

\$2.50

# Color Computer News



JULY/AUGUST 1981  
Volume 1 Number 2

REMARKS	3
Bill Sias	
LETTERS	6
You	
HIGH RES GRAPHICS	17
Tom Rosenbaum	
CASSETTE FILES	24
Richard White	
SPACE BOREDOM	26
Andrew Hubbel	
RANDOM THOUGHTS	27
Tom Garcia	
MORSE CODE	28
James Haan	
32K RAM UPGRADE	30
Bob Lentz	
PRIME NUMBERS	32
David Bodnar	



INTRODUCING  
**SAMUEL R. BINK**

Color Computer is a trademark of the Tandy Corporation.  
Color Computer News is published bi-monthly by REMarkable Software.  
Copyright (c) 1981 by REMarkable Software



# OUR FEARLESS CREW!



## REMARKS

I found that I was unintentionally given two pieces of bad advice about the last issue. The first was from a postal employee about not sending the magazine bulk rate. If you'd like to see how bad that advice was look at the stamp on the back of last issue, 52 cents each!! Multiply that by over 1000 and you'll see why my knees got weak. I changed printers at the last minute due to a quality and delivery problem. I found some typos too late to change them. I reread the survey and it sounded like I was going to throw away your surveys without reading them. The index had two typos alone. How can anyone misspell "how"? Most of all I worried about not getting enough articles to fill up this issue. Even with all of the problems I was proud of the first issue and with your help we'll get better as time goes on.

In order to keep the small amount of hair that I have left I'm making the following changes.

First, the magazine will be mailed bulk rate. If you need first class delivery it's an extra \$.50 per issue to cover the higher rate and the extra handling.

Second, White Enterprises will be doing the master copy. White Enterprises is remotely affiliated with REMarkable Software, and sells some excellent educational and business software for all of the TRS-80s. You can contact White Enterprises by writing to: White Enterprises, 432 Rutledge, Pentwater MI 49449. The reason another software house is doing the typesetting is that their printer can justify proportionally. That is, if you look at the articles in the last issue, the right margin was made even by adding spaces between words, proportional justification adds tiny spaces between letters. The difference is dramatic. Another thing about proportional justification is that it will allow more letters per line. This will perhaps make the magazine look smaller when it isn't. We tried a few pages and the difference is about 6 lines per page, extended over the entire magazine that could amount to a page or two each issue.

Third, deadlines will be enforced for everyone. The magazine gets mailed on the 24th of the month preceding the cover date. Therefore deadline is one month prior to the publication date. This is much shorter than most magazines. But I think we can live with this for a long time.

What do you think about going monthly? This would mean not only more regularity but the disadvantage of a smaller magazine. I'm prepared to go to press monthly but does greater frequency outweigh the disadvantages of a smaller magazine and higher cost per year? You have to decide. If we change to monthly it also means that I have to depend on you for more articles, but the quality must stay the same. I'll judge by the number of letters in favor plus the number of articles received.

## REMARKS

The classified ad section is still unused, perhaps because I haven't pushed it enough. The prices are \$5.00 per half inch for personal listings and \$15.00 for business listings. This is a split page so judge your listing accordingly.

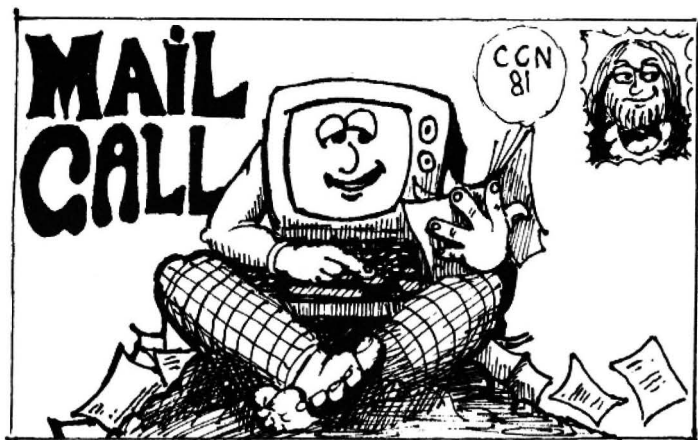
You should be aware that, because the way the Color Computer hardware is configured, piqqy-backing RAM chips to upgrade to 32K is incompatible with graphics. The Color Computer will not move the video display into the new bank without additional modification. The Micro Works has contributed an article about how to alleviate this problem for those of you that learned the hard way or feel you need 32K. But, please read the article carefully before you start cutting traces.

I just saw Radio Shack's new CC manual. The thing is pretty good. They actually show you how to get into hi-res, use the cassette and all of the other "stuff" that the earlier manual left out. But I was really quite upset by the section about hi-res, they take great pains to tell you that it's for "technical types" and that if you make a "typo" you may have to reset the computer. First, I'm not a "technical type" but I couldn't find anything there that I was afraid of. "You may have to reset the computer if you make an error", so what. It's not like the computer will blow up. If I have to reset my computer a million times to learn something new then I'll reset it a million times. Nothing irritates me more than to hear someone imply that anything is too hard for someone else to learn. I guess the clincher was the last paragraph on the first page where, after warning us about how hard all of this Machine language and direct memory addressing is, states, "Still with us? O.K. now that we've warned you....". Further more, to use hi-res you don't need any machine language just a few POKES from Basic. Direct memory addressing(?) I guess they mean that the POKE must address a memory location directly (I didn't know it could work any other way). Still there's enough there that you should get one if you can. The service manual is also available now for \$8.00. Your local Radio store may think that this is a controlled circulation item, if they do point out that it is in their latest catalog.

Reader service cards are really good things if you keep in mind the way magazines deal with them. I've gotten quite a few calls from people wanting to know why I haven't answered their request from the 80 Microcomputing reader service card. The reason is that I haven't got them yet. The usual routine for "binco cards" is the publisher waits for all of the cards to come in (at least 4 weeks after the publication date of the magazine), they are then entered into their (or a service bureau's) computer and sorted by advertiser. Then they print sheets of mailing labels for all the advertisers, stuff them in envelopes and mail them out bulk rate. Total elapsed time is 6 to 10 weeks from the date you mailed it in depending on which magazine and the volume of responses. We then must enter the names into our computer and sort out all of the duplicates, printout the new names, stuff all the envelopes and mail our reply. Our total time for reader service cards is about 2 days. Compare this with the people that wrote directly, they got the magazine before I got the reader service cards. I like reader service cards and will continue to use them, but if you need or want a fast reply you have to write, not just to us but everyone. One last item on response, many of you didn't get personal replies to your subscriptions and I'm really sorry but the last two weeks before it goes to the printer are busy.

I was amazed at the response to the survey by both the number of responses and the answers. I'm shocked at the number of people with Extended Basic, I really thought that they would be the minority. I was glad to see that everyone thought we are off to a good start. The format has changed a little due to your responses. The change to two columns per page was mixed but most everyone wanted the margin to stay wide so they could put the magazine in a three ring binder.





**HELP!**

I'm looking for a POKE statement that QUICKLY turns on the cassette motor (MOTOR ON has a delay too long for my critical program). Have you found one yet?  
Kenneth Armstrong  
Chicago

Try POKE 65313,4 to turn it on and POKE 65313,52 to turn it off.

Can text from RAM be saved to tape from CompuServe?  
Bruce Gustafson  
Roscoe, IL

Perhaps you may be interested in some of the things I have found while wandering around the CC. Many of these things appeared in a letter to the editor which I wrote to Creative Computing and appeared in the March issue. All addresses are given in decimal. The current cursor position is stored in locations 136 and 137. The contents of these locations range from 1024 to 1536, which corresponds to the range of "normal" alpha video RAM addresses. Location 282 controls lower case/reverse video alpha mode. When this location is non-zero, one gets lower case (reverse video). It is normally set at 255 (\$FF) and when one hits SHIFT 0, it gets negated, thus changing it to zero. Since I frequently hit SHIFT 0 when I mean to hit SHIFT 9 for close parenthesis, I POKE a non-zero value into 282 (other than 255) which effectively disables SHIFT 0. The line number currently being executed while a program is running is stored at locations 104, 105. I haven't found a great deal of use for that one, but there it is! (I feel it will become more useful when I get some more machine language programs going).  
Alexander Frazer Jr  
Fort Lauderdale, FL

I have seen references to 32K RAM upgrades, which I cannot do myself due to bad eyesight. Do you or do you know of someone or a company that will do it for me,  
Donn Jones  
Canal Winchester, OH

