN545
535 D4=D AND15:IFXZ)0THENR1$="-"
    :D4=ABS(D4-16)
540 GOSUB50:PR$=R1$+D$+","+R$:RE
    TURN
545 PB=D AND15
550 ONPB GOTO560,565,570,575,580
    ,585,590,595,600,590,605,610,615
    ,590,620
555 R$=","+R$+"+":IFXZ)0THEN590E
    LSE655
560 R$=","+R$+"++":GOTO655
565 R$=",-"+R$:IFXZ)0THEN590ELSE
    655
570 R$=",--"+R$:GOTO655
575 R$=","+R$:GOTO655
580 R$="B,"+R$:GOTO655
585 R$="A,"+R$:GOTO655
590 PR$="bad pbyte":RETURN
595 R$=","+R$:GOTO625
600 R$=","+R$:GOTO635
605 R$="D,"+R$:GOTO655
610 R$=",PC":GOTO625
615 R$=",PC:GOTO635
620 GOSUB300:R$=PR$:GOTO655
625 GOSUB100:D4=D:IFD)127THEND4=
    ABS(D-256):R1$="-"
630 GOSUB50:GOTO650
635 GOSUB100:DD=D:GOSUB100:AD=DD
    *256+D
640 IFAD)32767THENAD=ABS(AD-6553
    6):R1$="-"
645 GOSUB155:D$=AD$
650 R$=R1$+D$+R$
655 IFXZ)0THENPR$="("+R$+")" ELS
    EPR$=R$
660 RETURN
799 ' push/pull group
800 GOSUB100
805 IFD=0THENPR$="bad pbyte":RET
    URN
810 IF(D AND128))0THENPR$=",PC"
```

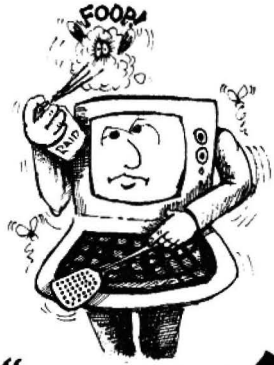
```
820 IF(D AND64))0THENPR$=PR$+",S
   /U"
830 IF(D AND32))0THENPR$=PR$+",Y"
835 IF(D AND16))0THENPR$=PR$+",X"
840 IF(D AND8))0THENPR$=PR$+",DP"
845 IF(D AND4))0THENPR$=PR$+",B"
850 IF(D AND2))0THENPR$=PR$+",A"
855 IF(D AND1))0THENPR$=PR$+",CC"
870 PR$=RIGHT$(PR$,LEN(PR$)-1)
875 IFPP=1THENPR$=PR$+T$ELSEPR$=
   T$+PR$
880 RETURN
899 ' transfer/exchange group
900 GOSUB100:X=0:J=(D AND240)/16
910 ONJ+1GOTO925,930,935,940,945
   ,950,920,920,955,960,965,970
920 PR$="bad pbyte":RETURN
925 PR$=PR$+"D":GOTO975
930 PR$=PR$+"X":GOTO975
935 PR$=PR$+"Y":GOTO975
940 PR$=PR$+"U":GOTO975
945 PR$=PR$+"S":GOTO975
950 PR$=PR$+"PC":GOTO975
955 PR$=PR$+"A":GOTO975
960 PR$=PR$+"B":GOTO975
965 PR$=PR$+"CC":GOTO975
970 PR$=PR$+"DP"
975 IFX=1THENRETURN
980 X=1:J=D AND15:PR$=PR$+",":GO
   TO910
999 ' hex. to dec.
1000 X=LEN(A$):X1=X+1:A=0
1010 IFX(1ORX)4THENRETURN
1020 FORZ=1TOX
1030 N=ASC(MID$(A$,X1-Z,1))
1040 IFN)47ANDN(58THENN=N-48:GOT
   O1070
1050 IFN)64ANDN(71THENN=N-55:GOT
   O1070
1060 N=0
1070 ONZ GOTO 1075,1080,1085,1090
1075 P=1:GOTO1095
1080 P=16:GOTO1095
1085 P=256:GOTO1095
1090 P=4096
1095 A=A+N*P:NEXTZ
1096 RETURN
1098 DATA 0,1,2,3,4,5,6,7,8,9,A,
   B,C,D,E,F
1099 ' program control
1100 CLEAR200:CLS
1105 T$="*":B=0
1110 DIMH$(15)
1115 DIMS(20):S=0
1120 FORZ=0TO15:READH$(Z):NEXTZ
1130 PRINT:INPUT"DISASSEMBLY STA
   RT ";A$
1131 IFA$="Z"THEN8000
1132 IFA$="P"THENPRINT:B=ABS(B-1
   ):IFB=1THENPRINT"printer on":GOT
   O1130 ELSEPRINT"printer off":GOT
   O1130
1135 IFA$=""THENA1=0:GOTO1140
1137 GOSUB1000:A1=A
1140 INPUT"END ADDRESS ";A$
1145 IFA$=""THENA$="FFFF"
1150 GOSUB1000:AE=A:IFAE(A1 THEN
   1140
1155 A=A1
1190 IFB=1THENPRINT#-2," "
1200 AP=A:FORZX=1TO16
1205 IFA=)AE ORA)65535THENZX=16:
   NEXTZX:GOTO1130
1210 OP$="":MN$="":PR$="":DA$=""
1215 L=0:X=0:AD=A
1220 GOSUB150:A$=AD$:GOSUB100:OP
   =D
1230 TP=(OP AND192)/64
1240 ONTP+1GOSUB2000,3000,4000,4
   000
1245 IFB=1THENPRINT#-2,A$;TAB(5)
   OP$;TAB(20)MN$;TAB(26);PR$;TAB(4
   5)DA$
1250 PRINT:PRINTA$;TAB(5)OP$;TAB
   (16)MN$;TAB(22);PR$;
1255 K$=INKEY$
1260 IFK$="X"THENZX=16:NEXTZX:GO
   TO1130
1262 IFK$="P"THENZX=16:NEXTZX:A$
   =K$:GOTO1132
1265 IFK$="Z"THENZX=16:NEXTZX:GO
   TO8000
1266 IFB=1THEN1278
1267 IFK$="R"THENZX=16:NEXTZX:GO
   TO1350
1268 IFK$="S"THENZX=16:NEXTZX:GO
   TO1300
1278 NEXTZX
1279 IFB=1THEN1200
1280 K$=INKEY$:IFK$=""THEN1280
1282 IFK$="Z"THEN8000
1283 IFK$="P"THENA$=K$:GOTO1132
1285 IFK$="X"THEN1130
1286 IFK$="S"THEN1300
1287 IFK$="R"THEN1350
1289 IFK$()CHR$(13)THEN1280
1290 PRINT:GOTO1200
1300 IFS=20THENPRINT:PRINT"stack
   full":GOTO1130
1310 S=S+1:S(S)=AP
1320 PRINT:PRINT"subroutine";S;"
   address";:INPUTA$
1325 IFA$=""THENA=0:GOTO1200
1330 GOSUB1000:GOTO1200
1350 IFS)0THENA=S(S):PRINT:PRINT
   "return";S:S=S-1:GOTO1200
1355 PRINT:PRINT"stack empty":GO
   TO1130
2000 IFOP(16 THEN3000
2005 IFOP)31 THEN2100
2010 ONOP-16 GOTO2020,2025,2030,
   2035,2035,2040,2045,2035,2050,20
   55,2035,2060,2065,2070,2075
2015 GOTO6000
2020 GOTO7000
2025 MN$="NOP":RETURN
```

