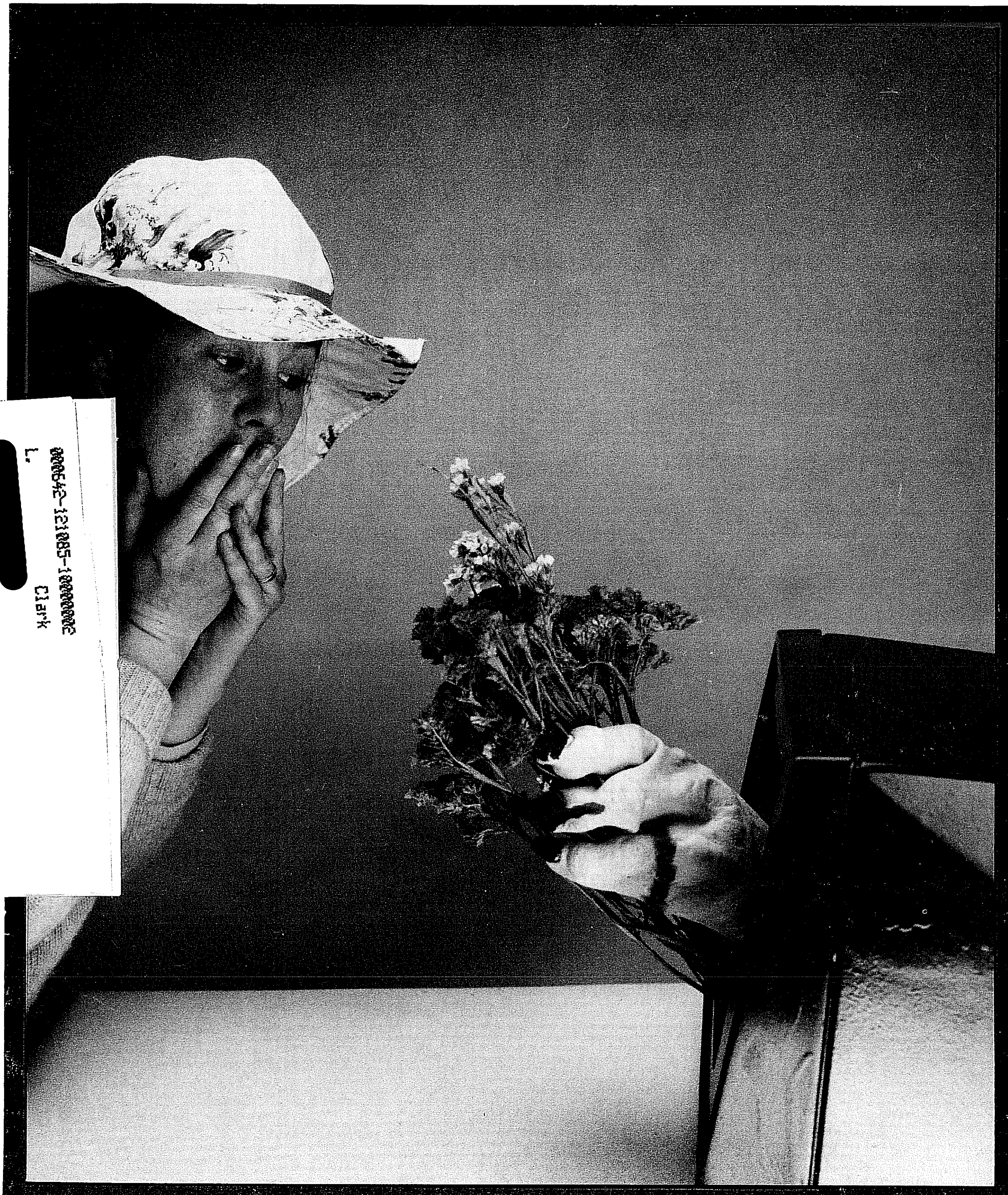
UNDERCOLOR

Vol. 1 No. 10 June 7, 1985



$2.00

A SPECTRUM PROJECTS SHOPPING LIST

A CHIP OFF THE OLD...

6821 Standard PIA $9.95
68766 (Fits Disk Basic Skt) Eprom.$24.95
27128 16K DOS Eprom (Or 2 DOS's!).$24.95
New SAM Chip w/heatsink (74LS785).$29.95
6809E CPU Chip $29.95
Ext Basic 1.1 ROM - NEW LOW PRICE.$29.95
28 pin Ext Basic upgrade 26-3134A.$34.95
Basic ROM 1.2 Chip (30% FASTER) ..$39.95
Disk ROM 1.1 (New DOS Command) ..$39.95
Eprom Eraser - 3 min erasure time.$49.95
Lower Kit Bd - Specify CoCoI/II**.$59.95
CoCo First Aid Kit - includes 2 PIAs, 6809E & SAM (Be Prepared!!!) $59.95
26-3134A & 26-3136A 64K Upgrade - 2 chip set (ONLY for new Korean CoCoIIA).$69.95
Eprom Prgmr (2ms speed/2K - 16K).$139.95
64K CoCo II - w/NEW keyboard $169.95

COCO LIBRARY...

CoCo Memory Map $12.00
Rainbow Book & Tape of Adventures.$14.95
Basic Programming Tricks Revealed.$14.95
The FACTS - Inside "guts" of CoCo.$14.95
500 Pokes, Peeks 'N Execs $16.95
Basic 09 Tour Guide $18.95
Disk Basic (1.0/1.1) Unraveled ...$19.95
Color Basic 1.2 Unraveled $19.95
New! CoCo II Service Manual $19.95
The Complete Rainbow Guide to OS9.$19.95
W/Two Disk Package of demo pgms ..$49.95

MORE GOOD STUFF...

27128/68764 Adapter - Allows you to plug-in 28pin EPROMs into 24pin sockets! Fits all controllers for 16K DOSs $19.95
CoCo Freeze Frame - Stop your CoCo dead in its tracks! Put games on "HOLD"$19.95
CoCo Light Pen with 6 programs ...$24.95
Spectrum Voice Pak - SAVE $30!! ..$39.95
PBH Parallel Interface - Beats Botek! 300-9600 baud w/ptr-modem switch .$69.95
PBJ WORD-PAK II - Hi-Res 80x24 display w/smooth scrolling & 8x10 matrix $139.95
PBJ C-C Bus - 6 slot interface ..$149.95
NOMAD - The CoCo Robot $199.95

All orders plus $3.00 S/H (Foreign $5.00)
COD add $2.00 extra
NYS Residents add Sales Tax

COCO CABLES AND...

Printer/Modem 15' Extender Cable .$14.95
Tired of unplugging devices from your RS232 port? Try a RS232 "Y" Cable.$19.95
Joystick/Mouse 10' Extender Cable.$19.95
Null Modem Cable - 40 pin to DB25.$24.95
Disk Interface/Rom Pak Extender - Move your disks & ROM Paks (3 feet) ..$29.95
Triple RS232 Switcher - Now select one of any three RS232 peripherals ...$29.95
40 Pin Dual "Y" Cable - Hook up a Disk w/Voice, Word Pak, CoCo Max,etc ..$29.95
Triple "Joyport" Switcher - Joystick, Mouse, Touch Pad and/or Light Pen $39.95
40 Pin Triple "Y" Cable - Hook up any 3-Voice/Word/RS232/Digitizer PAKs ..$39.95
Finally! 24" Multi-Pak Extender ..$39.95

OTHER GOOD STUFF...

C-10 tapes in any quantity .....49 cents
5 1/4" Diskettes in any quantity ..$1.49
Joystick, Cassette or Serial plug .$3.99
32K, 64K or 128K RAM Button $4.99
Rompak w/Blank PC Brd 27xx series .$9.95
The Disk Doubler - Doubleside your 5 1/4 diskettes for 160K more storage ..$19.95
Video Clear - This cable will reduce TV interference created by CoCo! $19.95
Video Reverser -Reduce eyestrain w/green letters on black background** $24.95
The Magic Box - Load Mod I/III Basic program tapes into the CoCo $24.95
DOS Switcher - Select from any two DOSs (Disk 1.0 1.1, JDOS) in J&M ctlr .$24.95
Stereo Pak - Hardware synthesizer used w/Musica 2...superb stereo sound! $39.95
CoCo Cooler- State D,E or CoCoII .$49.95
MicroWorks Video Digitizer $149.95
** N O T for 26-3134/36 CoCo II's

SPECTRUM PROJECTS
PO BOX 21272
93-15 86TH DRIVE
WOODHAVEN NY 11421

718-441-2807

SPECIAL 64K RAM CHIPS $29.95

Who's Who

Editor
Debra A. Marshall

Technical
Dennis Bathory Kitsz

Contributing Photographer
Charley Freiberg

Production Manager
Clare McCarthy

Paste-Up
Janet Patterson

Logistics Coordinator
JoAnn Trottier

Technical Consultant and Publisher
Dennis Kitsz

Office
(802)485-6440

Editorial Office
(207)785-5148

Published By: ColorPlus Inc., Box 6809, Roxbury, VT 05669. (802)485-6440.

Circulation Problems? Contact David Gunn, Box 6809, Roxbury, VT 05669, 802-485-6440.

Subscriptions: United States 1 year—$33; 2 years—$66. First class mail 1 year—$47; 2 years—$94. Canada and Mexico 1 year $43; 2 years $86. Overseas 1 year $53 (US dollars); Airmail $65 (US dollars). UnderColor is published 26 times a year. Call (802)485-6440, or write to UnderColor, Box 6809, Roxbury, VT 05669, Attn: Perley Swann.
Advertisers: rates available.
Editorial Office: c/o Debra Marshall, RFD 1, Box 786, Union, ME 04862, 207-785-5148. Submissions are welcome; include S.A.S.E. for returns. To receive author's guide, write: Author's Guide, Box 6809, Roxbury, VT 05669. Letters to Editor, manuscripts, article queries, etc. should be sent to editorial office.