Can anyone lend a hand?

```

0 CLEAR 500
10 CLS:L=0
20 READ AA$
25 IF LEN(AA$)/32>L THEN FOR I=0 TO 1000:NEXT L=L+1:CLS
30 IF AA$="END" THEN 9999
40 IF LEN(AA$)<32 THEN PRINT AA$:L=L+1
50 IF LEN(AA$)=32 THEN PRINT AA$;
60 IF LEN(AA$)<33 THEN 100
70 FOR I=32 TO 1 STEP -1
80 IF MID$(AA$,I,1)<>" " THEN 90
85 PRINT LEFT$(AA$,I); IF I<32 THEN PRINT
86 AA$=MID$(AA$,I+1)
87 GOTO 40
90 NEXT I
100 IF L<12 THEN 20
110 FOR I=0 TO 1000:NEXT
120 GOTO 10
200 DATA "REMarkable Software"
210 DATA "P.O. BOX 1192"
220 DATA "MUSKEGON, MI 49443"
230 DATA " "
240 DATA " "
250 DATA "DEAR BILL:"
260 DATA " "
270 DATA " "
280 DATA "HAVING READ YOUR FIRST ISSUE AND SENT IN MY SUBSCRIPTION
AND SURVEY, I WOULD ALSO LIKE TO CONTRIBUTE A PROGRAM TO THE USER'S
GROUP."
290 DATA "FOLLOWING THIS LETTER IS A GAME CALLED BLOCK THAT WILL RUN
IN 4K. I HOPE THAT THE CCN AND THE USER'S GROUP CONTINUES TO GROW
AND PROSPER."
310 DATA "                SINCERELY,"
330 DATA "                GREG R ESTEP"
350 DATA "                CORTE MADERA"
360 DATA "                CA    94925"
9990 DATA "END"

```

Appending (merging) programs from tape is accomplished by changing the values at addresses 25 and 26 to point to an address behind the Basic program in RAM. The new address should be the beginning of variable storage as indicated by the pointer at 27 and 28. After loading the second program the pointer at 25 and 26 must be restored to the start of Basic at 1531. The second program must have higher line numbers than the first. To merge the two programs type in the following basic statements in command mode (no line numbers):

```

1 CLOAD"first program"
2 PRINT PEEK (28)
3 POKE 25, PEEK(27)
4 POKE 26, PEEK(28) - 2
5 CLOAD"second program"
6 POKE 25, 6
7 POKE 26, 1

```

If PEEK(28) yields a value < 2 then steps 3 and 4 would be:

```

3 POKE 25, PEEK(27) - 1
4 POKE 26, PEEK(28) + 254

```

Robert Huxter  
Media, PA





## Color Computer Hi-Res Graphics By Tom Rosenbaum

One of the persistent mysteries of the Color Computer is the use of the high resolution graphics modes. As Extended Color Basic becomes more prevalent, more people will be exposed to the capabilities of high resolution but will be frustrated by the slowness of it. Virtually the only way to get around this limitation is to write programs in machine language, but in order to do that, one must learn how the graphics modes operate.

The display of the Color Computer may be tailored to any one of a number of different display modes. These modes are controlled by the MC6847 Video Display Generator (VDG). A summary of the VDG modes is contained in Table 1.

In order to access the various display modes of the Color Computer, it is necessary to program both the VDG and the Synchronous Address Multiplexer (SAM). The SAM provides data from the Random Access Memory (RAM) for the VDG to process and display. If the VDG and the SAM are not put into the same mode, a garbage display will be the result. This is because the VDG will be trying to process data for a particular display mode but the SAM will be providing data for a different display mode.

Before attempting to use the various display modes of the Color Computer, one must become aware of the manner in which the display information is stored in the computer. All of the video display data is memory mapped. Simply put, it means that the information seen on the display is stored in the memory of the computer as opposed to having its own special memory just for the video display. The normal display for BASIC programs requires 512 bytes and is located from 1024 to 1536. Try POKEing data into those addresses and see what happens. In most black and white display systems, such as the TRS 80 Model III, one bit in the display memory corresponds to one pixel (display element). If the bit is a one, the pixel is on (white); if the bit is a zero, the pixel is off (black). Some of the Color Computer modes use only two colors and store video display data as described above. The other modes use at least four colors - therefore, at least one pair of bits must be used to indicate the color of a pixel. All of the four color graphics modes divide a byte into four pairs of bits and each pair of bits defines the color for that pixel.

The 6847 VDG supports three basically different types of display modes:

1. ALPHANUMERIC
2. SEMIGRAPHIC
3. GRAPHIC

The alphanumeric mode allows alphanumeric characters to be displayed. Lowercase is displayed in inverse video - this is a built-in hardware feature of the 6847. The semigraphic modes partition the display area into a group of display blocks. Each block is subdivided into a number of graphic sub-blocks. Only one color may be specified for each block - all of the sub-blocks must be the same color as the color of the block. Each sub-block may be on (the color of the block) or off (black). The semi-4, semi-6 modes are coarse resolution which will generally not be used in programs. Semi-4 is used in BASICs SET and RESET functions (not PSET and PRESET). Semi-8, 12 and 24 use a complicated addressing method and are inefficient - 38 0/0 of the memory used in the video display is wasted. For these reasons you will generally not find much use for the semigraphic modes. Their only redeeming feature is that they allow all eight colors to be displayed at the same time.

The graphics modes (1C, 1R, 2C, 2R, 3C, 3R, 6C, 6R) are the most powerful and will be used by most high resolution programs. The 1R, 2R, 3R and 6R modes are basically black and white modes in which one bit of a display byte turns a pixel on (foreground color) or off (black). The 1C, 2C, 3C and 4C modes are color modes which each display byte is divided into four pairs of bits. Each pair of bits specifies one of four colors for the pixel it controls. The price which you pay for using the higher resolution graphics modes is having to use more memory for the video display. Table 2 summarizes the graphics display modes.

Color Computer Hi-Res Graphics  
By Tom Rosenbaum

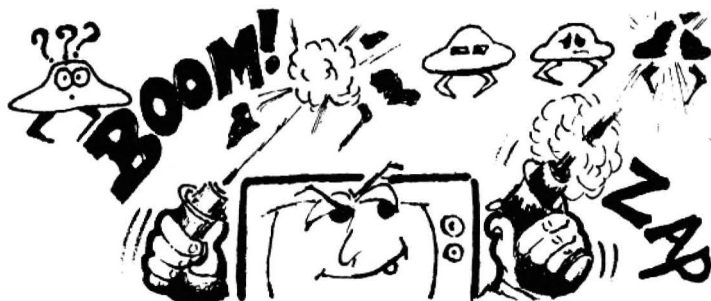
In order to better understand how to use the high resolution graphics modes, a sample program will be written. This program will use the highest resolution color graphics mode, 6C. This mode requires 6K of RAM and the video display will begin at \$400 (The dollar sign indicates that the number following is in hexadecimal format). This program will draw an invader from a Space Invaders program on the screen and move it around under the control of the right joystick. The program goes into a continuous loop in which it samples the joystick data, erases the old invader and draws a new invader at the new joystick horizontal and vertical coordinates. The invader is only allowed to occupy one of 32 horizontal and one of 64 vertical positions. These constraints are observed merely for ease of programming. The invader "shape" is a block of 16 bytes 2 wide by 8 deep - putting different values into this block of data will change the shape of the invader.


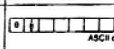

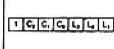

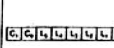
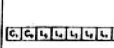

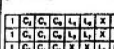
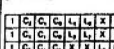

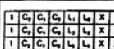
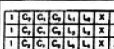

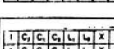
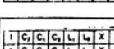




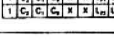
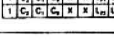

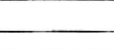
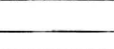

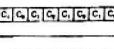
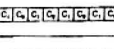

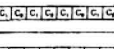
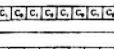

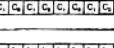
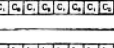






Typing "X" will escape the program and return to BASIC.