```
2030 MN$="SYNC":RETURN
2035 MN$="bad opcode":RETURN
2040 MN$="LBRA":GOTO400
2045 MN$="LBSR":GOTO400
2050 MN$="DAA":RETURN
2055 MN$="ORCC":BT=1:GOTO200
2060 MN$="ANDCC":BT=1:GOTO200
2065 MN$="SEX":RETURN
2070 MN$="EXG":GOTO900
2075 MN$="TFR":GOTO900
2100 IFOP)47THEN2200
2105 ONOP-32GOTO2115,2120,2125,2
   130,2135,2140,2145,2150,2155,216
  0,2165,2170,2175,2180,2185
2110 MN$="BRA":GOTO2190
2115 MN$="BRN":GOTO2190
2120 MN$="BHI":GOTO2190
2125 MN$="BLS":GOTO2190
2130 MN$="BCC":GOTO2190
2135 MN$="BCS":GOTO2190
2140 MN$="BNE":GOTO2190
2145 MN$="BEQ":GOTO2190
2150 MN$="BVC":GOTO2190
2155 MN$="BVS":GOTO2190
2160 MN$="BPL":GOTO2190
2165 MN$="BMI":GOTO2190
2170 MN$="BGE":GOTO2190
2175 MN$="BLT":GOTO2190
2180 MN$="BGT":GOTO2190
2185 MN$="BLE"
2190 IFL=1THENMN$="L"+MN$:GOTO400
2195 GOTO350
2200 ONOP-48GOTO2215,2220,2225,2
   230,2235,2240,2245,2250,2255,226
  0,2265,2270,2275,2250,2280
2210 MN$="LEAX":GOTO500
2215 MN$="LEAY":GOTO500
2220 MN$="LEAS":GOTO500
2225 MN$="LEAU":GOTO500
2230 PP=0:MN$="PSHS":GOTO800
2235 PP=1:MN$="PULS":GOTO800
2240 PP=0:MN$="PSHU":GOTO800
2245 PP=1:MN$="PULU":GOTO800
2250 MN$="bad opcode":RETURN
2255 MN$="RTS":RETURN
2260 MN$="ABX":RETURN
2265 MN$="RTI":RETURN
2270 MN$="CWAI":BT=1:GOTO200
2275 MN$="MUL":RETURN
2280 MN$="SWI":RETURN
3000 MD=(OP AND48)/16:X=OP AND15
3005 ONX GOTO3020,3020,3030,3040
   ,3020,3050,3060,3070,3080,3090,3
  020,3100,3110,3120,3130
3010 MN$="NEG":GOTO3140
3020 MN$="bad opcode":RETURN
3030 MN$="COM":GOTO3140
3040 MN$="LSR":GOTO3140
3050 MN$="ROR":GOTO3140
3060 MN$="ASR":GOTO3140
3070 MN$="ASL":GOTO3140
3080 MN$="ROL":GOTO3140
3090 MN$="DEC":GOTO3140
3100 MN$="INC":GOTO3140
3110 MN$="TST":GOTO3140
3120 MN$="JMP":IFOP=78OROP=94THE
   N3020ELSE3140
3130 MN$="CLR"
3140 IFOP(16THEN250
3150 ONMD+1GOTO3160,3170,500,300
3160 MN$=MN$+"A":RETURN
3170 MN$=MN$+"B":RETURN
4000 X=OP AND15:BT=1
4005 ONX GOTO4020,4030,4040,4050
   ,4060,4070,4080,4090,4100,4110,4
  120,4130,4140,4150,4160
4010 MN$="SUB":GOTO4180
4020 MN$="CMP":GOTO4180
4030 MN$="SBC":GOTO4180
4040 BT=2:MN$="SUBD":IFOP)191THE
   NMN$="ADDD":GOTO4190 ELSE4190
4050 MN$="AND":GOTO4180
4060 MN$="BIT":GOTO4180
4070 MN$="LD":GOTO4180
4080 MN$="ST":IFOP=135OROP=199TH
   EN4200ELSE4180
4090 MN$="EOR":GOTO4180
4100 MN$="ADC":GOTO4180
4110 MN$="OR":GOTO4180
4120 MN$="ADD":GOTO4180
4130 BT=2:MN$="CMPX":IFOP)191THE
   NMN$="LDD":GOTO4190 ELSE4190
4140 IFOP=205THEN4200 ELSEIFOP=1
   41THENMN$="BSR":GOTO4190
4145 MN$="JSR":IFOP)191THENMN$="
   STD":GOTO4190 ELSE4190
4150 MN$="LD":BT=2:GOTO4170
4160 MN$="ST":IFOP=143OROP=207TH
   EN4200
4170 IFOP(192THENX$="X"ELSEX$="U"
4175 GOTO4185
4180 IFOP(192THENX$="A"ELSEX$="B"
4185 MN$=MN$+X$
4190 MD=(OP AND48)/16:IFOP=141 T
   HEN MD=4
4195 ONMD+1GOTO200,250,500,300,3
   50
4200 MN$="bad opcode":RETURN
6000 GOSUB100:OP=D
6010 IFOP(33THEN6190
6020 IFD(48THENL=1:GOTO2100
6030 IFD=63THENMN$="SWI2":RETURN
6040 IFD)191THEN6140
6050 IFD(131THEN6190
6060 D1=D AND15:D2=(D AND240)/16
   -7
6070 IFD1=3THENMN$="CMPD":GOTO61
   20
6080 IFD1=12THENMN$="CMPY":GOTO6
   120
6090 IFD1=14THENMN$="LDY":GOTO61
   20
6100 IFD1=15THENMN$="STY":GOTO61
   20
6110 GOTO6190
```