Table of Contents

A friend of mine showed me a game he'd programmed. He owns an Apple II. He progammed the game using a disk called "The Arcade Machine," by Broderbund Software.

I was wondering if there is any disk like The Arcade Machine, which used simple one-word commands to program games, available for the Color Computer? If so, where may I get one?

Vince Matthews
Clinton, NC

"The Game Machine" is the only program I've heard of that does what you want. The program is sold on a ROMpak and you must use the ROMpak to play the games you create. In other words, you can't make a game and then give it to a friend. To play it your friend would have to have The Game Machine ROMpak. Unfortunately, I can't find the manufacturer's address.

Spectrum Projects (P.O. Box 21272, Woodhaven, NY, 11421) sells "Bjork Blocks," a utility for creating graphics screens for use in games.

Sorry I can't help you more.

In the November, 1983 issue of TCCM, *you answer an inquiry as follows: "Your computer may already have 64K RAM, you just can't access it. If you have the latest board, all you need is software (such as Telewriter-64 and Super Color Writer 3.0) to access the full 64K capability." I have the 64K RAM and the latest board, checked out recently by Computer Plus in Littleton, Mass. However, with Telewriter-64 I can only reach 24,896. I have therefore just purchased VIP Writer which gets me 41,668. Furthermore, I note in the recent ads for VIP Writer they claim I should be able to get up to 50,000.*

I was totally satisfied with the Telewriter-64 I was used to and liked but wanted the extra room that VIP Writer gave me without getting involved with Flex, etc. I would still like to use my Telewriter-64 if I could get more memory out of it. Have you any suggestions?

Monte G. Basbas
Wayland, MA

I suggest calling Cognitec directly and asking them what gives with the version of Telewriter you have. It should take advantage of the full memory of your computer.

I wish to thank Robert C. Lake, CDP, for writing the excellent program MOVSCRIP which appeared in The Color Computer Magazine in August 1983.

As you know, it can be used to customize printing from the Radio Shack Scripsit ROMpak to the Radio Shack LP VII printer. However, I have a Radio Shack DMP-120 printer which uses different codes for wide printing. The program runs fine except it doesn't do wide printing. I'm thankful to be able to have a back-up to the ROMpack but sure wish I could change the printing.

Since I got a tape from the Scripsit ROMpack, I figured I could solve my old problem of wipeouts of Radio Shack "Spectaculator" by using the same program. It didn't work out. The program ran up to line 610 where I got an FC error. the original line 610 is; POKE(J-24576,PEEK(J-24576)-(24576/256). *Can anyone help with either of these two problems?*

Jim Fairchild
Newark, DE

All you have to do is substitute the DMP width codes for the LP VII code in the program. Look up the codes in your manual andd re-run the modified program on your Scripsit cartridge.

Your second problem is that the balue Scripmov derives for J is incorrect for Spectaculator, hence the FC error. Unfortunately I don't have Spectaculator, so I can't help you. Does anyone else have a solution?

I have a Color Computer, roughly one year old. Three months ago I had somebody I know do a 32K piggyback upgrade; however, certain software wouldn't load into 32K RAM so I invested in 64K chips. The 64K upgrade went easily (I followed instructions from Dennis Kitsz of Green Mountain Micro), but—with each type upgrade in the computer I have sometimes experienced the cursor not moving *past its home position (to borrow an Apple term) upon power-up. When I unplug and walk away for a while, the computer returns to normal. The required time away from the computer varies from five minutes to 30 minutes.*

Why does this happen? Are the upgrades at fault? Is it my old (Sears) TV or printer or disk drive giving a current surge?

And finally, what damage will this cause, other than to my mental health?

Michael Feinberg
Silver Spring, MD

First, you might have heat-damaged one or more of the electronic components (not that you'd notice by looking at them), causing it/them to mess up the PIA controlling the keyboard.

Second, the keyboard cable might not be properly seated; try unplugging it, inspecting it carefully for

loose connections, cleaning the computer edge-card connector, and reseating the cable.

Third, you could have a bad RAM chip(s) affecting the keyboard rollover table in low RAM.

And finally, the CPU could be malfunctioning. (These last two are very unlikely.)

Or, it could be all these. Before you panic, swap the two PIAs with each other. If one of them is damaged your keyboard problem will cease (but you are just moving the problem elsewhere). If swapping the two chips cures your complaint, order a replacement PIA chip for the bad one (Spectrum Projects sells parts chips).

It probably isn't your upgrades since they either work or don't work. One area of suspicion is the extra address line you had to add: did you put a ferrite bead or dropping resistor on it? If not, that could cause all kinds of EMI interference with memory, but the interference would show up at random times, not just when you were powering up the computer.

Your TV, printer, and disk drives have nothing to do with the problem; to test that out just unplug and leave off the printer and disk drives and turn on the TV long before you turn on the computer. If surges like that were the problem, disconnecting the devices would cure the complaint. Adding a surge suppressor to the computer power line would prevent the problem.

Eventually the problem will reach the point where the computer won't work at all.

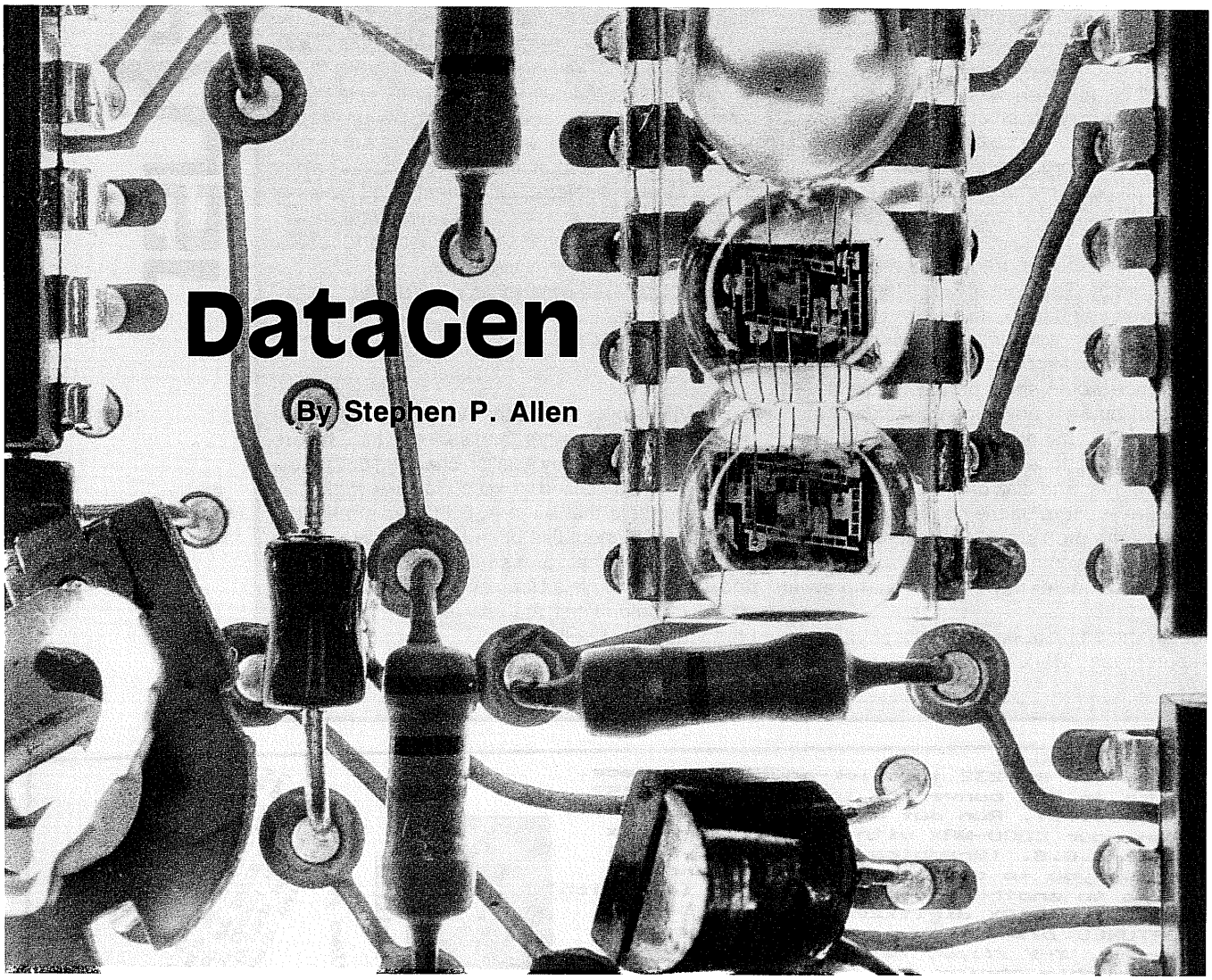
I use VIP Writer for almost all my correspondence. In fact, I am so impressed with the program that I sent away to England for the Dragon version, as I own both a CoCo and a Dragon 64.

I bought another program for my CoCo when I was using Color Script instead of VIP Writer, called "Spell 'n Fix (tape version)." This program was very satisfactory before I started to use VIP or Dragon Writer. The Spell 'n Fix program requires a file which is saved in ASCII format to work properly. VIP Writer claims to save its files in this format, but I have not been able to get it to work. I can save files with VIP, and then load them into Color Scripsit and resave them for use with the Spell 'n Fix.