TABLE 1. MODE SELECT

MC6883 and VDG MODE REGISTERS			PIA REGISTER BITS HEX ADDRESS (FF22)								DATA BITS		ALPHA/GRAPHIC MODE SELECT
V <sub>2</sub>	V <sub>1</sub>	V <sub>0</sub>	7	6	5	4	3	2	1	0	7	6	
0	0	0	0	X	X	0	CSS	N	N	N	0	0	Alphanumerics
0	0	0	0	X	X	0	CSS	N	N	N	0	1	Alphanumerics Inverted
0	0	0	0	X	X	0	X	N	N	N	1	X	Semigraphics - 4
0	0	0	0	X	X	1	CSS	N	N	N	X	X	Semigraphics - 6
0	1	0	0	X	X	0	X	N	N	N	1	X	Semigraphics - 8
1	0	0	0	X	X	0	X	N	N	N	1	X	Semigraphics - 12
1	1	0	0	X	X	0	X	N	N	N	1	X	Semigraphics - 24
0	0	1	1	0	0	0	CSS	N	N	N	X	X	64 x 64 Color Graphics
0	0	1	1	0	0	1	CSS	N	N	N	X	X	128 x 64 Graphics
0	1	0	1	0	1	0	CSS	N	N	N	X	X	128 x 64 Color Graphics
0	1	1	1	0	1	1	CSS	N	N	N	X	X	128 x 96 Graphics
1	0	0	1	1	0	0	CSS	N	N	N	X	X	128 x 96 Color Graphics
1	0	1	1	1	0	1	CSS	N	N	N	X	X	128 x 192 Graphics
1	1	0	1	1	1	0	CSS	N	N	N	X	X	128 x 192 Color Graphics
1	1	0	1	1	1	1	CSS	N	N	N	X	X	256 x 192 Graphics

X DON'T CARE  
N DO NOT CHANGE



COLOR SET	DATA BIT #	Color			Resolution		Data Byte(s)	Comments	
		Character Color	Background	Border	Columns x Rows	Detail			
0	0	Green	Black	Black	32 x 16	8 dots 12 dots			The Alphanumeric Internal mode uses an internal character generator which occupies the following bus dot by seven dot characters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ( ) _ - S P ' # \$ % & * + = / 0 1 2 3 4 5 6 7 8 9 : ; = ?
1	0	Orange	Black	Black					
	1	Black	Green	Black	64 x 32		one element		The Semigraphics-4 mode uses an internal "coarse graphics" generator in which a rectangle (eight dots by 12 dots) is divided into four equal parts. The luminance of each part is determined by a seven dot character. Color is determined by three bits. It requires 512 bytes of display memory.
X	X	Lx Cx C1 C0 Color 0 0 0 0 Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange							
	0	Black	Green	Black	64 x 48				The Semigraphics-6 mode is similar to the Semigraphics-4 mode with the following difference: The eight dot by twelve dot rectangle is divided into six equal parts. Color is determined by the two remaining bits. It requires 512 bytes of display memory.
X	X	Lx C1 C0 Color 0 0 0 0 Black 1 0 0 0 Green 1 0 1 1 Red							
	1	Black	Green	Black	64 x 64				The Semigraphics-8 mode requires four column consecutive addresses, and produces a 2x4 block. It requires 2048 bytes of display memory.
X	X	Lx C1 C1 C0 Color 0 0 0 0 Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange							
	X	Black	Green	Black	64 x 96				The Semigraphics-12 mode requires six column consecutive addresses, and produces a 2x6 block. It requires 3072 bytes of display memory.
X	X	Lx C1 C1 C0 Color 0 0 0 0 Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange							
	X	Black	Green	Black	64 x 192				The Semigraphics-24 mode requires twelve column consecutive addresses, and produces a 2x12 block. It requires 6144 bytes of display memory.
X	X	Lx C1 C1 C0 Color 0 0 0 0 Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange							
	0	Green	Green	Green	64 x 64				The Graphics-1C mode uses 1024 bytes of display RAM in which one pair of bits specifies one picture element.
1	X	Black	Green	Black					
	0	Black	Green	Black	128 x 64				The Graphics-1R mode uses 1024 bytes of display RAM in which one bit specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	128 x 64				The Graphics-2C mode uses 2048 bytes of display RAM in which one pair of bits specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	128 x 96				The Graphics-2R mode uses 1536 bytes of display RAM in which one bit specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	128 x 96				The Graphics-3C mode uses 3072 bytes of display RAM in which one pair of bits specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	128 x 192				The Graphics-3R mode uses 3072 bytes of display RAM in which one bit specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	128 x 192				The Graphics-4C mode uses 6144 bytes of display RAM in which one pair of bits specifies one picture element.
1	X	Black	Green	Black					
	0	Green	Green	Green	256 x 192				The Graphics-6R mode uses 6144 bytes of display RAM in which one bit specifies one picture element.
1	X	Black	Green	Black					

\*\*SAMPLE PROGRAM TO DEMONSTRATE HIGH  
 \*\*RESOLUTION GRAPHICS

A00A JOYST EQU \$A00A  
 A000 KEY EQU \$A000 KEYBOARD DRIVER ENTRY  
 0400 VIDRAM EQU \$400 TOP OF VIDEO RAM

1C00 ORG VIDRAM+\$1800 PUT PROGRAM ON TOP OF VIDEO  
 \* DISPLAY

\*\*SET UP 6C GRAPHICS MODE

1C00 86 E0 LDA #\$E0  
 1C02 B7 FF22 STA \$FF22 PROGRAM THE VDG (6847)  
 1C05 B7 FFC3 STA \$FFC3  
 1C08 B7 FFC5 STA \$FFC5 PROGRAM THE SAM (74LS783)

\*\*CLEAR THE SCREEN

1C0B 4F CLRA  
 1C0C 5F CLRB  
 1C0D 8E 0400 LDX #VIDRAM START OF VIDEO SCREEN  
 1C10 ED 81 LOOP0 STD ,X++ CLEAR TWO BYTES  
 1C12 8C 1C00 CMPX #VIDRAM+\$1800 ARE WE AT BOTTOM OF SCREEN  
 1C15 25 F9 BLO LOOP0 NO

\*\*\$200 & \$201 ARE TEMP STORAGE LOCATIONS FOR  
 \*\*THE HORIZONTAL AND VERTICAL COORDINATES OF  
 \*\*THE INVADER

1C17 F7 0200 STB \$200 START THE INVADER AT THE UPPER  
 1C1A F7 0201 STB \$201 HAND CORNER OF THE SCREEN

\*\*MAIN PROGRAM WHICH MOVES THE INVADER

1C1D AD 9F A000 MAIN JSR [KEY] CHECK FOR THE "X" KEY  
 1C21 81 58 CMPA #X IS IT?  
 1C23 26 03 ENE LOOP1 NO  
 1C25 7E A027 JMP \$A027 YES; GO BACK TO BASIC  
 1C28 AD 9F A00A LOOP1 JSR [JOYST] GET THE JOYSTICK DATA  
 \*HORIZONTAL DATA TO \$15A  
 \*VERTICAL DATA TO \$15B

\*\*THIS CODE WILL ERASE THE INVADER

1C2C 8D 30 BSR CALCAD GET SCREEN ADDR OF INVADER  
 1C2E 86 08 LDA #8 INVADER OCCUPIES 8 VERT ROWS  
 1C30 6F 80 LOOP2 CLR ,X+ CLEAR THE FIRST ROW  
 1C32 6F 84 CLR ,X OF THE INVADER  
 1C34 30 88 1F LEAX @1,X MOVE TO THE NEXT ROW  
 1C37 4A DECA CHECKED ALL 8 ROWS?  
 1C38 26 F6 BNE LOOP2 NO

\*\*GET THE NEW INVADER ADDR AND STORE IT  
 \*\*IN THE TEMP STORAGE LOCATION

1C3A B6 015A LDA \$15A GET NEW HORIZONTAL ADDR  
 1C3D B7 0200 STA \$200 STORE IT  
 1C40 B6 015B LDA \$15B GET NEW VERTICAL ADDR  
 1C43 B7 0201 STA \$201 STORE IT  
 1C46 8D 16 BSR CALCAD GET ABSOLUTE SCREEN ADDR

```

*           OF THE NEW INVADER ADDR
1C48 33 8D 002C      LEAU  TABLE,PCR  GET ADDR OF INVADER DATA
1C4C 86 08          LDA   #8           8 INVADER ROWS TO MOVE
1C4E E6 C0          LDB   ,U+         GET ONE BYTE OF INVADER
1C50 E7 80          STB   ,X+         DISPLAY IT
1C52 E6 C0          LDB   ,U+         GET ANOTHER BYTE OF INVADER
1C54 E7 84          STB   ,X           DISPLAY IT
1C56 30 88 1F      LEAX  31,X        MOVE TO NEXT ROW
1C59 4A            DECA          MOVED ALL 8?
1C5A 26 F2        BNE   LOOP3      NO
1C5C 20 BF        BRA   MAIN       GO BACK TO MAIN LOOP

```

```

**THIS ROUTINE WILL TAKE AN X-COORDINATE (0-63)
**STORED IN $200 AND A Y-COORDINATE (0-63)
**STORED IN $201 AND CONVERT THEM INTO AN
**ABSOLUTE SCREEN ADDRESS FOR THE INVADER

```

```

1C5E B6 0201      CALCAD LDA  $201      GET VERTICAL ADDR
1C61 81 3D        CMPA  #63-2     ROW 61 IS THE LAST ROW THE INVADER
*                *      MAY OCCUPY WITHOUT HAVING PART OF
*                *      IT EXTEND INTO RAM ABOVE THE VIDEO
*                *      DISPLAY AREA
1C63 25 02        BLO   LOOP10    INVADER IS NOT AT BOTTOM
1C65 86 3D        LDA   #61      HERE IT IS AT THE BOTTOM
1C67 C6 60        LDB   #96      %6 BYTES FOR EACH VERTICAL
*                *      INVADER POSITION
1C69 3D          MUL          GET VERTICAL OFFSET OF INVADER
1C6A 1F 02        TFR   D,Y       STORE IT IN Y
1C6C F6 0200      LDB   $200     GET HORIZONTAL ADDR OF INVADER
1C6F 54          LSRB          DIVIDE BY TWO - THE INVADER CAN ONLY BE
*                *      IN ONE OF 32 HORIZONTAL POSITIONS
1C70 4F          CLRA          CLEAR HIGH ORDER BYTE OF D REG
1C71 30 AB        LEAX  D,Y       ADD HOR AND VER OFFSETS AND PUT THEM IN ;
1C73 30 89 0400  LEAX  VIDRAM,X   ADD IN THE START OF VIDEO DISPLAY
1C77 39          RTS

```

```

**THIS TABLE DEFINES THE SHAPE OF THE INVADER

```

```

1C78 02 00 0A 80  TABLE  FCB   2,0,$A,$80
1C7C 2A A0 A6 68  FCB   $2A,$A0,$A6,$68
1C80 2A A0 30 30  FCB   $2A,$A0,$30,$30
1C84 C0 0C 30 30  FCB   $C0,$C,$30,$30

```

END

Mary, Lisa, Kathy and Sue...