```
6120 BT=2:OND2 GOTO200,250,500,3
   00
6130 GOTO6190
6140 D1=D AND15:D2=(D AND240)/16
   -11
6150 IFD1=14THENMN$="LDS":GOTO61
   80
6160 IFD1=15THENMN$="STS":GOTO61
   80
6170 GOTO6190
6180 BT=2:OND2 GOTO200,250,500,3
   00
6190 MN$="bad opcode":RETURN
7000 GOSUB100
7010 IFD(63THEN6190
7020 IFD=63THENMN$="SWI3":RETURN
7030 IFD(131THEN6190
7040 D1=D AND15:D2=(D AND240)/16
   -7
7050 IFD1=3THENMN$="CMPU":GOTO70
   80
7060 IFD1=12THENMN$="CMPS":GOTO7
   080
7070 GOTO6190
7080 BT=2:OND2 GOTO200,250,500,3
   00
7090 GOTO6190
8000 PRINT:INPUT"zap display sta
   rt ";A$
8005 IFA$="P"THENPRINT:B=ABS(B-1
   ):IFB=1THENPRINT"printer on":GOT
   08000 ELSEPRINT"printer off":GOT
   08000
8006 IFA$="R"THEN1130
8015 IFA$=""THENA=AP:GOTO8030
8020 GOSUB1000
8030 IFA(0THENA=0
8035 IFB=1AND A)65535THENA=0
8040 IFA)65440THENA=65440
8042 AB=A
8045 CLS:IFB=1THENPRINT#-2," "
8050 FORZL=1TO16
8055 AD=A
8060 GOSUB150
8065 IFB=1THENPRINT#-2,AD$;"    ";
8070 PRINT:PRINTAD$;"    ";
8075 OP$="":DA$="":DB$=""
8080 FORZG=1TO3
8090 D$="":FORZB=1TO2
8100 GOSUB100
8110 PRINTD$;
8115 IFB=1THENPRINT#-2,D$;
8120 IFD)31 ANDD(91THENDB$=DB$+C
   HR$(D)ELSEDB$=DB$+"."
8130 NEXTZB:PRINT"   ";
8135 IFB=1THENPRINT#-2,"   ";
8140 NEXTZG:PRINTDB$;
8141 IFB=1THENPRINT#-2,DB$
8142 K$=INKEY$:IFK$="X"ORK$="R"O
   RK$="P"THENZL=16:NEXTZL:GOTO8180

8150 NEXTZL
8155 IFB=1THENK$=INKEY$:GOTO8180
8160 K$=INKEY$:IFK$=""THEN8160
8165 IFK$="-"ORK$="="THENA=AB-96
   :GOTO8030
8170 IFK$=";"ORK$="+"THEN8030
8175 IFK$="A"THENA=AB:GOTO8030
8180 IFK$="X"THEN8000
8190 IFK$="R"THENA=AP:GOTO1190
8192 IFK$="P"THENA$=K$:GOTO8005
8195 IFB=1THEN8030
8200 GOTO8160
9000 PRINT@135,"DISASSEMBLER 680
   9"
9010 PRINT:PRINTTAB(7)"COPYRIGHT
    (C) 1981"
9020 PRINTTAB(3)"SOFT SECTOR MAR
   KETING INC."
9030 PRINT:PRINTTAB(6)"WRITTEN B
   Y L. ASHMUN"
9050 PRINT:PRINTTAB(7)"INSTRUCTI
   ONS ? Y/N"
9060 K$=INKEY$
9070 IFK$="N"THENRETURN
9080 IFK$="Y"THEN9100
9090 GOTO9060
9100 CLS
9110 PRINTTAB(7)"COMMANDS AVAILA
   BLE"
9115 PRINTTAB(7)"(DISASSEMBLY MO
   DE)"
9120 PRINT:PRINT" X = NEW START
   ADDRESS      **"
9130 PRINT" S = GOTO SUBROUTINE
          *"
9140 PRINT" R = RETURN FROM SUBR
   OUTINE *"
9150 PRINT" Z = GOTO ZAP DISPLAY
    MODE"
9160 PRINT" P = PRINTER ON/OFF S
   WITCH"
9162 PRINT" ENTER = NEXT PAGE
         *"
9165 PRINT:PRINT" * NOT ACTIVE W
   ITH PRINTER on."
9167 PRINT"** X WILL NOT CANCEL
   PRINTER"
9170 PRINT"      OUTPUT."
9180 PRINT:PRINTTAB(10)"PRESS EN
   TER"
9185 K$=INKEY$:IFK$()CHR$(13)THE
   N9185
9190 CLS
9200 PRINTTAB(7)"COMMANDS AVAILA
   BLE"
9210 PRINTTAB(7)"(zap display mo
   de)"
9220 PRINT:PRINT" X = NEW START
   ADDRESS  **"
9230 PRINT" + = INCREMENT ONE PA
   GE  *"
9240 PRINT" - = DECREMENT ONE PA
   GE  *"
9250 PRINT" A = REDISPLAY SAME P
   AGE *"
```

```
9260 PRINT" R = RETURN TO DISASS
   EMBLY MODE"
9270 PRINT" P = PRINTER ON/OFF S
   WITCH"
9275 PRINT:PRINT" * NOT ACTIVE W
   ITH PRINTER on."
9280 PRINT"** X WILL NOT CANCEL
   PRINTER"
9285 PRINT"      OUTPUT."
9290 PRINT:PRINTTAB(10)"PRESS EN
   TER";
9295 K$=INKEY$:IFK$()CHR$(13)THE
   N9295
9299 RETURN
9300 CLS
9310 PRINT"  PRINTER BAUD RATE S
   ELECTION"
9320 B=PEEK(150)
9330 IFB=190THENB$="300 BAUD"
9340 IFB=87THENB$="600 BAUD"
9350 IFB=41THENB$="1200 BAUD"
9351 IFB=18THENB$="2400 BAUD"
9380 PRINT:PRINT"   CURRENT RATE
   - ";B$
9390 PRINT:PRINT
9400 PRINT" 1 = 300 BAUD"
9410 PRINT" 2 = 600 BAUD"
9420 PRINT" 3 = 1200 BAUD"
9422 PRINT" 4 = 2400 BAUD"
9430 PRINT:PRINT"   PRESS ENTER
   FOR NO CHANGE"
9450 PRINT:PRINT:INPUT"    WHICH
   BAUD RATE ";B$
9455 POKE155,80
9460 IFB$=""THENRETURN
9470 IFB$="1"THENPOKE150,190
9480 IFB$="2"THENPOKE150,87
9490 IFB$="3"THENPOKE150,41
9492 IFB$="4"THENPOKE150,18
9495 RETURN
```



"BUGS!"

A Simple Screen Pointer Manipulation Program

by Mark Rothstein

Are you like me? I own an Extended Basic Color Computer, and I like to know what's going on inside. Sometimes I'd like to do peculiar things that Basic won't let me do; other times I'd just like to know. Well here's a short program that will start you on your way.

The program takes a value you enter, from 0 to 31, and uses that to move the video display area that is controlled by the Synchronous Address Multiplexer (SAM). Each unit represents a 512 byte offset. A "0" entry points you at zero; a "1" points the screen to location 512, etc.

This way you can see what's going on in Basic's RAM area, watch your program variables change, or just see how BASIC stores your program. Try running the program with a zero offset and see what happens when you press "Shift 0". Location $11A (the $ represents hexadecimal) will change from a blank space to an "@" in reverse video. Also, if you press any key beside the up arrow, you will see characters changing in locations $152 - 159. This is where Basic figures out what key you've pressed.

Still interested? Try replacing line 230 with:

230 CSAVE"TEST"

or

230 RENUM

Can you see which locations are changing now? Watch out, though. RENUM and BREAK don't return to the program. They exit back to Basic with the screen pointer exactly where you left it. This could be the middle of nowhere! To get out of this, press the RESET button, or hit:

BREAK <ENTER>
RUN <ENTER>
2 <ENTER>

This resets the pointer back to where Basic expects it -- $400. You'll have to enter the keys without seeing the visual feedback on the screen, though.

Other interesting areas are cassette file names at locations $1D2 and $1DA; also the screen buffer at $2E1. Incidentally the screen buffer stores only the latest 90 keyboard entries. Or hook up the joysticks, and read the joystick value in the program loop:

add 225 J=JOYSTK(0)
and change 230 IF INKEY$="" THEN 225.

Now as you change the position of the joysticks note that locations $15A thru $15D also change.

This is only the start! Although Radio Shack doesn't provide you with listings of what goes on inside, the Color Computer is certainly powerful enough to help you figure it out.