Can you give me some information on this matter?

Charles M. MacLeod
Taunton, MA

The VIP people tell me that VIP Writer files must be printed to tape rather than saved to tape before the files can be read by spelling checking programs. The technician I talked to wasn't familiar with Spell 'n Fix, but said that if the ASCII file was readable by Scripsit, it should've been readable by Spell 'n Fix. They have their own spelling program for VIP and it reads any file as long as it is in ASCII format, so they couldn't tell me why the VIP files were unreadable by Spell 'n Fix.



DataGen

By Stephen P. Allen

When you've written a Basic program that calls an Assembly Language subroutine, the usual procedure is to represent the Assembly Language with data statements that get read and POKEd into memory at run time. For the programmer it can be a real hassle getting those data statements written and put into the program correctly. One misspelled datum and the routine jumps out the window!

DataGen is a utility that vastly simplifies this process. It's based on a program called Append Data by Peter Stoloff that originally appeared in *Color Computer News* (June '83). His program impressed and intrigued me, and I spent some time playing with it . . . here's what I ended up with.

Append Data and DataGen both do the same thing: automatically generate data statements directly into Basic memory. Peter's program used a Basic program to read and POKE an Assembly Language routine, and process the start and end addresses of the memory block to be converted. You'd run Append Data, and have Append Data plus data statements. After deleting Append Date you could type in or otherwise merge your target Basic program.

There are some interesting differences and additions in DataGen: it's pure Assembly Language and can co-reside with any other Assembly Language or Basic programs. In other words, you can have DataGen, your Basic target program, and your Assembly Language routine all in memory at the same time with no conflict, and produce a finished Basic program instantly.

I've added a checksum routine; a simple "modulo 8" checksum is automatically generated and appended to the data. (In a mod 8 checksum the maximum size of the sum is $FF, or 255. A rough equivalent in decimal would be to add three and nine and, discarding the carry, get two.)

The data produced are different; here's an example:

Append Data: DATABDA1C126FB39

DataGen: DATA 189,161,193,38,251,57

You can easily see the difference. His data are much more compact, but also harder to read (I wrote DataGen the way I did just to be different).

DataGen uses two ROM routines that, used together, make keyboard input a snap to implement in your Assembly Language programming. The two routines are at $A390 and $9F.

In Basic, whenever you see that flashing multi-color cursor, the 6809 processor is executing the routine at $A390, Basic's input routine. The device that supplies the input is specified by memory location $6F (DEVNUM, in Microsoft parlance). The input device can be keyboard or cassette (for CLOADing an ASCII program) in standard Basic. In Extended Basic, $A390 is used by CLOAD to input from the RS-232 port. I'll discuss keyboard input here, as that's what this program uses.

When $6F = 0, the input device is the keyboard. $A390 flashes the cursor and waits for key presses. From this routine you can enter text, backspace, clear the current line, or clear the screen. The routine ends when you hit Enter or Break. Upon return, the carry flag is set if the last key was Break, or cleared if the last key was Enter. This makes it easy to JSR $A390: BCS EXIT, where Exit is a break routine. The B register has the total number of key presses, and the X register points one byte before the address of the keyboard buffer. This routine was designed to be used with another powerful subroutine—the routine at $9F.

$9F gets the next character, whose address is contained in $A6 and $A7, and evaluates it. It gets the next character from your Basic program when the program is running, and is also used in the tokenization of a typed-in program line after you hit Enter. The address of the next character is part of the routine—it's self-modifying code!

Upon return from $9F, the A register has the next character. The carry flag is set if the character is within the range $30–$39 (ASCII 0–9), providing a quick check for presence of a number. If the character is zero or a colon the zero flag is set. All this using only the A register! That's why the self-modifying code. One other thing, $9F automatically skips spaces. If it "gets" a space ($20) it loops back and tries again. Incidentally, by my count, this routine (which can also be called at $A5, to re-evaluate the current character) is the most often-called routine in the Basic ROMs, followed by the routine that checks for a comma and gives back an $SN ERROR message if it ain't there.

$AF67 (Line 410) is the routine Basic uses to evaluate a typed-in Basic line number. CVTHEX (Line 600) is the same routine rewritten to handle hex input.

$AD01 (Line 1170) is a routine to find a Basic line in memory. If the line exists its address is returned in the X register and the zero flag is set. If not, either the next higher line's address or the end of the Basic program is returned. The use of this routine makes it possible to execute DataGen several times on the same block of data, overwriting previously produced data statements. Just remember that when DataGen is done, it sets the end of its data statements as the end of your Basic program, and you should normally set a starting line number beyond the highest line number of your Basic program.

When DataGen is done it writes a datum of 1000, then the checksum, which will be a number between zero and 255. A read/POKE routine would go like this:

```
100 CS=0: AD=XXXX 'starting address of routine
110 READ OP: IF OP=1000 THEN 130
120 POKE AD,OP: AD=AD+1:CS=CS+OP AND 255: GOTO110
130 READ CK:IFCK=CS THEN CLEAN EXIT ELSE DATA BAD: END
```

When DataGen is actually generating the program you'll see a brief flash at the bottom of the screen. That's the workspace for CONVRT. I find the speed of assembly language to be an endless source of wonder—here's another chance to see it in action.

DataGen is relocatable (as any utility must be). It ordinarily originates on graphics page 1, where it's out of the way of Basic, but can be shifted around if need be. *(end)*