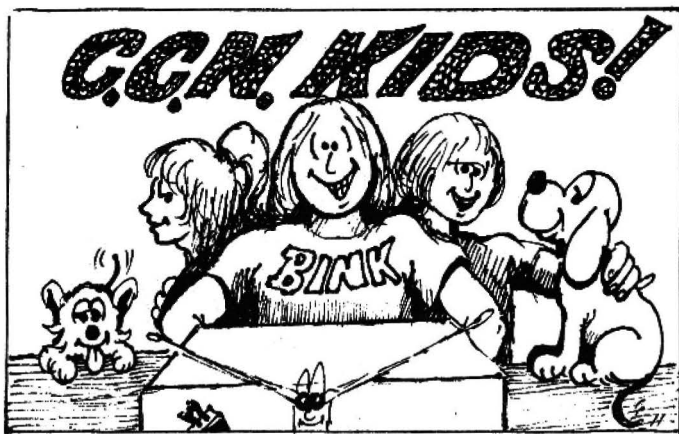
Where are you? We have had many fine articles and programs sent to us. Thanks for the terrific response! One thing does concern us, however. We have not had anything submitted by women. Why not? We need you. WE know there are many of you out there with color computers. We also know that you are creating your own programs. Let our readers know, too!

We are always looking for good articles. But we want to hear from the women as well as the men. It would also encourage other women to "get into" computers if they knew what other women, like themselves, are doing.

So...let us know. We'll be waiting to hear from you.



Several people asked when I was going to run a mod for higher clock speed, I doubt that I ever will. To make the Color Computer run faster just type POKE 65495,0 and to slow it back down POKE 65494,0. DON'T use high speed during I/O.



Here it is, the very first KID'S PAGE and it's all for you!! This page will be written by and for kids age 12 and under. We need your questions, comments, stories, drawings, programs and what-ers. The only exception to the age rule will be the editor, who is an ancient 29 year old and the publisher, who is extremely young and 27 (they're both a couple of kids anyway).

To get started we have a program written by a five year old girl (now seven) and two short ones by one of the old kids.

We hope you like it and will start sending us some really neat things you have done. Please send your questions and drawings, too. We want this page to be just what YOU want it to be. Next issue's best program will receive a SAM Coloring Book.

5 REM BY HEATHER SIAS

10 CLS

20 A\$=" THE DOG CHASED THE CAT"

30 FOR X=0 TO 450

40 PRINTGX,A\$

50 NEXT X

60 GOTO 30

10 INPUT"WHAT IS YOUR NAME" ;A\$

20 FOR X=LEN(A\$) TO 1 STEP -1

30 B\$=B\$+MID\$(A\$,X,1)

40 NEXT X

50 PRINT"HELLO " ;B\$



## 6809 Machine Code

Last issue we looked at addressing and most of the opcodes. This month we'll back off a little and just build and/or look at some tools.

To write machine code you need an understanding of the MPU and a way to get the machine code into the machine. We'll learn more and more about the MPU as we go along but this month let's look at how to get the machine code in.

You could build a house with just a hammer, or fix your car with just a wrench, but it wouldn't be fun or efficient. The same is true of writing machine code. You could write the code with just the POKE statement in Basic, but I wouldn't care to. Common tools for machine language programming are: Editors, Assemblers, Monitors, Single Steppers, Disassemblers, and a few other odd ball programs. We'll look at the functions each of these perform and then write a tool or two.

Editor/Assembler is the most common tool. The editor is used to prepare what is called a Source program, which is our program written in mnemonics. The assembler translates the source program into an Object program, or code that the computer can execute directly. A monitor is a program that allows you to examine memory and change it. A good monitor also allows you to read and write tapes or disk files containing a machine language program. A disassembler is the opposite of an assembler in that it changes Object code into Source code. Single Steppers are programs that cause the machine code program to run very slowly so that you can see exactly what is happening. Good single steppers disassemble as they step.

That was fast. With the exception of the Editor and the Assembler, all of the programs mentioned above are "debugging" tools. If I had to choose one tool out of all of the above I guess I would choose the Monitor, so I guess we can call the Monitor a wrench.

The biggest problem with the Color Computer right now is the completely empty tool box. I don't wish to leave anyone out but as of right now I am aware of the following tools for the CC:

Monitors	The Micro Works
	Computer Ware
	Color Computer News
	Data Soft, Inc
Disassemblers	Soft Sector
	Marketing
	The Micro Works
	Data Soft, Inc
Editor/Assemblers	Computer Ware
	The Micro Works
	Data Soft, Inc
Single Steppers	Data Soft, Inc

(for the sake of accuracy, Data Soft produces a monitor called Sigmon that does all of the above, they are not separate programs)

If you have any experience with any of the above we could use a review of each of them. Before anyone gets upset with me, I know there must be others out there and if any of you vendors think I left you out let me know.

Believe it or not this issue's main project is written in Basic. It's an almost full featured Monitor. I'll give you the documentation for it and then the Code itself for you to type in. Don't knock it too much, I wrote it all one Saturday morning. All the commands require the name of the command, a starting address and an ending address. The commands are EDIT, ASCII DUMP, HEX DUMP, SAVE, LOAD, BREAKPOINT, and EXECUTE.

The EDIT command allows you to modify memory and enter a hand assembled machine language program. ASCII Dump shows the ASCII equivalent of each memory location within the specified range. HEX dump shows the hexadecimal value at each memory location. SAVE will allow you to save the machine language program on cassette tape, note that the monitor must be used to reload the program. LOAD is used to enter the programs saved by the SAVE command back into memory. BREAKPOINT sets or resets breakpoints (forces the program to stop execution and return to the monitor). EXECUTE allows you to run your machine language program with some degree of control. The last command is BASIC. BASIC writes a Basic program which will POKE your program into memory and execute it without needing the monitor in memory.



## 6809 Machine Code

The program created by the BASIC command is useful for writing programs assisted by machine code. If you have Extended Basic you could change the BASIC subroutine to use the CLOADM command and delete the hex and decimal routines and replace them with &H and HEX\$.

Here it is! CCN'S first tool and the first entry in the contributed software library.

```

10 REM
20 REM COLOR COMPUTER BASIC MONITOR
30 REM CCN JULY/AUGUST 1981
40 REM BY BILL SIAB
50 REM
60 HEX$="0123456789ABCDEF": CLS
70 PRINT"  CCN BASIC MONITOR": PRINT
100 INPUT"COMMAND";C$,BA$,EA$
102 H$=BA$: GOSUB 10000: BA=D: H$=EA$: GOSUB 10000: EA=D
110 IF LEFT$(C$,2)="ED" THEN 1000
120 IF LEFT$(C$,2)="AS" THEN 2000
130 IF LEFT$(C$,2)="HE" THEN 3000
140 IF LEFT$(C$,2)="SA" THEN 4000
150 IF LEFT$(C$,2)="LO" THEN 5000
160 IF LEFT$(C$,2)="BR" THEN 6000
170 IF LEFT$(C$,2)="EX" THEN 7000
180 IF LEFT$(C$,2)="BA" THEN 8000
200 PRINT"UNIDENTIFIED COMMAND": FOR TIME=1 TO 100: NEXT
205 GOTO 70
999 REM EDIT MODE
1000 CLS: FOR ADDR=BA TO EA
1010 D=PEEK(ADDR): GOSUB 12000
1020 PRINT A$: INPUT H$: GOSUB 10000: BYTE=D
1030 POKE ADDR,BYTE
1040 NEXT ADDR
1050 GOTO 70
1999 REM ASCII DUMP
2000 CLS: FOR ADDR=BA TO EA STEP 15
2020 D=ADDR: GOSUB 11000: PRINT A$
2030 FOR OF=0 TO 14
2040 PRINT CHR$(PEEK(ADDR+OF));" ";
2050 NEXT OF: PRINT
2060 NEXT ADDR: GOTO 70
3000 CLS: FOR ADDR=BA TO EA STEP 4
3020 D=ADDR: GOSUB 11000: PRINT A$:
3030 FOR OF=0 TO 3
3035 D=PEEK(ADDR+OF)
3037 GOSUB 12000
3040 PRINT TAB((OF*5)+7)A$:
3050 NEXT OF
3055 PRINT
3060 NEXT ADDR
3070 FOR T=1 TO 200: NEXT: GOTO 70
3999 REM SAVE A MACHINE LANGUAGE TAPE (well sort of)
4000 CLS: INPUT"NAME";NA$
4010 OPEN"O",-1,NA$
4020 PRINT#-1,BA
4030 FOR ADDR=BA TO EA
4040 PRINT#-1,PEEK(ADDR)
4050 NEXT
4060 PRINT#-1,999
4070 CLOSE: GOTO 70
4999 REM LOAD A MACHINE LANGUAGE TAPE