```
10 'THIS IS A PROGRAM THAT ALLOWS
   YOU TO PEEK INTO RAM
20 'BY MOVING THE VIDEO DISPLAY
   AREA IN THE SAM
30 '
40 'BY MARK ROTHSTEIN
50 '  3123 WALNUT AVE.
60 '  OWINGS MILLS, MD 21117
70 '
80 'DIRECTIONS:
90 'WHEN THE COMPUTER ASKS FOR AN
   OFFSET, ENTER A VALUE FROM 0 TO 31
100 'THIS MOVES THE VIDEO DISPLAY
   IN UNITS OF 1/2 K BYTES
110 '
120 'AFTER YOU HIT <ENTER>, THE
   COMPUTER WILL MOVE THE DISPLAY
   POINTER.
130 'TO TRY A NEW AREA OF MEMORY,
   JUST PRESS THE UP ARROW.
140 'THE COMPUTER WILL REPLACE THE
   POINTER SO YOU CAN SEE WHAT YOU ARE
   DOING.
150 '
160 'GET THE INPUT OFFSET VALUE.
170 INPUT"OFFSET";O
180 '
190 'GO TO THE SUBROUTINE TO DO ALL
   THE HARD WORK
200 GOSUB 300
210 '
220 'THEN WAIT FOR AN UP ARROW INPUT
230 IF INKEY$<>"up arrow" THEN 230
240 '
250 'NOW RESTORE THE DISPLAY (USING
   THE SAME SUBROUTINE!)
260 O=2: GOSUB 300
270 GOTO 170
```

Continued on Page 47

CSAVE INSURANCE

by Jorge Mir
12851 W. Balboa Dr.
New Berlin, WI 53151

Have you ever CSAVEd a program after many hours of debugging, endless revisions and tiresome tests only to subsequently find out that you had a defective tape? or for whatever reason the program cannot be CLOADed again? If it happened once, that's once too many times!

Here is a simple way to assure yourself of a good copy (or several good copies) before you turn your machine off. When you are ready to CSAVE the program, type the following and hit <ENTER>:

PRINT PEEK(25) PEEK(26)

The screen will then show two numbers, such as:

30 1

This is the decimal address where your Basic program starts. Jot it down so you don't forget it. Next, type the following and hit <ENTER>:

PRINT PEEK(27) PEEK(28)

The screen will again show two numbers. This is the decimal address for the start of simple variables. Jot those numbers down so you don't forget them either. CSAVE your program as many times as you wish. Now, type the following and hit <ENTER>.

POKE 25, PEEK(27): POKE 26, PEEK(28)

This moves the Basic program beyond the end of the original program so that your original program is not erased from memory when loading a new program.

Now, CLOAD the copies you made (one at a time, of course) and check each copy out by RUNning it or LISTing it. If everything as OK, then go ahead and turn your machine off. The copies you made are OK. However, if you are not able to CLOAD the new copies, or when LISTing or RUNning them you find there are problems, then don't worry! You still have the original copy in memory!

To get back to the original copy of your program just type the following (the '#' indicates the first code you jotted down as discussed above):

POKE 25,#: POKE 26,#

This resets the start of Basic back to the beginning of the original program. You can now RUN or LIST the original program. You can also CSAVE it again! Then, you can follow the same routine over again until you know you have a good copy or copies on tape.

By the way if you want to return to the second program, You can type the following (using the second set of numbers you jotted down) and hit <ENTER>:

POKE 25,#: POKE 26,#

That's all there is to it!

continued from page 19

```
);CHR$(207);CHR$(207);CHR$(154);