Program Listing: DataGen

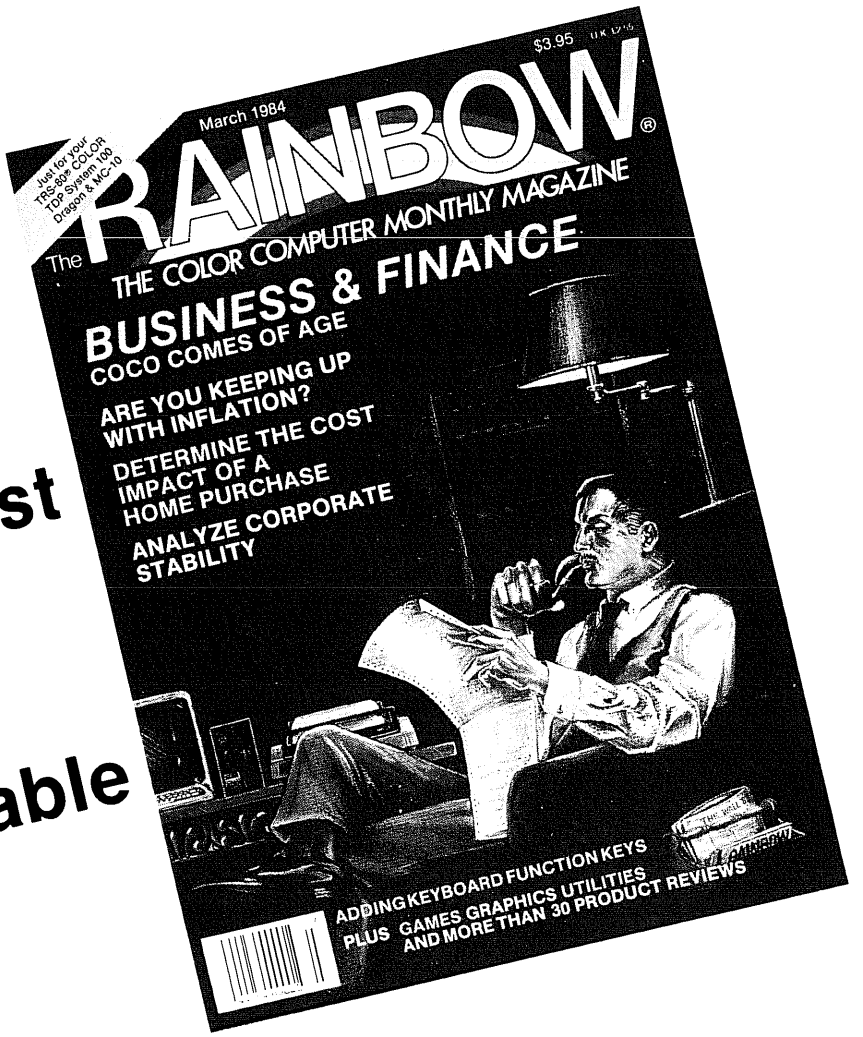
```
                        00200 * ORIGIN IS ON ECB GRAPHICS PAGE ONE:
                        00210 * CHANGE TO $F00 FOR DISK BASIC
                        00220
0700                    00230          ORG    $700
                        00240
                        00250
                        00260
                        00270
0700 BD A928            00280 START    JSR    $A928   clear the screen
0703 30 8D 0130         00290 START2   LEAX   STAD,PCR point to message
0707 8D 26              00300          BSR    PUTMES  put it up and get input
0709 8D 2F              00310          BSR    CVTHEX  convert start address...
070B ED 8C 67           00320          STD    <$1,PCR ... and store it
                        00330
070E 30 8D 013E         00340          LEAX   ENDAD,PCR point to message
0712 8D 1B              00350          BSR    PUTMES  put it up, get input
0714 8D 24              00360          BSR    CVTHEX  convert end address...
0716 ED 8C 5E           00370          STD    <$2,PCR ...and store it
                        00380
0719 30 8D 014A         00390 LINE     LEAX   LINENO,PCR point to message
071D 8D 10              00400          BSR    PUTMES  put it up, get input
071F BD AF67            00410          JSR    $AF67   convert decimal input
0722 DC 2B              00420          LDD    <$2B    converted number
0724 27 F3              00430          BEQ    LINE    (zero not allowed)
0726 ED 8C 50           00440          STD    <LN,PCR store line number
0729 20 53              00450          BRA    GETDAT  go do it!
                        00460
                        00470
                        00480 * Print a message and get keyboard input
                        00490
072B AD 9F A002         00500 LOOP20   JSR    [$A002] print a character
072F A6 80              00510 PUTMES   LDA    ,X+     get next character
0731 26 F8              00520          BNE    LOOP20  if not end of message
0733 BD A390            00530          JSR    $A390   else get keyboard input
0736 9F A6              00540          STX    <$A6    point $9F routine to kbd buffer
0738 0E 9F              00550          JMP    <$9F    get first character and return
                        00560
                        00570 *** Convert a typed in Hex number
                        00580 *** to binary in D register
                        00590
073A 9E 8A              00600 CVTHEX   LDX    <$8A    zero...
073C 9F 2B              00610          STX    <$2B    ... to workspace
                        00620
073E 25 0A              00630 LBL10    BCS    LBL20   if ASCII zero to nine
0740 80 07              00640          SUBA   #7      else try to make $3A to $3F
0742 81 3F              00650          CMPA   #$3F    too high
0744 22 1F              00660          BHI    ERROR
0746 81 3A              00670          CMPA   #$3A
0748 25 1B              00680          BLO    ERROR   too low
074A 80 30              00690 LBL20    SUBA   #$30    convert $30-$3F to 0-$F
074C 97 01              00700          STA    <1      (temporarily stored)
                        00710
074E DC 2B              00720          LDD    <$2B    number so far
                        00730          ASLB
0750 58                 00740          ROLA           shift over
0751 49                 00750          ASLB           (multiply by 16)
0752 58                 00760          ROLA
0753 49                 00770          ASLB
0754 58                 00780          ROLA
0755 49                 00790          ASLB
0756 58                 00800          ROLA
0757 49                 00810          ADDB   <1      add latest digit
0758 DB 01              00820          ADCA   #0
075A 89 00              00830          STD    <$2B    save result
075C DD 2B              00840          JSR    $9F     another digit?
075E 9D 9F              00850          BNE    LBL10   yes...
0760 26 DC              00860
                        00870
0762 DC 2B              00880          LDD    <$2B    ... else get final result...
0764 39                 00890          RTS            ... and return
                        00900
```

```
                        00910 *** Error routine for
                        00920 *** incorrectly typed Hex
                        00930
0765 32   62            00940 ERROR   LEAS    2,S       'pop return address
0767 30   8D 0117       00950         LEAX    ERMES,PCR point to error message
076B A6   80            00960 LOOP30  LDA     ,X+       get next character
076D 27   94            00970         BEQ     START20 if done, try another round
076F AD   9F A002       00980         JSR     [$A002] else print character
0773 20   F6            00990         BRA     LOOP30  keep going
                        01000
                        01010
                        01020 * Variables stored here:
                        01030
0775                    01040 S1      RMB     2       start address of M/L block
0777                    01050 S2      RMB     2       end address of M/L block
0779                    01060 LN      RMB     2       Basic line number to build on
077B                    01070 LOC     RMB     2       points to start of Basic line
077D                    01080 CKSUM   RMB     1       checksum forms here
                        01090
                        01100
                        01110 ***  APPEND-DATA  ***
                        01120 * (with a few changes)
                        01130
077E 32   7F            01140 GETDAT  LEAS    -1,S    make room for temp storage
0780 10AE 8C F1         01150         LDY     <S1,PCR Point Y to start of M/L
0784 EC   8C F2         01160         LDD     <LN,PCR starting line number
0787 BD   AD01          01170         JSR     $AD01   find line in memory
078A 6F   8C F0         01180         CLR     <CKSUM,PCR
                        01190
                        01200 *  Now X points to Basic,
                        01210 *       Y points to M/L
                        01220
078D AF   8C EB         01230 NEWLN   STX     <LOC,PCR save address of current line
0790 30   02            01240         LEAX    2,X     move past line-link pointer
0792 EC   8C E4         01250         LDD     <LN,PCR get line number...
0795 ED   81            01260         STD     ,X++    ... and store in BASIC line
0797 C3   000A          01270         ADDD    #10     bump line number...
079A ED   8C DC         01280         STD     <LN,PCR ... and save for next pass
079D CC   8620          01290         LDD     #$8620  "DATA" token, with space
07A0 ED   81            01300         STD     ,X++    to start of line
07A2 6F   E4            01310         CLR     ,S      #data this line = zero
07A4 20   04            01320         BRA     NOCOM   no comma first time through
                        01330
07A6 86   2C            01340 LOOP    LDA     #',     ASCII comma...
07A8 A7   80            01350         STA     ,X+     ... to BASIC memory
07AA 10AC 8C C9         01360 NOCOM   CMPY    <S2,PCR block done?
07AE 2E   63            01370         BGT     DONE    if yes
07B0 A6   A0            01380         LDA     ,Y+     next byte from M/L block
07B2 34   02            01390         PSHS    A
07B4 AB   8C C6         01400         ADDA    <CKSUM,PCR
07B7 A7   8C C3         01410         STA     <CKSUM,PCR
07BA 35   02            01420         PULS    A
07BC 8D   10            01430         BSR     CONVRT  make ASCII and store in Basic
07BE 6C   E4            01440         INC     ,S      count one more datum in line
07C0 86   07            01450         LDA     #7      change to suit yerself
07C2 A1   E4            01460         CMPA    ,S      done this line?
07C4 26   E0            01470         BNE     LOOP    not yet
                        01480
                        01490 *  Process end of Basic line
                        01500
07C6 6F   80            01510         CLR     ,X+     flag end of BASIC line
                        01520
                        01530 * X now points to next Basic line
                        01540
07C8 AF   9C B0         01550         STX     <[LOC,PCR] link new line to last one
07CB 20   C0            01560         BRA     NEWLN   keep going
                        01570
                        01580
                        01590 *  Convert data in "A" register *
                        01600 *  to 3-digit number and insert *
                        01610 *  into Basic memory           *
                        01620
                        01630
07CD                    01640 LZFLG   RMB     1       flag for supressed leading zero
                        01650
                        01660 * First, make 3-digit number
                        01670
07CE CE   05F2          01680 CONVRT  LDU     #$5F2   video screen is temp workspace
07D1 6F   C4            01690         CLR     ,U
07D3 6F   C2            01700         CLR     ,-U     clear out three bytes
07D5 6F   C2            01710         CLR     ,-U     decimal number forms here
                        01720
07D7 6C   C4            01730 LOOP1   INC     ,U      bump hundreds counter
07D9 80   64            01740         SUBA    #100    decrement number
07DB 24   FA            01750         BCC     LOOP1   if not past zero
07DD 8B   64            01760         ADDA    #100    too far: adjust
07DF 6A   C0            01770         DEC     ,U+     adjust counter and move up
07E1 6C   C4            01780 LOOP2   INC     ,U
07E3 80   0A            01790         SUBA    #10     same for tens counter
07E5 24   FA            01800         BCC     LOOP2
07E7 8B   0A            01810         ADDA    #10
07E9 6A   C0            01820         DEC     ,U+
07EB 6C   C4            01830 LOOP3   INC     ,U
07ED 80   01            01840         SUBA    #1      ... also for ones counter
07EF 24   FA            01850         BCC     LOOP3
07F1 6A   C4            01860         DEC     ,U
                        01870
                        01880 * Now insert into Basic program
                        01890
07F3 33   5E            01900         LEAU    -2,U    point to hundreds counter
07F5 C6   03            01910         LDB     #3      three digits to show
07F7 6F   8C D3         01920         CLR     <LZFLG,PCR leading zeros supressed
                        01930
07FA A6   C0            01940 LOOPA   LDA     ,U+     get number
07FC 6D   8C CE         01950         TST     <LZFLG,PCR leading zeros supressed?
07FF 26   0A            01960         BNE     PUTIT   if not
0801 C5   02            01970         BITB    #2      last digit?
0803 27   06            01980         BEQ     PUTIT   if so
                        02020
080B 8B   30            02030 PUTIT   ADDA    #$30    convert to ASCII
080D A7   80            02040         STA     ,X+     and store in BASIC line
080F 5A                 02050 NEXT    DECB            done three digits?
0810 26   E8            02060         BNE     LOOPA   not yet
0812 39                 02070         RTS
                        02080
                        02090
                        02100 **       DONE!          **
                        02110 ** CLEAN UP AND GO HOME **
                        02120
                        02130 *   Write Datum of 1000
                        02140
0813 CC   3130          02150 DONE    LDD     #$3130  ASCII "10"
0816 ED   81            02160         STD     ,X++    to BASIC line
0818 4A                 02170         DECA            ASCII "00"
0819 ED   81            02180         STD     ,X++
                        02190
                        02200 *   Write checksum
                        02210
081B 86   2C            02220         LDA     #',     one more comma
081D A7   80            02230         STA     ,X+     to Basic program
081F A6   8D FF5A       02240         LDA     CKSUM,PCR
0823 8D   A9            02250         BSR     CONVRT  put checksum at end of program
                        02260
                        02270 * Flag End of Basic program
                        02280
0825 6F   80            02290         CLR     ,X+     flag end of line
                        02300
0827 AF   9D FF50       02310         STX     [LOC,PCR] link last line
082B 6F   80            02320         CLR     ,X+     flag end of BASIC program
082D 6F   80            02330         CLR     ,X+
082F 9F   1B            02340         STX     <$1B    end program pointer
0831 9F   1D            02350         STX     <$1D    end variables pointer
0833 9F   1F            02360         STX     <$1F    start free space pointer
0835 35   82            02370         PULS    A,PC    adjust stack and return
```

The Biggest

The Best

The Indispensable



The latest news on the Color Computer grapevine is that more and more people are discovering **The Rainbow.**®

Now in its third year, **The Rainbow** has become the standard by which other Color Computer magazines are compared. And no wonder! **The Rainbow** towers above the crowd, now offering up to 300 pages each month, including as many as two dozen type-in-and-run program listings, a host of articles and in excess of 30 hardware and software product reviews.