```

## 6809 MACHINE CODE

```

5000 CLS: INPUT"NAME";NA#
5010 OPEN"I",-1,NA#
5020 INPUT#-1,BA
5030 INPUT#-1,BYTE
5035 IF BYTE=999 THEN CLOSE: GOTO 70
5040 POKE BA,BYTE
5050 BA=BA+1
5060 GOTO 5030
5999 REM SET OR RESET BREAKPOINT
6000 CLS: INPUT"SET OR RESET";Z#
6010 IF LEFT*(Z#,1)="R" THEN FOR OF=0 TO 2: POKE BP+OF,OLD(OF):
NEXT OF: GOTO 70
6020 INPUT"BREAKPOINT AT";H#: GOSUB 10000: BP=D
6030 FOR OF=0 TO 2: OLD(OF)=PEEK(BP+OF): NEXT OF
6035 POKE BP,14: POKE BP+1,180: POKE BP+2,244
6040 GOTO 70
6999 REM EXECUTE MACHINE LANGUAGE PROGRAM
7000 INPUT"ARGUMENT TO BE PASSED";AR
7005 INPUT"ENTRY POINT";H1#
7010 H#=H1#:GOSUB 10000
7020 DEFUSR(O)=D
7030 RV=USR(AR)
7035 PRINT"RETURNED VALUE =";RV
7040 GOTO 70
7999 REM WRITE BASIC TAPE
8000 CLS: INPUT"NAME";NA#
8002 OPEN"D",-1,NA#: A#="10 FOR X="+STR*(BA)+" TO "+STR*(EA)+"":
READ A: POKE X,A: NEXT X"
8010 PRINT#-1,A#: LN=20
8020 FOR ADDR=BA TO EA STEP 10
8030 A#=STR*(LN)+" DATA"
8040 FOR OF=0 TO 9: A#=A#+STR*(PEEK(ADDR+OF)):IF OF<9 THEN A#=A#
+", "
8045 NEXT OF
8050 PRINT#-1,A#: LN=LN+10: A#="": NEXT ADDR
8055 INPUT"ENTRY POINT";H#: GOSUB 10000: EP=D
8056 A#=STR*(LN+10)+" EXEC "+STR*(EP)
8057 PRINT#-1,A#
8060 CLOSE: GOTO 70
9900 DATA 1, 16, 256, 4096
9999 REM HEX TO DECIMAL SUBROUTINE
10000 D=0: RESTORE
10010 Z=LEN(H#)
10020 FOR K=Z TO 1 STEP -1
10030 READ M
10040 FOR J=1 TO 16
10050 IF MID*(H#,K,1)=MID*(HEX#,J,1) THEN Z=J-1: J=16
10060 NEXT J
10070 D=D+Z*M
10080 NEXT K
10090 RETURN
10999 REM DECIMAL TO HEX SUBROUTINE
11000 A#="": H4=INT(D/4096)
11010 H3=INT((D-H4*4096)/256)
11020 H2=INT((D-((H4*4096)+(H3*256)))/16)
11030 H1=D-((H4*4096)+(H3*256)+(H2*16))
11035 A#=MID*(HEX#,H4+1,1)+MID*(HEX#,H3+1,1)+MID*(HEX#,H2+1,1)+M
ID*(HEX#,H1+1,1)

```

## 6809 MACHINE CODE

```
11040 RETURN
11999 REM 2 DIGIT DECIMAL TO HEX
12000 A$="": H2=INT(D/16)
12010 H1=D-H2*16
12015 A$=MID$(HEX$,H2+1,1)+MID$(HEX$,H1+1,1)
12020 RETURN
```

If you have another computer you can use it there also, it doesn't care what MPU it operates on, so it should work with any Basic that has PEEK, POKE and MID\$, with the exception of the BASIC command which will work only with disk or other ascii storage device (like CC's cassette) and the BREAKPOINT, the only change needed in breakpoint is to change the codes to whatever it is on the other machine you use. If you only have 4K you should remove all REM lines, shorten the variable names to 2 letters (in fact, everyone should, I just like flashy variables) and combine lines wherever possible. The 4 and 16K versions are available from the CCN library for \$7.95, which also saves the chore of typing it in.

If the phone calls I've been getting are any indication I would say that the next thing to discuss is how you read or use the information given in an article about machine language programming. The explanation is both simple and complex. A good assembler produces a six column display, the first column contains addresses, the next contains op-codes, followed by the labels used by the programmer, next is the mnemonic column, then the operand column and last the comment column. If you have an assembler you would just type in the label, mnemonic, operand and the comment columns and let the assembler do the work. If you have a monitor you would edit memory at the locations shown in the address column to the values in the opcode column. Either method will put the machine language program into memory or on cassette tape. The next step is to read the comments and try to figure what the program is to do and last you need to debug the program. Debugging machine language programs will force you to learn more than any amount of reading or even programming on your own. The way to do this is to mentally divide the program into small pieces and test each of these pieces alone. This is done with the BREAKPOINT command in our Basic monitor or as detailed in the manual for the other monitors on the market. Set a breakpoint at the end point of the part of the program you are testing and execute the program, at the breakpoint the monitor will return control to you and you may modify the program as necessary by either correcting misentries or inserting new code as needed. As we progress and learn more about the 6809 all of these problems will fade. The disaster now is that not only must you learn a new "language" you also have to become a software mechanic and try to learn all of the new jargon all at once. We all went through it at one point or another and if you stick with it you'll master it all.

The next tool is perhaps the easiest to write, if you don't care about speed or efficiency. Disassemblers are handy programs and extremely simple to write in Basic, but very slow in operation. There are a lot of approaches to disassemblers. The fastest Basic version would be just a giant program full of IF/THEN statements. The slowest but more memory efficient would be a READ/DATA type of program, i.e. put all of the mnemonics in DATA statements in order and have a PEEK statement followed by a FOR/NEXT loop from 0 to the value that you PEEKed and READ one item with each execution of the loop. When you fall out of the loop you have the correct mnemonic, you then RESTORE and PEEK again. The fastest version would be to take the chart on the next few pages and figure out the algorithm used. I'm going to save my Basic Disassembler for next issue and let's have a contest. Who can write the fastest Basic Disassembler?



Cassette Files  
by Richard A. White

The Basic commands for cassette file use are in the 4K Basic, but nowhere do the manuals tell you how to use them. These commands are the same as those in Model I and Model III disk basic for sequential file. It took me a while and more than a few bucks worth of books and magazines to figure this out. Cassette operation with previous machines has been a pain at best so interest in and published literature on cassette file systems is minimal. The Color Computer saves and loads reliably at 1500 rather than unreliably at 500 baud as with Model I so using sequential filing techniques on tape may return. In addition the Color Computer is not particularly fussy about which tape you use. I have quite a few programs on Radio Shack's 3 for \$1.99 C-30 cassettes, but you need to adjust the volume control when going from one tape type to another. Computer quality tape is not necessary.

The key to loading sequential files to cassette is to write your program so the numerical data or strings are in subscripted variables and then PRINT#1 these using a FOR-NEXT loop. The file is loaded back into the computer in the same way using INPUT#-1 or LINE INPUT#1 if you have Extended Basic.

Listing 1 is a file save and load subroutine taken from a program I rewrote from a magazine article. The basic program which had been written on an HP 3000, hardly a personal computer, needed considerable compression to fit the Color Computer, but worked exactly as it did on the big machine when I was done. Where has our toy gone? Anyway, in this program points are stored as their subscripted X and Y values and lines are identified by their start point L1(K) and end point L2(K). In the listing we start by entering whether we want to store the file "O" for out or load the file "I" for in. Line 1010 lets us set up the recorder at the proper place on the tape and set it to play or record. Finally, the file name is entered and the Color Computer does the rest using FOR/NEXT loops. The program is dimensioned to handle 100 points and 30 lines. If these numbers of points and lines are not defined, zeros are filed and loaded back. The end of file EOF is not really needed but is shown to illustrate its use. Line 1080 is a memory saving trick. Where you exit a FOR-TO loop early, it is good to set your count variable at or above the highest loop value and then use a NEXT. This cancels the count out of memory freeing space that otherwise would be held open looking for the count to continue.