170 PRINT
180 PRINT@128+C,CHR$(151);CHR$(1
55);
190 IF PEEK(344)=247THEN C=C+1
200 IF PEEK(343)=247THEN C=C-1
210 IF C>29THENC=29
220 IF C<1THENC=1
230 R9=R8:R8=R7:R7=R6:R6=R5:R5=R
4:R4=R3:R3=R2:R2=R1:R1=R
240 IF C<=R9 THEN C=C+2:GOSUB 31
0
250 IF C>=R9+6 THEN C=C-2:GOSUB
310
260 PRINT@160+C,CHR$(226);CHR$(2
25);
270 IF CR>4 THEN 110
280 PT=PT+1
290 NEXT Y
300 GOTO110
310 SOUND 150,5
320 PRINT@160+C,CHR$(130);CHR$(1
29);
330 PRINT@10,"CRASH!!!!!";
340 PRINT@ 42,"POINTS ="PT;
360 TT=TT+PT
370 PT=0
380 CRASH=CRASH+1
390 IF CRASH>4 THEN 420
400 FOR T=1 TO 1000:NEXT
410 RETURN
420 PRINT@ 100,"TOTAL POINTS ="T
T;
430 PRINT@450,"PRESS <ENTER> FOR
 A NEW GAME  ";
440 INPUT X
450 PRINT@416,""
460 TT=0
470 CR=0
480 RUN90
```

DISKS!
by Bill Sias

Last issue we had ads about two new disk controllers for the Color Computer. Since that went to press Tandy has announced their's and rumors have mentioned three others. In this article I will attempt to answer some questions people have asked me about disks, their operation and use. Most of this discussion will relate to how to use a disk and DOS and in future articles we'll get into it deeper as in how it performs it's operations and the necessary hardware.

What is a disk?

Physically a disk is similar to a soft 45 rpm record inside a perminent jacket. Although it is very flexable one of the quickest way to damage it is to bend it. It has three holes in it. The large one is the opening for the read / write head. You can see the disk through this hole and by touching the disk through it you can totally destroy the data on the disk. The next is the center hole this is used by the disk drive for positioning and to allow the disk to spin within it's jacket. The last hole is very small and is used by the disk drive to view a small hole on the disk itself so that it is constantly indexed to the correct position. On "soft sectored" disks their is one of these index holes, on "hard sectored" disks their are ten holes. Soft sectored disks are used by Radio Shack and Exatron and Hard sectored are used by Tallgrass Technology. In use there is very little functional difference between the two types of disks.

What do I gain by owning a "Disk"?

Well, the first thing you gain is rapid loading and saving of programs. Unscientifically speaking, about 1000 times faster than cassette. The second thing you gain is the ability to store data efficiently. If you have tried cassette data files at all you know the gross inefficency of that medium. With a disk you can have immediate access to somewhere between 86 and 200K of information with a cassette you can handle only what your memory allows. Another thing that you gain is tidyness. Personally I store one program per cassette, that saves me the hassle of having to SKIP over any other files before I come to the program I want (no, I don't own stock in a cassette manufacturing firm). With my disks I have one with all of my games, another with all of my utilities, etc. I can load any of them faster than you can find the right cassette.

What disadvantages does a Disk have?

Well the first and most obvious is cost. But even that is relative, when you consider the fact that a single disk holds much more than a cassette and can cost less than one of Radio Shack's certified cassettes, which in the long run will save you money (unless you are one of those folks that fill a C60 cassette completely). The other disadvantage is that, in my opinion, you will have to have 32K with any of the systems to make it practical.

What does it cost?

Again that depends on the system that you buy. Let's look at the three major systems available now. This comparision assumes that you don't want to modify your computer or void your warrentee in any way. It also assumes that you now have Extended Basic and 16K of RAM. All prices are taken from either last issue of CCN or Radio Shack's Catalog.

DISKS!

Item	Radio Shack	Exatron	Tallgrass
32K RAM	$149.95	NA	$149.95
Controller	$599.95	$298.00	$ 99.95
1 Drive	NA	$329.95	$329.95
DOS	NA	$ 29.95	$ 69.95
Total	$749.90	$657.90	$649.80

What about compatability?

Judging by past performance, Radio Shack will use their own DOS exclusively. Both Exatron and Tallgrass have introduced their own DOS. Compatability between the three of them will depend primarily on the manufacturers themselves and other outside vendors. Exatron is already working on a Radio Shack compatable version for their CCI so that will be compatable, with Tallgrass' hard sectored disks it will be difficult at best to develope a DOS compatable with Radio Shack's. Steve Odneal and I are working on making FLEX[tm] available for the Color Computer using Exatron's CCI. From past experience I feel safe in saying that someone, somewhere will probably create a method of interchange between them.

Can you compare the three systems in use?

Not really. The only experience I have with any of them is the Exatron CCI. I can say, however, that I like it very much. One of the advantages it has is that the DOS resides in RAM that is located just above the memory that the program paks usually use, which makes it easily modifiable. For example, I have been used to using CPM, TRS-DOS and NEWDOS 80, therefore I found the command CAT very hard to get used to (most systems use DIR), so I changed it to DIR. The other thing I liked was the fact that it has a command to load Model I BASIC programs into the CC, which allowed me to use a lot of software I had put away. I also have added a new command that I call CAT (I thought you didn't like that?) that reads the directorys of all of my disks and puts them into a disk file, this allows me to keep a record of all my programs and where they are located on one disk. Granted not everyone is going to modify the DOS so that "advantage" may have limited appeal except when you consider that without the RAM at locations $C000 and up developing a system that will allow you to use the FLEX[tm] operating system from TSC would have been difficult if not impossible. This alone will open up more software for the Color Computer than Radio Shack could ever develope and allows us a method of interchange with our "big" brothers like SWTPC, GIMIX and Smoke Signal Broadcasting. Another feature it has is the ability to "back-up" ROM paks. At first I was upset about this feature as I thought it was just another tool for software pirates, but I recently used it to put Radio Shack's Personal Finance program on disk for a friend and patched it so it would write the file to the disk instead of the cassette.

Which one will CCN be supporting?

All of them. Right now we have only the Exatron CCI and as such it is the only one we directly support right now, and since it will be compatable with Radio Shack's we will be better able to support it by using the Exatron system. We will, however, accept articles about any or all of them and as we purchase them will make a solid commitment to each. We have not made support of any of them magazine policy.

DISKS!

You keep using the term "DOS", what does it mean?

DOS, pronounced Doss, stands for Disk Operating System. As time goes on we'll have more articles about all of these systems and more detail about DOSs in general and specifically. But for now you probably have more information than you wanted anyway. If you have any experience with any of these please write and let others know about it.

Tell me more about Exatron's CCI.

Exatron's DOS is quite interesting, first of all it really doesn't have a "DOS level" and a "BASIC level", you are operating from both all of the time. For example, you can use any DOS command as a statement from your BASIC program. Machine language programmers will find the extra 16K, located above the BASIC ROMs very handy. I talked with one fellow who is developing a word processor using the Exatron system and he likes it because the word processor is located in the high bank of 16K which allows a 32K text buffer. Because the DOS only requires 8K of RAM that leaves another 8K that is protected from BASIC for you to add your own commands to the system, I use that area for my SORT command and have used it to draw graphic screens before displaying them in my machine language programs. In fact this month's letters column includes a program that puts a screen print routine in that memory space.

You sound like you think the Exatron CCI is the best.

First you have to keep in mind my situation. Because of the magazine I have to stay compatable with as many of you as possible without buying all of the systems that will ever become available, with the CCI I can do just that. In some cases it may require that I personally write the software to be compatable but that's alot cheaper than owning all of the controllers. Second, I write alot of long machine language programs and I need as much memory as I can get, with the CCI I have the full 48K at my disposal. In addition, being able to "instantly" load and save my source code files adds some insurance to never losing them, with cassette I save the files at the end of the session, now I save frequently.

A Simple Screen Pointer

```
280 '
290 'SUBROUTINE
300 A0=&HFFC6
310 '
320 FOR I=1 TO 7
330 A=INT(O/2)
340 B=O-2*A
350 O=A
360 POKE B+A0,0
370 A0=A0+2
380 NEXT I
390 RETURN
400 END
```


COLOR COMPUTER DISK SYSTEM

SPECIAL LIMITED OFFER

CO RESIDENT EDITOR/ASSEMBLER

TEXTPRO
TEXT EDITOR/PROCESSOR

SYSTEM MONITOR

DATAPROS
DATA COMMUNICATIONS PACKAGE

5566 Ricochet Avenue
Las Vegas, Nevada 89118

CER-COMP
(702) 452-0632

All Orders Shipped From Stock
Add $1.00 Postage – MC/Visa Add 3%

Kid's Page

```
 1 'LIGHTNIN
 2 'LIGHTNING 1.0
 3 'by Billy Sills (age 8)
 4 'July 26, 1981
 10 CLS2
 20 PRINT"POW BANG"
 30 CLS4
 40 CLS8
 50 GOTO 20
```

I saw your ad for the Kid's Page and thought I would send in a program. I have a TRS 80 Color Computer. I an 11 years old. I have enclosed a program that my brother helped me on and a picture.