We lead the pack in Color Computer publications and are devoted *exclusively* to the TRS-80® Color, TDP-100 and Dragon-32. We made our climb to the top by continually offering the best and the most by such well-known authors and innovators as Bob Albrecht and Don Inman, and games from top programmers like Robert Tyson, Fred Scerbo and John Fraysse. **The Rainbow** offers the most in entertainment and education, home uses, technical details and hardware projects, tutorials, utilities, graphics and special features like Rainbow Scoreboard and our CoCo Clubs section.

For only $28* a year, you get the keys to all the secrets locked in your CoCo!

Are you searching through the jungle of claims and clamor? Climb above it all. Look up. Find **The Rainbow.**

THE RAINBOW
(502) 228-4492

9529 U.S. Highway 42
The Falsoft Bldg.
P.O. Box 385
Prospect, KY 40059

THE RAINBOW is a publication of Falsoft, Inc.

YES! Sign me up for a year (12 issues) of **THE RAINBOW.**

Name _____

Address _____

City _____ State _____ ZIP _____

Payment Enclosed ☐

Charge Visa ☐ MasterCard ☐ American Express ☐

My Account# _____

Signature _____ Card Expiration Date _____

In order to hold down non-editorial costs, we do not bill for subscriptions.

AMERICAN EXPRESS

MasterCard

VISA

*Subscriptions to **the RAINBOW** are $28 ($31 after Jan. 1, 1985) a year in the United States. Canadian rate $35 U.S. ($38 after Jan. 1, 1985). Surface rate to other countries $65 U.S. ($68 after Jan. 1, 1985); air rate $100 U.S. ($103 after Jan. 1, 1985). All subscriptions begin with the current issue. Please allow up to 5-6 weeks for first copy. U.S. FUNDS only.

Prices subject to change.

```
                        02380
                        02390
                        02400 ** All messages start with <Enter>
                        02410 ** And end with Zero
                        02420
0837    0D              02430 STAD    FCB     $0D
0838    53              02440         FCC     /STARTING ADDRESS (HEX)?/
        54
        41
        52
        54
        49
        4E
        47
        20
        41
        44
        44
        52
        45
        53
        53
        20
        28
        48
        45
        58
        29
        3F
084F    00              02450         FCB     0
0850    0D              02460 ENDAD   FCB     $0D
0851    45              02470         FCC     /ENDING ADDRESS (HEX)?/
        4E
        44
        49
        4E
        47
        20
        41
        44
        44
        52
        45
        53
        53
        20
        28
        48
        45
        58
        29
        3F
0866    00              02480         FCB     0
0867    0D              02490 LINENO  FCB     $0D
0868    53              02500         FCC     /STARTING LINE# (DECIMAL)?/
        54
        41
        52
        54
        49
        4E
        47
        20
        4C
        49
        4E
        45
        23
        20
        28
        44
        45
        43
        49
        4D
        41
        4C
        29
        3F
0881    00              02510         FCB     0
0882    0D              02520 ERMES   FCB     $0D
0883    65              02530         FCC     /error/
        72
        72
        6F
        72
0888    00              02540 ZEND    FCB     0
        0700            02550         END     START
00000 TOTAL ERRORS

CKSUM   077D            LN      0779          NEXT    080F
CONVRT  07CE            LOC     077B          NOCOM   07AA
CVTHEX  073A            LOOP    07A6          PUTIT   080B
DONE    0813            LOOP1   07D7          PUTMES  072F
ENDAD   0850            LOOP2   07E1          S1      0775
ERMES   0882            LOOP20  072B          S2      0777
ERROR   0765            LOOP3   07EB          STAD    0837
GETDAT  077E            LOOP30  076B          START   0700
LBL10   073E            LOOPA   07FA          START2  0703
LBL20   074A            LZFLG   07CD          ZEND    0888
LINE    0719            NEWLN   078D
LINENO  0867
```

Fm: William Harris (to W. Day)

Isn't smart enough? It also could be that he works a job during the day and doesn't have the time to put into a complete file system. I've written a chess program for the CoCo (which, unfortunately, is still not up to an adequate level of play, as far as I'm concerned), but right now, working 10 hour days, I haven't the time to improve its play, let alone fiddle with a disk opening book. If this makes me too unintelligent to write good software for the CoCo, so be it.

Fm: Wayne Day (to W. Harris)

Bill, Notice that in all of this discussion, I have tried to emphasize what I believe to be the differences in responsibilities of someone who writes software for himself, or a small circle of friends, or for something like the SIG, and the person who sets out to write something intended for commercial distribution.

The guy that writes it for himself should feel free to do anything that makes him comfortable.

The guy who's trying to sell it owes something to the purchasers of his software, something besides "Well, it works on *my* machine."

Fm: William Harris (to W. Day)

I agree with that. But—in my case, I started my program with the idea that if it never amounted to much, I wouldn't be too disappointed; and if it played like a champ, I'd try to sell it. Now are you saying that even though I started this as a personal project, I am forever more barred from selling the program? Of course not.

In my own case, the lack of disk I/O only deprives the program of a good, but not vitally necessary, feature. Indeed, if I were to take the route you are apparently suggesting (program for myself and my own personal group of acquaintances only) I could very easily solve the problem of disk I/O by just storing my information on a data disk, completely bypassing the DOS directory scheme.

Unfortunately, I can see the dilemma of a professional programmer, who has an idea for an excellent graphics program. He sits down, sketches the program out, gets rudiments of it running—and then he reaches the decision about whether or not he should continue. The time and effort spent in writing a set of disk file handing routines for a program which will require many more months of work is likely to completely deter him from completing his program.

Fm: Wayne Day (to W. Harris)

Well, let me answer your question this way . . .

If you can reasonably assure yourself, and still be able to sleep at night, that your program will run properly on any three CoCos I choose (assuming, of

course, they have any special hardware that is needed, like a Workpak or RS-232 pak, or whatever special needs are built into the program), then I've got absolutely no problem with it.

My choices are:

1. "D" Board CoCo with 1.0 Basic, 1.0 Extended, and 1.0 Disk.

2. TDP-100 with 1.1 Basic, 1.0 Extended and 1.1 Disk.

3. CoCo-II (Vers. 3)—the new one Marty was telling us all about, with 1.3 Basic, 1.2 Extended and 1.3 Disk (or whatever it had in it).

The absolute main thing I'm trying to say in all of this is something on which a lot of folks seem to be missing the point. All I want is for programmers to be able to write their software so that it runs on *all Color Computers*, not just the one they have in their hands right now!!

The reason for that is that I want all CoCo owners to be able to purchase software and be reasonably sure it will work on their machines, not like a lot of programs.

Given your example, you've got everything except Disk I/O; then why not use the documented DSKCON call at $C004?

Heck, though it's not exactly in the best of taste, a lot of folks just take the Shack's code for a particular needed operation and stuff it into their program, maybe changing it up or cleaning it up a bit to get around the legal points.

I really don't care *how* it's done . . . just that it does need to be done to ensure the integrity of the investment of those who have CoCo's now, and of those who will be getting them in the future.

Fm: William Harris (to W. Day)

Well, again, in my case it'd be easy to just stuff information onto the disk with DSKCON. (And if I should eventually build that into my program, it'd probably the way I would have to go.) But while your sentiments are admirable, and I agree wholeheartedly with them in that programmers *should* write a program that'll run on everything, for any program that uses the disk and any programmer who lacks a couple of weeks (80 hours time, maybe?) to write disk routines, Tandy/Microsoft have made it fiendishly difficult.

My own program makes no undocumented calls (just CHROUT and POLCAT), and it has worked on all the CoCos I've been able to try it on.

Fm: Wayne Day (to W. Harris)

What, then, is the basis of the argument, William? In your example, it seems you've fulfilled what I think should be done—you've used the documented calls. You've made it compatible.

If you're writing the program with the idea that you're gonna sell it, you owe it to the people who are going to buy the program to give them something they can use.

It may be time-intensive, true, but would *you* want to spend $50 on some product only to find out it won't run on your machine, then be told that the warranty everyone uses says "sorry, it's as is where is"?

(Two replies follow:)

Fm: A. Flexser (to W. Day)

Wayne, where are all these programs you keep complaining won't run on all ROM versions? The only ones I've run into that fit into that category (commercial programs, I'm talking about) are old programs that came out *prior* to the new ROM versions, and, if they are still around, now exist in version that run on the new ROMs as well as the old. I agree entirely that it would be utterly reprehensible for anyone to be currently selling a program that doesn't run on all current and past ROMs, but I know of absolutely no case where anyone is actually doing this. Have you actually run into any cases of what you're deploring?

"I really don't care *how* it's done . . . to ensure the integrity of the investment . . ."

*(The reply series immediately following, marked by *, relates to this first reply:)*

***Fm: Wayne Day** (to A. Flexser)

The old Telewriter is on the top of the list, Art. But, no, since I have the old ROMs, in both my machines, I don't have any problems . . . just responding to the original question about compatibility.