The basic subroutine in Listing 1 can be adopted by changing the number and names of the variables and final values in the FOR-TO loops. The listing shows numerical variables, but string variables work as well.

Subroutine to Save or Load a Cassette File.

```
1000 INPUT "TO SAVE A FILE ENTER 0,  
TO LOAD ENTER 1";E$  
1010 INPUT "SET RECORDER TO RECORD  
OR PLAY. PRESS ENTER ONCE FOR  
MOTOR ON THEN AGAIN FOR MOTOR  
OFF";I1: AUDIO ON: MOTOR ON: INPUT  
I1: MOTOR OFF  
1020 INPUT "ENTER FILE NAME 8  
CHARACTERS OR LESS";NA$  
1030 OPEN E$, -1, NA$: IF E$="I" THEN  
1060  
1040 FOR K=1 TO 100: PRINT#-1, X(K),  
Y(K): NEXT  
1050 FOR K=1 TO 30: PRINT#-1, L1(K),  
L2(K): NEXT: GOTO 1090  
1060 FOR K=1 TO 100: INPUT#-1, X(K),  
Y(K): IF -1=EOF(-1) THEN 1080 ELSE  
NEXT  
1070 FOR K=1 TO 30: INPUT#-1, L1(K),  
L2(K): IF -1=EOF(-1) THEN 1080 ELSE  
NEXT  
1080 K=100: NEXT  
1090 CLOSE -1: GOTO XXXX
```

This permits operating the recorder to position tape.

This line saves data to tape.

This line also saves data to tape.

This line inputs the data from the tape. The EOF is probably not needed since we print the same number of statements as we input.

Close file. GOTO could be RETURN.

```

10 PMODE 3,1: SCREEN 1,1: PCLS
20 LINE(76,24)-(160,84),PSET,BF
30 LINE(160,84)-(180,116),PSET
40 LINE(76,84)-(53,116),PSET
50 LINE(60,124)-(176,124),PSET
60 LINE(53,116)-(60,124),PSET
70 LINE(180,116)-(176,124),PSET
80 LINE(80,88)-(152,88),PSET
90 LINE(68,116)-(164,116),PSET
100 LINE(80,88)-(68,116),PSET
110 LINE(152,88)-(164,116),PSET
120 FOR X=84 TO 152 STEP 8
130 FOR Y=92 TO 110 STEP 4
140 PSET(X,Y,2): NEXT Y,X
150 LINE(90,108)-(140,112),PSET,BF
160 CIRCLE(160,40),10
170 CIRCLE(77,40),10
180 LINE(67,41)-(50,64),PSET
190 LINE(70,82)-(50,64),PSET
200 LINE(74,50)-(64,64),PSET
210 LINE(76,76)-(64,64),PSET
220 CIRCLE(75,79),7
230 CIRCLE(71,86),3
240 LINE(170,40)-(182,62),PSET
250 LINE(182,62)-(166,80),PSET
260 LINE(162,54)-(168,62),PSET
270 LINE(168,62)-(160,72),PSET
280 CIRCLE(161,79),7
290 CIRCLE(165,86),3
300 LINE(80,28)-(156,80),PRESET,BF
310 CIRCLE(104,42),10
320 CIRCLE(120,40),10
330 CIRCLE(114,47),25,4,1,0,.40
340 CIRCLE(106,46),5
350 CIRCLE(122,44),5
354 CIRCLE(112,58),7,1,1,15,.50
356 LINE(108,57)-(112,48),PSET
360 PRINT(0,0),3,4
370 PRINT(74,52),3,4
380 PRINT(163,57),3,4
385 FOR X=1 TO 100
390 L$="L"+STR$(RND(255))
400 PLAY L$
420 CIRCLE(114,47),27,4,1,0,.30
430 PLAY CHR$(RND(7)+64)
440 CIRCLE(114,47),27,1,1,0,.30
450 NEXT
460 GOTO 385

```

## COLOR COMPUTER COMPUTERWARE® has it all!

### FUN & GAMES



**PAC ATTACK**  
Increase challenging  
puzzles game with great  
sound and action  
cassette ..... \$24.95  
disk ..... \$29.95  
See more games & products  
available!



**STARSHIP  
CHAMELON**  
Defend your starship &  
planet against Gabolator  
attacks of bombs, anti-  
matter & space mines. Fast  
action, graphics, & sound  
cassette ..... \$24.95  
disk ..... \$29.95



**EL DIABlero**  
You awaken alone in the  
middle of the desert. Your  
merchandise has been  
chopped! You face the  
evil El Diablero monster. Pure  
adventure!  
cassette ..... \$24.95  
disk ..... \$29.95

### PROGRAMMING TOOLS



**MACRO  
ASSEMBLER**  
Macro assembler 6800  
assemble with library files  
and cross reference  
program  
Disk Computer disk \$49.95  
FLEX™ disk \$60.00  
Cassette assembler disk  
inserter ..... \$24.95  
See a listing of 170  
other programs listed



**PASCAL**  
Object's compact title  
PASCAL for training  
structure programming  
includes compiler, Proce-  
interpreter, editor  
supervisor, & samples  
(Req. 32K)  
cassette ..... \$49.95  
disk ..... \$60.00



**HOME MONEY  
MANAGER**  
Cassette checkbook organizer  
with printed reports for  
deposits, expenses,  
transactions & Chart of  
Accounts ..... \$19.95



**ADDRESS  
FACTORY**  
Complete name and  
address mailing list with  
special code selection and  
sorts for labels  
cassette ..... \$17.95  
disk ..... \$22.95



**SCRUB WORD  
PROCESSOR**  
Complete word processor  
for program editor with  
editing, footings, right &  
left justification, centering,  
justification, tabs, and  
more!  
(Cassette editor also available) ..... \$49.95

TO ORDER:  
add shipping of  
\$2.00 plus \$2.00  
for Canada, USA  
& Mexico  
acceptance



call or write  
Box 688  
Encinitas, Ca. 92024  
(714) 438-3512

**3-D BRICKAWAY** © 1982 by BRITT MONK, cdp

Add a new dimension to your game!  
Fast action, machine language, 3D  
arcade game. High res graphics,  
realistic sounds. Fun to play!

Requires 16K, joysticks; sold on  
cassette.

only \$14.00 post paid!  
from BRITT MONK, CDP  
P.O. Box 802  
Elyria, Ohio 44036



Space Boredom  
by Andrew Hubbel

"You are at the helm of a Starship. Imagine that your video screen is a window looking into outer space. Your laser sights are in the center of this "viewscreen". Try to destroy as many enemy ships as possible before colliding with them." So starts the introduction to Quasar Commander.

Actually, however, the game resembles a shooting gallery more than a starship. After selecting among several pages of options (skill level, time or shots limit, etc.) you begin the actual game. Your targets consist initially of a large number of moving dots which, if you "manuver" skillfully grow into three different types of targets: Scouts, which are fairly well behaved and vaguely resemble some Space Invaders, are the most numerous and are worth three points each when shot, (under some options, however, you must destroy them to get to the other targets.) Battle Cruisers, which resemble conventional aircraft, are worth 15 points apiece but are extremely hard to destroy. (each must fill nearly one-quarter of the entire screen before your shots will destroy it.) If you let any of the enemy ships collide with you (generally it will be a Battle Cruiser which refuses to die) you lose 10 points.

I have several criticisms of this game. First, it is essentially black and white. (Actually it has dark purple detail on a pale yellow background with colored instructions and red flashes when a ship is destroyed.) While this does not necessarily make it a bad game, this is, after all the "Color" computer.

Second, the joystick controls are difficult to use. The left joystick controls speed and has the firing button while the right joystick controls horizontal and vertical motion. These are analog controls - the null position is simply an arbitrary point and the slightest movement of the joystick alters the controlled function. There is no tactile feedback and it is very difficult to co-ordinate both joysticks while watching the screen.

Another complaint is that the scoring system does not work as described in the instruction manual. When you reach 100 points your score is decreased by 100 and you are supposed to receive a bonus of 50 "minutes" or 15 shots depending on the limit option chosen. What actually happens is that your remaining time is set to 49 "minutes" or your remaining shots to 15, which, if you happen to be doing well, may actually be a penalty.

Perhaps the biggest criticism of this game, however, is that it does not realistically portray what is promised. There are no external reference points (i.e. stars) and no illusion of depth. Your gun sight never moves. Instead the entire background moves together. Some larger targets will occasionally move against the background, but it appears that one has lost control of their movement rather than that they are closer. All "distant" objects are the same size with no identifying characteristics or indication of relative distance. As they grow in size, changing shape and growing independantly, they still appear to be moving in only two dimensions rather than three. The result appears to be partially random, partially controlled shooting gallery, not a starship.