```
1 REM JUNK LETTER
5 REM BY NELL RUX
10 PRINT"  THIS PROGRAM IS JUNK"
20 PRINT: PRINT: PRINT: PRINT: PRINT
:PRINT
23 PRINT"      AND SO ARE YOU.."
24 PRINT: PRINT: PRINT
25 PRINT"           SINCERELY,"
27 PRINT"         YOUR COMPUTER"
30 T=RND(255)
40 X=RND(8)
50 CLS(X)
60 SOUND T,2
70 GOTO 10
```

Dear Sirs;
I 12 years of age and very interested in computers. I have a Radio Shack TRS-80 Color Computer which gets a lot of use. I have a question. How does a joystick work???
Sincerely,
Adam Rux

Adam,
The answer to your question would make a very long article depending on the detail the author went into. Anyway, before you can understand how it works you have to know what it is. The joystick itself contains two variable resistors connected to the handle. As you move the handle you cause the resistance to change. The Color Computer has a circuit inside called an "Analog to Digital Converter". The ADC'S job is to measure this resistance and give the Color Computer a number that is proportional to the amount of resistance it found in each of the variable resistors. The entire process is more complex than this the basic princepal is true. I hope this makes it a bit (bad pun) clearer. I have a question. Do you REALLY eat goldfish? YUCCHHH!!!
Bill

```
 1 REM GOLD FISH
 5 REM BY ADAM RUX
10 PRINT"THIS IS THE GOLDFISH GAME"
20 PRINT" "
30 PRINT"WOULD YOU LIKE
INSTRUCTIONS?"
40 INPUT D$
50 IF LEFT$(D$,1)="N" THEN 70
60 PRINT"THE OBJECT OF THE GAME IS
TO EAT AS MANY GOLDFISH YOU CAN
WITHOUT BERPING."
70 A=RND(100)
75 PRINT
80 PRINT"HERE'S THE GOLDFISH..."
81 FOR K=1 TO 2000
82 NEXT K
85 PRINT "**** ********* ********
******"
90 PRINT"<<<<< <<<<<CHOW
BROTHER>>>>> >>>>>"
91 PRINT "**** ********** *******
******"
92 FOR T=1 TO 500
95 NEXT T
96 FOR I=1 TO 15
100 PRINT" I "
110 NEXT I
120 IF A<50 GOTO 140
130 IF A>50 GOTO 150
140 PRINT" I ******* SLOOSSHHH
********** THAT ONE SLID DOWN YOUR
THROAT LIKE MILK !!!!!!!!!!!!"
145 GOTO 70
150 PRINT" I ******* BERRRP
************ YOU NOW FACE THE
CONSEQUENCES OF INDIGESTION !!!!!!!!!
THE ONLY REMEDY IS PEPTO-BISMOL."
160 PRINT" DO YOU WANT TO EAT SOME
MORE"
170 INPUT B$
180 IF LEFT$(B$,1)="Y" THEN 70
185 PRINT"    GOOD BY"
190 END
```

TAPETYPE

Have you ever wondered just what is on that unlabeled tape in the bottom of your tape basket? Or where a machine language cassette loads? Or why your favorite program tape gets an I/O error?

This program tells you just what is on a tape, record by record. Each record that is encountered is dumped verbatim to the screen, while the program decodes all the information as to record type, file type, load addresses, checksum errors and so forth.

The listing of the program which is given below is interesting as an example of Position Independent Code (P.I.C.), in-line parameters and stack based variables.

This program may be typed into your Editor Assembler (it was written on the SDS80C from the Micro Works). If you don't feel like typing, it is available in object on cassette from the Micro Works for $14.95.

```
0001 0600                          NAM TAPETYPE
                                   *
                                   *  ANDREW E. PHELPS
                                   *  THE MICRO WORKS
                                   *  21 SEPTEMBER 1981

                                   *  ROM ENTRY POINTS -
0002 A701                          SREAD  EQU $A701        SYNC AND READ
0003 A70B                          READ   EQU $A70B        PLAIN READ
0004 A77C                          SYNC   EQU $A77C        READ $55'S

                                   *  BLOCK READ PARAMETERS -
0005 007C                          BTYPE  EQU $7C          BLOCK TYPE
0006 007D                          BLEN   EQU $7D          BLOCK LENGTH
0007 007E                          BADDR  EQU $7E          BLOCK ADDRESS

0008 0400                          SCREEN EQU $400
0009 0500                          BUFFER EQU SCREEN+$100
                                   *  VARIABLES (ON STACK) -
0010 0000                          FILLEN EQU 0            FILE LENGTH
0011 0002                          SFLAG  EQU 2            SYNC FLAG
0012 0003                          EOFLAG EQU 3            FOUND END-OF-FILE
0013 0004                          STAD   EQU 4            START ADDRESS
0014 0006                          VLEN   EQU 6            TOTAL STACK BYTES


0015 0600 327A                     START  LEAS -VLEN,S     VAR SPACE
0016 0602 1F43                            TFR S,U          VARIABLE POINTER
0017 0604 17017D                          LBSR INITSC      SET UP SCREEN
0018 0607 CC0000                          LDD #0
0019 060A EDC4                            STD FILLEN,U     CLEAR LEN
0020 060C ED44                            STD STAD,U       CLEAR S.A.
0021 060E 6F42                            CLR SFLAG,U      SYNC FIRST
0022 0610 6F43                            CLR EOFLAG,U     NOT END FILE

                                   *  MAIN LOOP -
                                   *  READ A RECORD AND CALL THE
                                   *  APPROPRIATE PROCESSOR.
0023 0612 8E0500                   RECORD LDX #BUFFER
0024 0615 9F7E                            STX BADDR        BLOCK TO SCREEN
0025 0617 B60400                          LDA SCREEN
0026 061A 8A40                            ORA #$40
0027 061C B70400                          STA SCREEN       FLASH CORNER
0028 061F 6D42                            TST SFLAG,U
0029 0621 2705                            BEQ A@
```

```
0030 0623 BDA70B            JSR READ          NON-SYNC READ
0031 0626 2003              BRA B@
0032 0628 BDA701      A@    JSR SREAD         SYNC READ
0033 062B             B@
0034 062B 2712              BEQ C@            CHECKSUM ERROR?
0035 062D 108E042F          LDY #SCREEN+ES
0036 0631 170190            LBSR MSG
0037 0634 2A4552524F        FCC /*ERROR*/
0038 063B 8680              LDA #$80          FILL BLACK
0039 063D 2010              BRA R@
0040 063F 108E042F    C@    LDY #SCREEN+ES
0041 0643 17017E            LBSR MSG
0042 0646 4F4B202020        FCC /OK     /
0043 064D 8660              LDA #$60          FILL GREEN
0044 064F 170149      R@    LBSR FILL         FILL SCREEN

0045 0652 6D43              TST EOFLAG,U      NEW FILE?
0046 0654 270A              BEQ S@            SKIP IF NOT
0047 0656 108E046C          LDY #SCREEN+FN
0048 065A 170182            LBSR CLR4         BLANK OUT NAME
0049 065D 17017F            LBSR CLR4
0050 0660 6F43        S@    CLR EOFLAG,U

0051 0662 108E044E          LDY #SCREEN+RT
0052 0666 967C              LDA BTYPE         BLOCK TYPE
0053 0668 10270017          LBEQ HEADER       0=NEW FILE
0054 066C 4C                INC A             WAS IT -1?
0055 066D 102700DC          LBEQ EOFILE       END OF FILE
0056 0671 8102              CMPA #2           WAS IT +1?
0057 0673 102700BF          LBEQ DATABL       DATA BLOCK
0058 0677 17014A            LBSR MSG
0059 067A 494C4C4547        FCC /ILLEGAL/
0060 0681 208F              BRA RECORD