***Fm: A. Flexser** (to W. Day)

But the old Telewriter was updated very quickly as soon as the new ROMs appeared, along with virtually everything else that needed an update. And it does a version check just like Vidtex, and uses one of two direct calls now depending on version. So, if Vidtex meets your criteria, I don't see who *doesn't.*

***Fm: Wayne Day** (to A. Flexser)

That just goes to show you, thank goodness, that most of the folks selling programs agree with me, then. You gots to make some provision for remaining compatible with the rest of the world!

Fm: William Harris (to W. Day)

Hark back a couple of messages. In my particular case, the lack of meaningful disk I/O (meaningful defined as Radio Shack DOS compatible—not just stuffing data onto the disk in some arbitrary manner that only my program can understand) is only a minor flaw. In my case, unless I do some kind of copying from Radio Shack ROM or DOS, or unless I do some undocumented calls to ROM, my program will never have disk I/O. I just don't have the time to fool around with it. But in an application that requires disk I/O, the argument still remains: Radio Shack has put up a significant barrier (involving *anyone*, expert of novice) that will quite likely prevent many a good programmer from writing and carrying through to market a program that needs to access the disk.

(Two replies follow:)

Fm: Wayne Day (to W. Harris)

I agree with that, William . . . just saying that there are ways around the Shack's barrier, and that commercial programs *need* to find some way around them.

Fm: Paul Hoffman (to W. Harris)

Bill, I know for a fact that if you don't want to write your own DOS, Peter Stark (of Star-Kits) will license the use of "Star-DOS" for any commercial applications. He tells me that there *are* programs out there using it. And the licensee only pays for the portions he needs in the particular application.

Fm: William Harris (to P. Hoffman)

License Star-DOS from a company that has publicly stated that it wants nothing more to do with the CoCo market? (See STAR.KIT in XA0.)

Furthermore, when's the last time you saw a product advertised that uses Star-DOS? I myself can only vaguely remember Starkits' advertisements about Star—DOS.

Fm: William Harris (to Marty)

Not to mention the fact that only checking for version numbers can *also* be a

lazy programming technique, compared to writing a complete file handling system!

Fm: Marty (to W. Harris)

As Sandy Trevor and I were discussing in an earlier thread, there is a middle ground between writing your own DOS from scratch and merely checking the version numbers. One is to check version number by looking at the DSKCON vector. This of course has potential problems if a new version disk ROM comes out that has the same DSKCON location but different locations of the file handling routines. A slightly more complex approach is to look for the two- and four-byte sequence of the entry point to the undocumented routine you seek to use. Most often when they alter the ROM the code for the beginning of a given routine is not altered. This technique has a lot of potential for making a program that uses the undocumented routines compatible with future generation ROMs. It's not foolproof, of course, but it does have a lot going for it.

What do you think of that idea . . . William? Sandy?

Fm: William Harris (to Marty)

I think that it's still going through a lot of wasted effort. But, wasted effort is better than no effort at all, and this scheme is the best one that doesn't flat-out require authorship of another DOS.

I believe your suggestion that even at this late date Tandy include thoroughly documented routines is the best idea yet. This would indeed have the advantage of fixing in concrete any and all checks for old DOS's entry points, as well as giving future authors something they can aim at.

I would also suggest that their next DOS be *finally* debugged. Some of their bugs (PCLEAR, although it's not a DOS bug, is the most (in)famous one! But the Close bug, the Field bug, and others can easily be called to mind) should have been caught even during the simplest of testing procedures. It makes me believe that whoever was responsible for the Basic, both at Tandy and Microsoft, didn't run it through much serious testing.

Despite my jumping on Radio Shack's case, however, I would like to say that their Basic and DOS, while buggy, are infinitely preferrable to Apple DOS 3.3! While we CoCo owners gripe and stew about *our* bugs, you should see what *they* have to put up with! (continued).

Fm: Roger Bouchard (to W. Harris)

The 'Field' bug? A new one for me. Can you please elaborate, as I use that command a lot in my graphic editor programs. (See, at least one person reads every word you write!)

Fm: William Harris (to R. Bouchard)

The condition occurs when you are manipulating multiple file buffers. Open 1, Field 1, Open 2, Field 2, (do some stuff), Close 2, LSET (a field from 1) = ?SE Error. Apparently the closing of file 2 makes it forget not only the buffer two fields, but also the buffer one fields.

Fm: Roger Bouchard (to W. Harris)

I see. Thanks for the info. Personally, I might not have to worry since I usually close everything at once when I'm finished. *(end)*

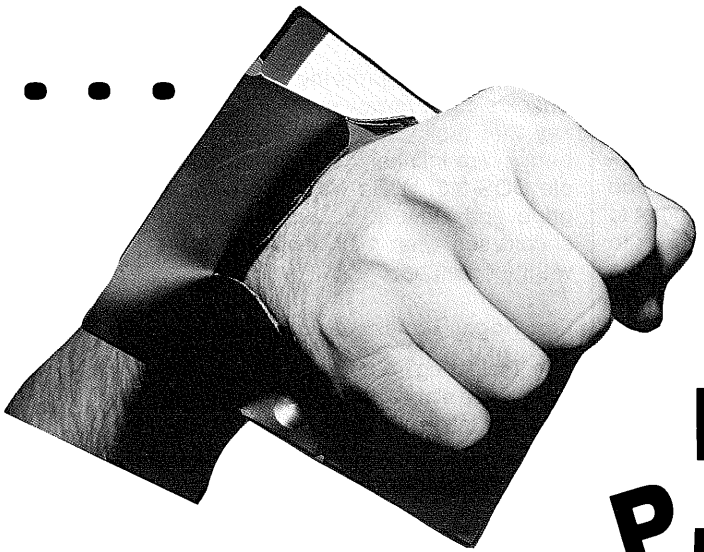
CLASSIFIED ADS NOTICE

Readers: Classified ad space is available at the rate of $1.25 per word. No headings or bold type will be allowed in user ads. Be sure to include your name, address, and telephone number for responses. Payment must accompany your ad order. For deadline, call UnderColor at (802) 485-6440

Commercial classified ads are also available at higher rates. Advertisers call (802) 485-6440 for rates and deadlines.

256K Internal upgrade, hardware and software. From $29.95. J & R Electronics. P.O. Box 2572. Columbia, MD 21045

Things You Ought To Know . . .



Disk Punch

By Mark Haverstock

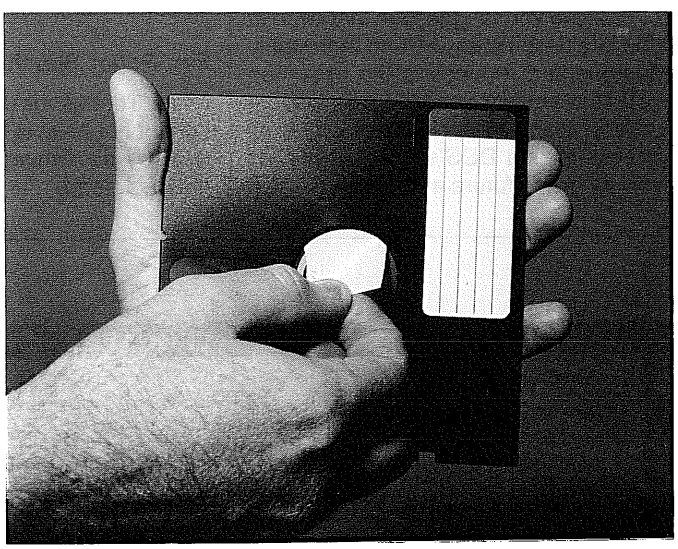
Using the *other side* of a disk is a time-honored method passed down from veteran computer users to novices. This article is for those who have been too timid to try.

A disk is a thin circular piece of mylar coated with magnetic material on both sides. They are available in single, double, and quad density ratings, single- or double-sided. The Color disk system specifies single-sided, double-density (SSDD). The manufacturer verifies an SSDD can read and write on one side of the disk at the stated double density. They make no claims or guarantees as to the fitness of the other side. Experience, however, shows 95 to 100 percent success in using the other side.

OK. Now go get: 1 paper punch; 1 1 × 3 inch piece of waxed paper or backing sheet from tractor feed labels; scissors, and rubber cement. Copy the template (Fig. 1). Mount it on a sturdy surface, such as posterboard. Punch out the index hole and write protect notches on your template. Match your template with a disk to be sure the holes are properly aligned.

Using a blank unformatted disk, line up the disk and template with the write protect notches opposite each other (Fig. 2). Carefully trace the index hole and notch onto the disk in pencil; turn the disk over and repeat.

Punch out the write protect notch on your disk first. The index holes are slightly more difficult, take care not to damage the magnetic surface of the disk. Use your thumb to lift up the black shell of the disk near the center hub. Insert the 1 inch wide piece of waxed paper or label backing to protect the disk surface (see photo). Carefully insert the paper punch and punch the index hole position marked on the cover of the disk (photo). Repeat this procedure on the other side.