Quasar Commander seems to be grossly overpriced. A few fanatics may enjoy these games, but I suspect that most Color Computer owners, like me, would prefer something more realistic and entertaining, particularly if it is going to cost \$30 - \$40. I am certainly not going to waste any more money on Program Paks, at least until I actually see one that is worth playing (and paying for).

---

For those who get Extended Basic and find their 16K only prints 8487 on PRINT MEM, typing PCLEAR 1 will get 13095. The March issue of R/S TRS-80 Microcomputer News said we would have 14.5K. I spent an hour trying to ask R/S on the hot-line and never got through.

Ralph O.R. Schubert  
Tulsa, OK

Random Thoughts  
by Tom Garcia

There are three different Color Computers from Radio Shack. First, there is the basic (no pun intended) Plain Jane model with standard Color BASIC and 4K of RAM. At least it's called a 4K model. Enter a PRINT MEM command and you will find that you only have 2,343 bytes to work with. Where did the rest of the memory go? The computer is using some of "your" RAM for it's internal work. Well, you can write a lot of programs with 2.3K bytes. As a beginner I sort of like the limited memory. I think that it is helping me to become a better programmer when I have to write programs in such a way as to conserve my limited amount of memory. I haven't run out yet so I guess I'm doing OK.

Computer number two is the extended basic model upgraded to 16K of RAM. It might cost you a hundred dollars or so to hire out the modification work but you can do it yourself for about a third of the price. Here is what I suggest that you do: Go to (or should that be GOTO?) your friendly Radio Shack store and buy a copy of Part #26-3001/3002. That is the TRS-80 Color Computer Service Manual. While you are there you might also consider picking up part number 276-1574 which is an IC insertion and removal tool kit. The manual costs \$9.95 and the tool kit is \$6.95. Now go home and read the manual. You are bound to learn something new about your computer and if you are at all mechanically inclined the parts list and the schematic are "must have" information. Your 16K upgrade project will involve the info on page 14 which is disassembly and reassembly dope, plus page 65 for printed circuit board depiction. Ready to give it a try? Obtain 8 RAM chips, type 4116, to replace the factory installed 4K chips. There are two jumpers that have to be moved from the factory installed 4K position to the immediately adjacent 16K pin. You can find these by looking very carefully at the above mentioned page 65. Check between U4 and U8 for one of them. The other one is just to the right of the 40 pin U10 SAM chip. After you have replaced the eight chips and moved the two jumpers you should have a 16K machine. Well, not quite 16K. The computer will still use almost a K and a half of RAM for it's own purposes, such as video display generation. Oh, the 4116 chips are available from many sources and should cost no more than \$4 each. Don't let anyone sell you any jumpers. You would have needed them for the Model 1 but you won't need them for Color Computer modification.

Computer number three is the 16K Extended Color BASIC model. That's the one that runs \$200 more list price and you will have to decide (or probably already have if you are reading COLOR COMPUTER NEWS) if it's worth the extra money. From what I have read concerning Extended Color it is just not too exciting when I consider what I use a computer for. I use it for number manipulating, string handling, file keeping, and word processing. I don't play games or draw pictures. It may be that I don't know what I'm missing but only time will tell. I keep thinking that something else may come along that I would rather plug into the ROM expansion socket than the Radio Shack part. My local dealer wants over a hundred bucks for the part plus \$25 to install it. Gee, sure would like to have the renumber feature that is part of Extended Basic. So far, the big disadvantage to my lack of the 16K Extended Color option is that I can't run my friends programs on my computer. They can run mine OK but they use commands that I can't execute when I try to load and run their programs. That, plus the future availability of commercial programs that will require the Extended Color will probably force me to upgrade at some point down the road.

---

How can I get Extended Basic for less than \$99.00,  
Mark Lockwood

First of all you really can't get the Extended Basic for \$99.00. Radio Shack charges \$17.50 to install it, which makes Extended Basic cost \$116.50. But if you would like to get it for \$99.00 you could try ordering part number 26-3018 from your local Radio Shack.

The Incredible Shrinking Program  
by James Haan

About two years ago when I first got my Model 1 one of the first programs I wrote was a program to send morse code. The first version worked great and only used 14K of RAM. As an exercise in self discipline I decided that the next step was to make the program operate on a 4K Model 1. After several attempts I succeeded in making the program operate at the target memory size and amazingly enough you couldn't tell the difference between the two versions unless you listed them. In March of this year I got a Color Computer and in memory of the good old days I decided to rewrite the old program to run on the new machine. The first problem was to discover a method of turning the cassette on and off rapidly, the Z80 OUT command doesn't exist on the Color's 6809. After experimenting and reading, Bill and I found that memory location 65313 is connected to the cassette relay. Oddly enough only the numbers 4 and 52 will cause anything noticeable to happen. 4 turns the cassette on and 52 turns it off as you can see by looking at lines 8 and 9 in the listing below. The first version ran great in a 16K machine and I again saw the challenge of self discipline before me. It seems that a 4K Color Computer really only has a little over 2K to operate with after scratchpad and overhead. This was going to be a bigger challenge than before due to the fact that the Model 1 version was compressed to 2476 bytes and now I had to compress the compressed version to run in 2400 bytes of memory. By examining the elements of Morse code you will see that many of the characters are combinations of other characters and by using this information it is possible to send characters that the program never defines. For example, since the letter T is just a single Dah (or dash to non-hams) I was able to leave the variable A undefined and jump to the subroutine that sends Dahs and the subroutine would return after a single execution. If you examine the code closely you will see other characters that use this technique. One word of caution, the program uses very few spaces between commands so check your typing carefully before you decide that the listing is incorrect. The spacing that most listings contain were eliminated deliberately to conserve memory and the program should be typed in exactly the same way. Be careful, debugging a program with no spaces can be difficult.

```
1 CLEAR70
2 CLS:R=0:PRINT"TO SEND CQ ENTER 1":PRINT"TO ANSWER CQ 2":
PRINT"TO CONT QSO 3":PRINT"FOR PRACTICE 4":INPUT"FOR QSO 5":Y
3 INPUT"HOW FAST":B:A=350/B:IFY=1THEN7
4 IFY=4THEN7
5 INPUT"HIS CALL SIGN":Z$:IFY=2THEN7
6 INPUT"HIS NAME":N$:INPUT"HIS RST":R$
7 ON Y GOSUB26,28,36,18,36
8 POKE65313,4:FORN=1TOR:NEXT:POKE65313,52:FORN=1TOR:NEXT:RETURN
9 POKE65313,4:FORN=1TOR*3:NEXT:POKE65313,52:FORN=1TOR:NEXT:RETURN
10 S=LEN(C$):FORT=1TOS:FORN=1TOR*2:NEXTN
11 F$=MID$(C$,T,1):PRINTF$:IFF$=" "THEN15
12 U=ASC(F$):U=U-47:IFU<1THEN131
13 ON U GOSUB129,108,111,114,117,119,121,123,125,127,131,131,131,
131,131,131,131,50,53,55,57,59,61,63,65,67,69,71,73,75,78,81,
83,85,87,89,9,93,95,97,99,102,105
14 IFF$<>" "THEN15
15 FORN=1TOR*8:NEXT
16 NEXT T:IFR=1THEN2
17 RETURN
18 CLS
19 C$="" :FORQ=1TOS:K=RND(36):K=K+47:IFK>57THEN21
20 K$=CHR$(K):GOTO23
21 K=K+7:K$=CHR$(K):GOTO23
22 GOTO19
23 C$=C$+K$:NEXT
24 C$=C$+" ":GOSUB10
```



MORSE CODE LISTING CONTINUED

```

25 GOTO19
26 C$="CQ CQ CQ DE W89QHX W89QHX W89QHX K
27 R=1:GOTO10
28 FORK=1T03
29 C$=Z$+" ":GOSUB10
30 NEXT
31 C$="DE ":GOSUB10
32 FORK=1T03
33 C$="W89QHX ":GOSUB10
34 NEXT
35 R=1:C$="K":GOTO10
36 C$=Z$+" DE W89QHX":GOSUB10
37 IFN$="THEN40
38 C$=" RR FB "+N$+" ":GOSUB10
39 IFY=3THEN48
40 C$="YUOR RST IS ":GOSUB10
41 C$=R$+" "+R$:GOSUB10
42 C$=" MY QTH IS MUSKEGON,MICH MUSKEGON,
MICH. MY NAME IS JIM JIM. ":GOSUB10
43 IFN$="THEN45
44 C$="WELL "+N$:GOSUB10
45 C$=" IT IS TIME TO TURN IT BACK TO YOU "
GOSUB10
46 C$=" HOW DO YOU COPY ":GOSUB10
47 C$=Z$+" DE W89QHX K":R=1:GOTO10
48 PRINT:INPUT"IT IS YOUR TURN TO SEND,
TO SIGN OFF PRESS ENTER";L$
49 GOTO44
50 GOSUB8:GOTO9
53 GOSUB78:GOTO67
55 GOSUB78:GOTO78
57 GOSUB78
59 GOTO8
61 GOSUB67:GOTO78
63 GOSUB9:GOTO78
65 GOSUB67
67 GOSUB8:GOTO8
69 GOSUB50:GOTO75
71 GOSUB78:GOTO9
73 GOSUB50:GOTO67
75 GOSUB9:GOTO9
78 GOSUB9:GOTO8
81 GOSUB75:GOTO9
83 GOSUB50:GOTO78
85 GOSUB75:GOTO50
87 GOSUB50:GOTO8
89 GOSUB67:GOTO8
93 GOSUB67:GOTO9
95 GOSUB67:GOTO50
97 GOSUB50:GOTO9
99 GOSUB78:GOTO50
102 GOSUB78:GOTO75
105 GOSUB75:GOTO67
108 GOSUB97:GOTO75
111 GOSUB67:GOTO81
114 GOSUB89:GOTO75
117 GOSUB65:GOTO9
119 GOSUB65:GOTO8
121 GOSUB9:GOTO65
123 GOSUB75:GOTO89
125 GOSUB81:GOTO67
127 GOSUB81:GOTO78
129 GOSUB81:GOTO75
131 IFF$<>". "THEN134
132 GOSUB63:GOTO97
134 IFF$<>". "THENRETURN
135 GOSUB87:GOTO71