                      *  HEADER BLOCK -
                      *  DISPLAY FILE NAME & INFO
                      *  AND SELECT SYNC / NO SYNC
0061 0683 17013E      HEADER LBSR MSG
0062 0686 4845414445        FCC /HEADER /
0063 068D 108E04DC          LDY #SCREEN+EA
0064 0691 17014B            LBSR CLR4         CLEAR END ADDR
0065 0694 108E046C          LDY #SCREEN+FN
0066 0698 8E0500            LDX #BUFFER
0067 069B 170135            LBSR COPY         DISPLAY NAME
0068 069E 108E048C          LDY #SCREEN+FT
0069 06A2 B60508            LDA BUFFER+8      FILE TYPE
0070 06A5 2712              BEQ BASFIL        BASIC?
0071 06A7 4A                DEC A
0072 06A8 271B              BEQ DATFIL        DATA FILE?
0073 06AA 4A                DEC A
0074 06AB 2724              BEQ MACFIL        MACHINE LANG.?
0075 06AD 170114            LBSR MSG
0076 06B0 494C4C4547        FCC /ILLEGAL/
0077 06B7 203A              BRA CLRIT

0078 06B9 170108      BASFIL LBSR MSG
0079 06BC 4241534943        FCC /BASIC  /
0080 06C3 202E              BRA CLRIT
0081 06C5 1700FC      DATFIL LBSR MSG
0082 06C8 4441544120        FCC /DATA   /
```

```
0083 06CF 2022              BRA CLRIT
0084 06D1 1700F0      MACFIL LBSR MSG
0085 06D4 4D41434849        FCC /MACHINE/
0086 06DB 108E04B3          LDY #SCREEN+SA
0087 06DF FC050B            LDD BUFFER+11
0088 06E2 1700C4            LBSR HEXOUT
0089 06E5 108E04CF          LDY #SCREEN+LA
0090 06E9 FC050D            LDD BUFFER+13
0091 06EC ED44              STD STAD,U      SAVE LOAD AD
0092 06EE 1700B8            LBSR HEXOUT
0093 06F1 2013              BRA NEXT

0094 06F3 108E04B3    CLRIT LDY #SCREEN+SA
0095 06F7 1700E5            LBSR CLR4
0096 06FA 108E04CF          LDY #SCREEN+LA
0097 06FE 1700DE            LBSR CLR4
0098 0701 CC0000            LDD #0
0099 0704 ED44              STD STAD,U      NO LOAD AD

0100 0706 CC0000      NEXT  LDD #0
0101 0709 EDC4              STD FILLEN,U    CLEAR LENGTH
0102 070B 8D6F              BSR WLEN        DISPLAY LENGTH

0103 070D 108E0494          LDY #SCREEN+FT+8
0104 0711 B6050A            LDA BUFFER+10   GAP TYPE
0105 0714 270C              BEQ L@          SKIP IF NO GAPS
0106 0716 1700AB            LBSR MSG
0107 0719 4153434949        FCC /ASCII  /
0108 0720 2011              BRA M@
0109 0722 17009F      L@    LBSR MSG
0110 0725 42494E4152        FCC /BINARY /
0111 072C 8601              LDA #1
0112 072E A742              STA SFLAG,U     FLAG NO SYNCS
0113 0730 BDA77C            JSR SYNC        DO FIRST SYNC
0114 0733 16FEDC      M@    LBRA RECORD

                    *  DATA BLOCK -
                    *  DISPLAY AND COUNT FILE SIZE
0115 0736 17008B    DATABL LBSR MSG
0116 0739 4441544120        FCC /DATA   /
0117 0740 ECC4              LDD FILLEN,U
0118 0742 DB7D              ADDB BLEN       ADD ON FILE LEN
0119 0744 8900              ADCA #0
0120 0746 EDC4              STD FILLEN,U
0121 0748 8D32              BSR WLEN        DISPLA FILE LEN
0122 074A 16FEC5            LBRA RECORD

                    *  END OF FILE BLOCK -
                    *  DISPLAY FINAL SIZE
                    *  (NAME AND LENGTH ARE ZEROED
                    *  IN CASE NEXT BLOCK IS NOT
                    *  A HEADER.)
0123 074D 8D75      EOFILE BSR MSG
0124 074F 454E444649        FCC /ENDFILE/
0125 0756 6F42              CLR SFLAG,U     BACK TO SYNCS
0126 0758 8601              LDA #1
0127 075A A743              STA EOFLAG,U    CLEAR NAME
0128 075C 8D1E              BSR WLEN        DISPLAY LENGTH
0129 075E 108E04FC          LDY #SCREEN+FL+14
0130 0762 ECC4              LDD FILLEN,U
```

```
0131 0764 8D43              BSR HEXOUT      DISPLAY AGAIN
0132 0766 ECC4              LDD FILLEN,U
0133 0768 E344              ADDD STAD,U     START ADDR
0134 076A 830001            SUBD #1         "THRU"- NOT "TO"
0135 076D 108E04DC          LDY #SCREEN+EA
0136 0771 170035            LBSR HEXOUT     DISPLAY END
0137 0774 CC0000            LDD #0
0138 0777 EDC4              STD FILLEN,U    ZERO LENGTH
0139 0779 16FE96            LBRA RECORD

                        *  DISPLAY CURRENT LENGTH -
0140 077C 108E04EE      WLEN    LDY #SCREEN+FL
0141 0780 ECC4                  LDD FILLEN,U
0142 0782 2025                  BRA HEXOUT

                        *  SET UP INITIAL SCREEN -
0143 0784 8E0400        INITSC LDX #SCREEN
0144 0787 318D005F             LEAY BANNER,PCR
0145 078B A6A4          F@      LDA 0,Y         END OF PATTERN?
0146 078D 2702                  BEQ G@
0147 078F E6A0                  LDB ,Y+
0148 0791 CA40          G@      ORB #$40        NON-INVERT
0149 0793 E780                  STB ,X+
0150 0795 8C0600                CMPX #SCREEN+$200
0151 0798 25F1                  BLO F@
0152 079A 39                    RTS
                        *  FILL UNUSED BUFFER SPACE -
                        *  CALLED WITH COLOR IN "A".
0153 079B 8E0500        FILL    LDX #BUFFER
0154 079E D67D                  LDB BLEN        BLOCK LENGTH
0155 07A0 3A                    ABX
0156 07A1 A780          D@      STA ,X+
0157 07A3 8C0600                CMPX #BUFFER+$100
0158 07A6 26F9                  BNE D@
0159 07A8 39                    RTS

                        *  DISPLAY NUMBER IN HEX -
                        *  NUMBER IN "D", SCREEN ADDRESS
                        *  IN "Y"
0160 07A9 8D02          HEXOUT BSR HEXBYT
0161 07AB 1F98                  TFR B,A
0162 07AD 3402          HEXBYT PSHS A
0163 07AF 44                    LSR A
0164 07B0 44                    LSR A
0165 07B1 44                    LSR A
0166 07B2 44                    LSR A
0167 07B3 8D02                  BSR HEXNYB
0168 07B5 3502                  PULS A
0169 07B7 840F          HEXNYB ANDA #$0F
0170 07B9 8109                  CMPA #9
0171 07BB 2302                  BLS Q@
0172 07BD 8BC7                  ADDA #7-$40
0173 07BF 8B70          Q@      ADDA #$70
0174 07C1 A7A0                  STA ,Y+
0175 07C3 39                    RTS