Photos Thanks to Frank Lazzerri

Photo 1. Insert Wax Paper

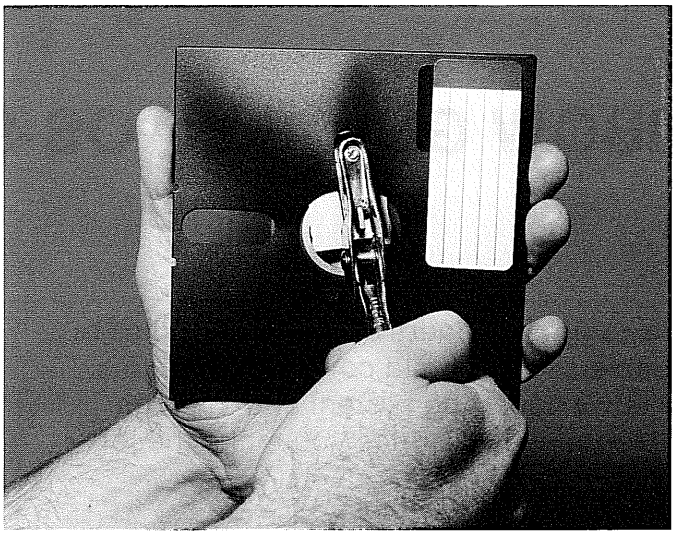


Photo 2. Punch Hole

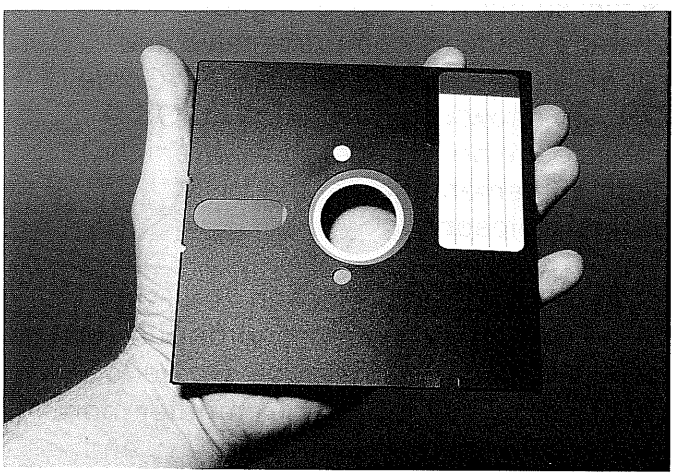


Photo 3. Finished *Other Side*

Next, examine the disk. The write protect notches and index holes should be fairly symmetrical.

Try formatting the new side of the disk using DSKINI. If all has gone well, the OK prompt will appear.

If you weren't so lucky: *WP error* indicates that the write protect notch wasn't aligned properly. Widen the hole slightly with the paper punch and try again.

I/O error will probably mean that the index holes on the front and back aren't lined up properly. Check this by slowly rotating the disk until the index hole on the disk shows through. If it doesn't show through clearly, widen the offending hole in the outside cover slightly. Then try to format it again; if it still gives an I/O error, put a write protect notch on the second side. Unless you have a disk utility capable of "locking out" bad sectors, don't bother trying to use the *other side.*

One further characteristic of the disk: the white media inside the disk lubricates and cleans the diskette. Foreign matter is trapped on its surface. When the disk is used on the *other side*, the reverse process happens. If you keep your disks in a clean environment, this will be of little concern. Users who subject their disks to cigarette smoke, dust, dirt or excessive use may want to reconsider, or clean up their act. *(end)*

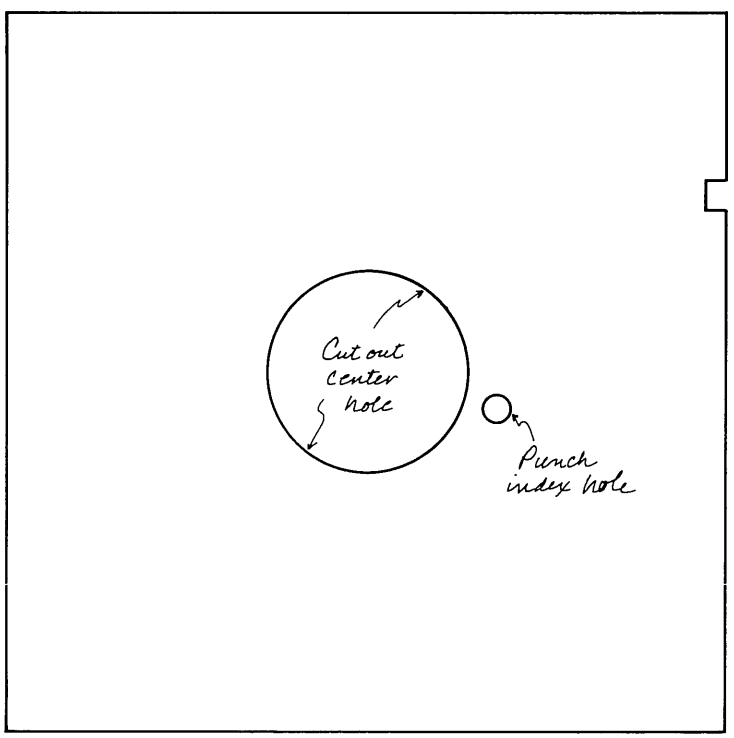


Figure 1. Template

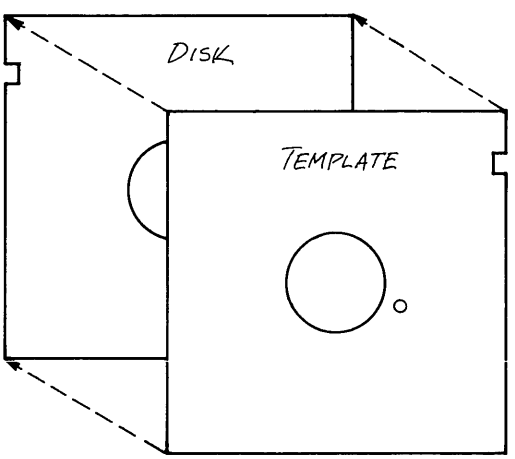
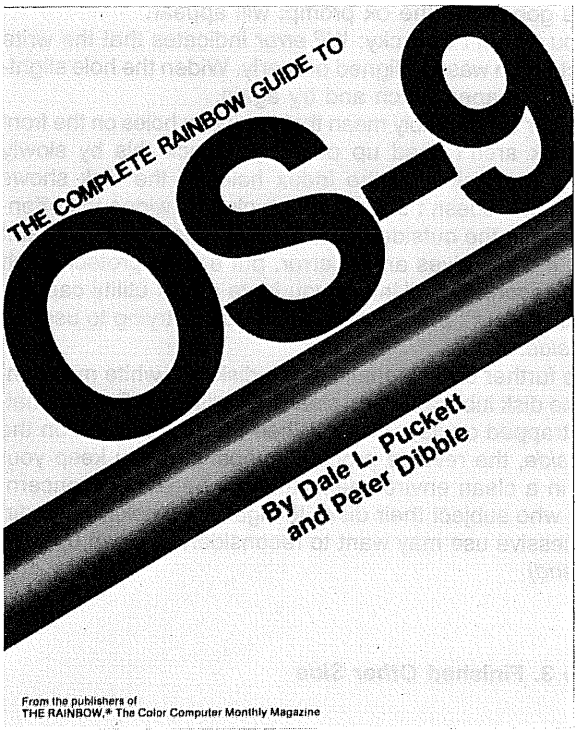


Figure 2. Line-Up

More than a book . . .

A MILESTONE

THE COMPLETE RAINBOW GUIDE TO OS-9



Today's programmers use short modules of readable code to build complex programs. The OS-9 operating system and the high level languages it brings you make the job easy.

OS-9 has so many things going for it that you need a guide as comprehensive and thorough as *The Complete Rainbow Guide To OS-9* to show you how to talk to OS-9 and realize the potential of this extremely efficient implementation of the UNIX operating philosophy.

Co-authored by Dale L. Puckett and Peter Dibble — two of the foremost authorities on OS-9 — *The Complete Rainbow Guide to OS-9* demystifies the dynamic operating system that gives the Color Computer more power and flexibility than many of the high- cost computers on the market . . . and gives you the ability and confidence to reach new programming heights.

With *The Complete Rainbow Guide To OS-9*, you will be prepared to take full advantage of the multitasking system that is setting new standards for Color Computer programming. For only $19.95!

Also Available!

The Rainbow Guide To OS-9 Disk. An adjunct to and complement to the book. You'll want the book for the tutorials, and the package of two disks to save the many hours of typing in lengthy programs.

Two-Disk Package, $31

☐ Please send me *The Complete Rainbow Guide To* OS-9 for $19.95.*

☐ Please send me The Rainbow Guide To OS-9 Disk (a package of two disks) for $31.* Does not include book.

Name _____

Address _____

City _____ State _____ ZIP _____

☐ My check in the amount of _____ is enclosed.