```



## 32K RAM for the Color Computer

by Bob Lentz

1. To provide an additional 16K of RAM to a 16K Color Computer, obtain eight more 4116 dynamic RAM chips. These may be "piggy-backed" by placing them over the existing chips and soldering all pins (except Pin-4) onto the corresponding pins of the chip below. Pin-4 should be bent up out of the way to connect in the next step. See figure 1 for a sample piggy-backed chip.

2. Connect together the eight Pin-4s which were left unconnected in the step above. Connect these through a 33 Ohm resistor to Pin-35 of the 6883 SAM chip (U10). See figure 2 for a diagram of the completed modification.

At this point, your computer has 32K of RAM; you may put it back together and try it. (Note: the 4/16K jumpers on the PC board should be left in the 16K position, and no software changes are necessary). Try typing PRINT MEM. The exact value of MEM depends on how much RAM your BASIC takes up, but in any case it should be over 24000.

If you wish to be able to put the video display into the upper bank of RAM, you will need to continue with steps three and four below. This would be necessary if, for example, you wanted to declare more than 16K worth of Hi-Res screens under Extended Basic. For most applications the following steps are not necessary.

3. Remove the PC board from the box, and remove the ground shields to expose the bottom of the board. Cut the traces shown in figure 3. Also cut the trace on the top of the board which connects to Pin-11 of the 74LS273 (U6). See figure 4 for a diagram of this.

4. Refer to the before-and-after figures (5 and 6). Wire the circuit shown in figure 6 using the unused sections of the 74LS273 to latch the display data from either of the two RAS lines.

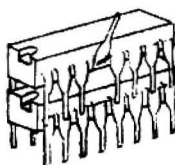


FIGURE 1.

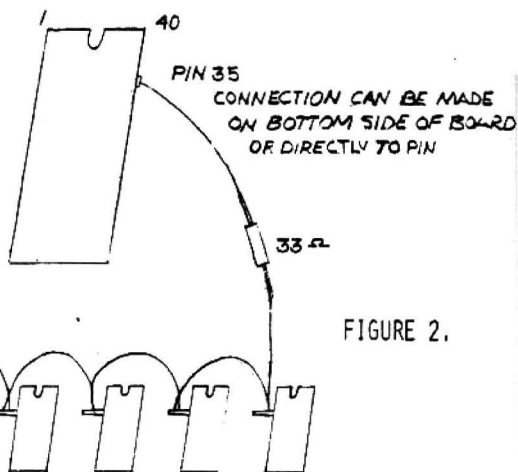
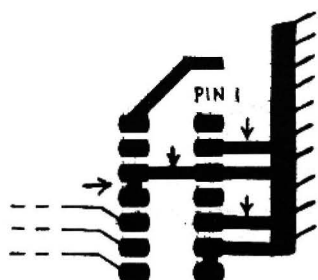


FIGURE 2.



U29 - CIRCUIT SIDE  
ARROWS INDICATE TRACE CUTS

FIGURE 3.

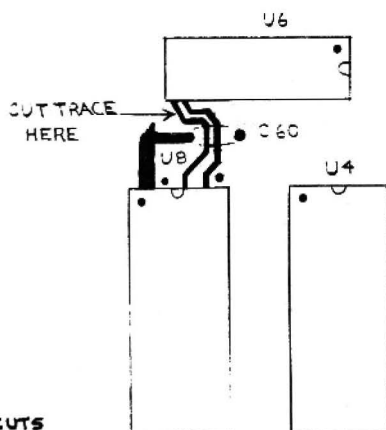


FIGURE 4.

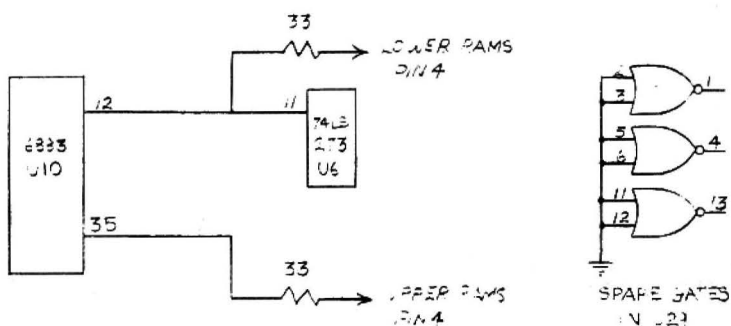


FIGURE 5.

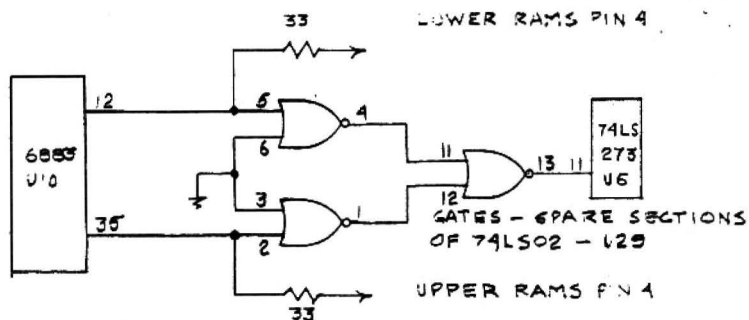


FIGURE 6.

Prime Number Generator  
by David Bodnar

Prime numbers have always held a magical fascination for me and for many of my 6th grade math students. Ever since pocket calculators made long division no more difficult than simple addition I have challenged my students to find 4, 5 and 6 digit prime numbers.

We test numbers for primeness\* by dividing the number by all of the primes up to the number's square root. If none of these division problems come out evenly then the number is prime.

I developed the following program to generate a list of primes. It is based on the test procedures outlined above and it is the fastest prime number generator I have seen.

One unique feature of the program is the array A(N). It is an internal list of prime numbers that is updated each time a new prime number is found.

The program is written in non-Extended Basic and requires 16K to generate a fairly large list of primes. If you have Extended Basic you must PCLEAR 1 before running to free enough memory to dimension array A(N). You could also change line 140 to IF X / A(N) >= SQR(X) THEN NEXT X since the Extended Basic has a square root function.

I recommend that you remove all REMarks and spaces and renumber by 10s to allow maximum memory.

\* I make up words too!

```
5 PCLEAR 1 'OPTIONAL OF EXTENDED BASIC
10 REM GENERATES A LIST OF PRIMES A(N) IS THE LIST
30 REM THIS DIM A IS ALL 16K MEMORY CAN HOLD
40 DIM A(2400)
50 I=2400*2400
60 CLS
70 A(1)=3
80 PRINT 2;
90 FOR X=3 TO I STEP 2
100 'N IS COUNTER FOR TEST PRIME ARRAY
110 N=0
120 N=N+1
130 IF X=A(N) THEN 180;'IF X=3 THEN PRINT AND ADD TO ARRAY
140 IF X/A(N)-INT(X/A(N))=0 THEN NEXT X: 'DOES X DIVIDED BY TEST PRIME
COME OUT EVEN?
150 IF A(N)>X/A(N) THEN 180;'IS TEST PRIME GREATER THAN SQR(X)?
160 GOTO 120: 'RETURN FOR NEXT LARGER DIVISOR
170 IF W=2400 THEN 220: 'IF UP TO LIMIT END
180 W=W+1: 'COUNTS THE PRIMES
190 A(W)=X: 'PUT NEW PRIME INTO ARRAY OF PRIMES
200 PRINT X;
210 NEXT X
220 END
230 D. BODNAR- 4-29-81
```

---

Were you aware that Radio Shack will send at no charge a write-up called "High Resolution Graphics Using Color Basic". Call the Hot line at (800) 433-1679.  
Thomas Ernst  
and  
Dennis Hay-Chapman