                        *  DISPLAY 7-CHARACTER MESSAGE
                        *  (IN-LINE PARAMETER)
                        *  SCREEN ADDRESS IN "Y".
```

```
0176 07C4 3510          MSG     PULS X
0177 07C6 C607                  LDB #7
0178 07C8 A680          E@      LDA ,X+
0179 07CA 8A40                  ORA #$40
0180 07CC A7A0                  STA ,Y+
0181 07CE 5A                    DEC B
0182 07CF 26F7                  BNE E@
0183 07D1 6E84                  JMP 0,X

                        *  COPY 8 BYTES FROM X TO Y -
0184 07D3 C608          COPY    LDB #8
0185 07D5 A680          O@      LDA ,X+
0186 07D7 8A40                  ORA #$40
0187 07D9 A7A0                  STA ,Y+
0188 07DB 5A                    DEC B
0189 07DC 26F7                  BNE O@
0190 07DE 39                    RTS

                        *  CLEAR FOUR SPACES ON SCREEN
0191 07DF 8660          CLR4    LDA #$60
0192 07E1 A7A0                  STA ,Y+
0193 07E3 A7A0                  STA ,Y+
0194 07E5 A7A0                  STA ,Y+
0195 07E7 A7A0                  STA ,Y+
0196 07E9 39                    RTS
0197 07EA 2A54415045    BANNER  FCC /*TAPETYPE       /
0198 07FA 2054484520            FCC / THE MICRO WORKS/
0199 080A 4552524F52            FCC /ERROR STATUS =  /
0200 002F               ES      EQU $2F
0201 081A 2020202020            FCC /                /
0202 082A 5245434F52            FCC /RECORD TYPE =   /
0203 004E               RT      EQU $4E
0204 083A 2020202020            FCC /                /
0205 084A 46494C4520            FCC /FILE NAME =     /
0206 006C               FN      EQU $6C
0207 085A 2020202020            FCC /                /
0208 086A 46494C4520            FCC /FILE TYPE =     /
0209 008C               FT      EQU $8C
0210 087A 2020202020            FCC /                /
0211 088A 5452414E53            FCC /TRANSFER ADDRESS/
0212 089A 203D202020            FCC / =              /
0213 00B3               SA      EQU $B3
0214 08AA 4C4F414420            FCC /LOAD ADDRESS =  /
0215 00CF               LA      EQU $CF
0216 08BA 2020202020            FCC /       THRU     /
0217 00DC               EA      EQU $DC
0218 08CA 46494C4520            FCC /FILE LENGTH =   /
0219 00EE               FL      EQU $EE
0220 08DA 00                    FCB 0

0221 08DB                       END START
```

```
BADDR   007E  BANNER  07EA  BASFIL  06B9  BLEN    007D
BTYPE   007C  BUFFER  0500  CLR4    07DF  CLRIT   06F3
COPY    07D3  DATABL  0736  DATFIL  06C5  EA      00DC
EOFILE  074D  EOFLAG  0003  ES      002F  FILL    079B
FILLEN  0000  FL      00EE  FN      006C  FT      008C
HEADER  0683  HEXBYT  07AD  HEXNYB  07B7  HEXOUT  07A9
INITSC  0784  LA      00CF  MACFIL  06D1  MSG     07C4
NEXT    0706  READ    A70B  RECORD  0612  RT      004E
SA      00B3  SCREEN  0400  SFLAG   0002  SREAD   A701
STAD    0004  START   0600  SYNC    A77C  VLEN    0006
WLEN    077C
```

Color Computer News Magna-zine Service.

This New Device Will Give You A Three Weeks Vacation!!!

Well actually, the "vacation" is from the tedium of hand typing the programs published in **Color Computer News**. Even if you are a fairly good typist (i.e. you use *more* than two fingers, and you *don't* have to look at the keyboard!) it would take you about *twelve hours* to type in most of the programs in an average **Color Computer News** issue — and *then* you have to de-bug the programs on top of that! Save your "finger energy" for scratching your head while you think great thoughts and leave the program typing to the **CCN Magna-zine Service**. We guarantee that our monthly program tapes will save even the fastest typist many hours of frustration!! Relief for your tired fingers is just a **CLOAD** away!

Each month, CCN Magna-zine subscribers receive a top quality digital cassette which contains about a half dozen programs from their favorite CC-80 magazine, **Color Computer News**. American and Canadian subscriptions are available for just $42.00 (plus $6.00 first class postage) for a full 12 issues and can start with any issue number you specify. Single issues are also available for the low price of just $6.00 each plus $1.00 postage. Subscription postage to all other countries is $15.00 per year (sent via AO Air Mail). Overseas single issue postage is $2.00 per tape. (Florida residents add $.30 sales tax for single tape purchases *only*.)

The **CCN Magna-zine Service** is staffed by people who are highly qualified in cassette tape mastering and production and who use only top quality, custom loaded, all American made digital cassettes. Each tape is *fully* guaranteed for one full year against *any and* ***all*** hazards — up to and including the tape being crushed by a falling meteor!! Just return the original tape (or at least the piece with our label on it!) along with $1.00 for return postage, and that issue will be instantly replaced — no questions asked! Who else offers you such a guarantee???

To start your own subscription to the **CCN Magna-zine**, just fill out the coupon (a photo copy or a plain piece of paper with the proper information is just fine!) and mail it to: **CCN Magna-zine Service**, Box 68, Safety Harbor, Florida 33572. Include your check (personal checks are OK) or money order and be sure to indicate which **Color Computer News** issue you want your subscription to begin with if it is anything other than the next as yet unpublished issue number.

You already *know* about the high quality programming articles that have set **Color Computer News** apart from all other computer magazines, therefore, you *also* know what to expect from our cassette tape version!!! So, don't delay any longer — send in for your own subscription today! Spend your time *computing*, ***NOT*** *typing!!!*

YES! *Sign me up for a one year's subscription to the CCN Magna-zine! Enclosed is my check/money order for the full amount (including postage) of $48.00 (domestic and Canada) or $57.00 (overseas).*

NAME _____

STREET ADDRESS _____ *APT. #* _____

CITY _____ *STATE* _____ *ZIP* _____

Begin with issue number _____ ***instead of the next regular issue.***

CCN Magna-zine Service ~ Box 68 ~ Safety Harbor, FL 33572

I'd like to dedicate this book Color Computer News The Best of 1981 to all of the subscribers that believed in our project in the first half of 1981 and also to the advertisers that "came aboard" in our first four issues. To thank these advertisers we have placed a small copy of their current ad in this book.

While going through the old issues I was reminded of my philosophy for this medium. My cause was to find a method for Color Computer owners to exchange ideas and techinques with each other and to help one another with the problems that occur in the pursuit of their hobby. The fact that the medium of exchange is paper is inconsequential. I think we've (you and I) have done our job well.

The only sign of life is growth and I hope you'll see in these pages that CCN is very much "Alive and Well".