☐ VISA ☐ MasterCard ☐ American Express

Account Number _____

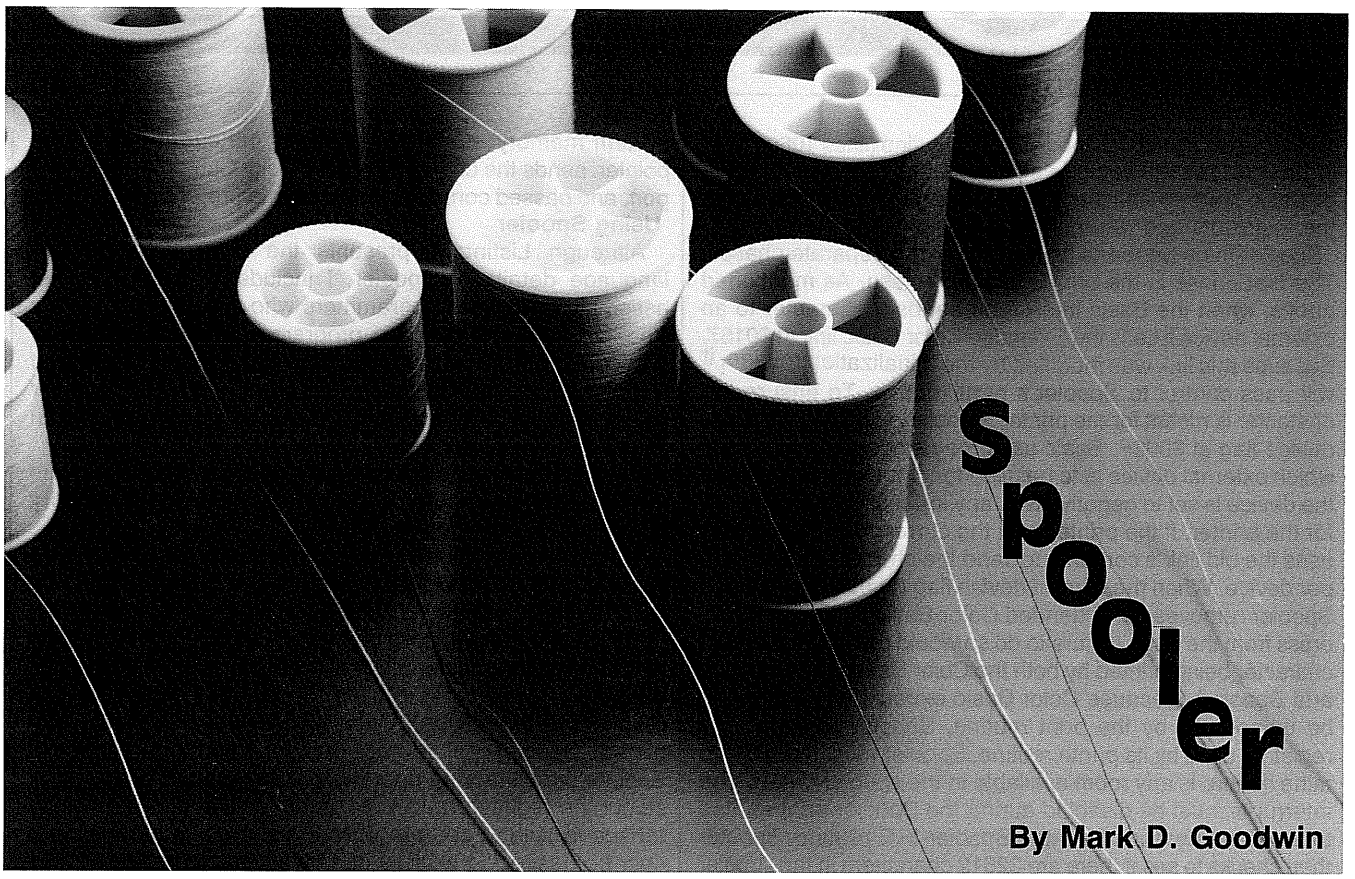
Interbank No. (MC Only) _____

Card Expiration Date _____

Falsoft, Inc., The Falsoft Building, 9529 U.S. Highway 42, P.O. Box 385, Prospect, KY 40059

*Add $1 shipping and handling per book. Canada and Mexico add $2. All other foreign countries add $4. Allow 4 weeks for delivery. KY residents add 5% sales tax. In order to hold down costs, we do not bill.

OS-9 is a trademark of Microware Systems Corporation.



spooler

By Mark D. Goodwin

Except for a few that operate at extremely high speeds, printers are usually the slowest link in the data processing chain. While waiting for a printer, most computers are unable to perform other tasks. When places between the computer and the printer, a printer spooler will appear to be a printer to the computer. Whenever printer operations are needed, the computer sends all printer data to the spooler. On receipt of the printer data, the spooler checks the printer status. If the printer is ready to accept data, the spooler sends the data on to the printer. Because the computer and the spooler are many times faster than the printer, this direct transfer of data will continue only for a short time. When the printer is no longer to accept data from the spooler, the spooler automatically buffers any new printer data into an internal memory buffer. When the printer is able to accept more data, the spooler sends the buffered data to the printer. If the spooler's buffer should becom empty, the spooler once again transfers any new printer data directly.

The large internal memory buffers of most printer spoolers let the computer free itself for other tasks. Most printer spoolers are fairly expensive and will often cost as much or more than the computer itself. This is praticularly true of an inexpensive computer like the Color Computer.

After checking the cost of printer spoolers, and finding out just how expensive they are, I decided I could live without one. Then I upgraded to 64K. With that addition, I wondered if i could put the extra 32K of RAM to use as a spooler buffer. After some experimentation, I produced the assembly language program which appears as Listing. By using bank selection and a primitive multitasking system, Spooler effectively simulates a printer spooler. Besides spooling printer data, Spooler can send data to the printer at any one of 15 different bauds. These bauds range from an excruciatingly slow 50 baud to a lightning fast 9600 baud.

Once Spooler is placed in protected high memory, the initialization routine (lines 300-590 of Listing 1) is used to link Spooler with the Color Basic interpreter. This routine first disables the Color Computer's interrupts. Spooler uses the Color interrupt system to perform a primitive multitasking operation. After disabling the interrupts, the initialization routing performs a 64K memory check, by switching the upper 32K RAM bank into system memory. With the 32K bank switched in, Spooler loads the 32K bank's first byte into accumulator A. This upper 32K RAM byte is inverted and resaved in the 32K bank. By subtracting the new contents of the upper 32K bank's first byte from the accumulator, Spooler can determine if the Color Computer has 64K of RAM available. If so, the accumulator will hold a non-zero result. With the memory check completed, the initialization routine switches the Color Basic ROMs back into the system. Then the initialization routine aborts if the upper 32K bank wasn't found. Otherwise, the initialization routine sets a flag indicating a successful memory check.

After completing a successful memory check, the initialization routine sets the links with the Color Basic interpreter. To operate successfully, Spooler must be linked with Color Basic in two ways: to alert Spooler when a character is being sent to the printer, Spooler uses the Extended Color Basic link located at $0167. This Extended Color Basic link is called whenever a character is sent to an external device. The initialization routine saves the contents of the link and saves a junp to Spooler's print routine (lines 700-1020 of Listing 1) as the link's new contents. Second, Spooler must be able to send buffered characters to the printerr without interfering with normal Color Basic operations. This multitasking operation is accomplished by using the normal interrupt (IRQ) vector at $010C. The Color Computer's 6809 microporcessor performs a jump to the IRQ vector whenever an IRQ interrupt occurs. Under normal operation the Color Computer generates an IRQ interrupt 60 times a second. By linking with the IRQ vector, Spooler can send buffered characters to the printer at this rate. Spooler saves the contents of the IRQ vec-

tor and saves a jump to Spooler's IRQ routine (lines 1060–1630 of Listing 1) as the vector's new contents. With the two Color Basic links established, the initialization routine enables the interrupts and returns to the Basic interpreter.

Spooler them performs its function in two separate phases. The first phase is the print character phase. As mentioned above, when the Basic interpreter sends a character to an external device it calls the Extended Color Basic link at $0167. Because this link was modified by the initialization routine, it will pass control to Spooler's print routine. To ensure the character is meant for the printer, Spooler examines the Basic device flag at $006F. Basic uses the device flag to indicate which external device is to be used for input or output. When the device is set to negative two, any input or output is meant for the printer. If the printer isn't the current device, Spooler uses the old link's contents to send the character to the proper device. When the current device flag *is* set for the printer, Spooler removes the Extended Color Basic link's return address from the stack (failure to do so would result in the printer character being printed by both the Color Basic printer routine and Spooler). Because Color Basic expects all registers to be unmodified by the print routine, Spooler saves all the registers used by its printe routine. Spooler must them determine if there is any room available in the spooler buffer. The interrupts must be disabled during the Spooler print routine to avoid interference from the Spooler IRQ routine. Spooler then checks to see if there are 32512 characters in the spooler buffer, a full buffer. If so, Spooler must enter a looping phase until the IRQ routine sends at least one character to the printer. Because Spooler has diabled the interrupts, Spooler's IRQ routine is turned off. Spooler enables the interrupts and waits for one interrupt before looping back to the buffer check routine. When there is enough room available in the spooler buffer, Spooler updates and saves the number of buffered characters counter, retrieves the next buffer location pointer, switches the RAM bank into system memory, saves the printer character in the RAM buffer, switches the ROMs back into system memory, updates and saves the next buffer location pointer, updates the Color Basic line position, restores the registers, and returns to the calling routine.

When characters are in the spooler buffer, Spooler needs a way to sent them to the printer. Spooler performs this phase of its operation via the IRQ routine. When the 6809 microprocessor detects an IRQ a jump is made to the IRQ vector at $010C. Because the initialization routine has modified the IRQ vector's contents, all jumps to the IRQ vector will in turn pass control to the Spooler's IRQ routine. This IRQ routine checks to see if Spooler has been turned off. If so, the IRQ routine passes control to the Basic interpreter's IRQ routine. Next, the IRQ routine checks to see if the printer is ready to accept data. The printer status check is performed by examining bit 0 of address $FF22. If this bit is set to a logic one, the printer isn't ready and the IRQ routine passes control to the Basic interpreter's IRQ routine. If the printer is ready, the IRQ routine must check for buffered characters. Spooler's print routine maintains a count of the number of buffered characters; therefore, the IRQ needs only examine this counter for buffered characters. If there is at least one buffered character, the IRQ routine saves an updated number of characters counter, retrieves the current character pointer, switches the RAM bank into system memory, recovers a character from the RAM bank, switched the ROMs back into system memory, updates and saves the current character pointer, sends the character out the Color Computer's RS-232 port, and passed control to the Basic interpreter's IRQ routine.

Using Spooler

Although Listing 1 presents the complete assembly language details of Spooler, I included it only to assist assembly language programmers who may wish to modify Spooler for use in their own programs. To use Spooler with any bersion of Basic, Listing 2 is a Basic POKE version of Spooler. This version will automatically execute Spooler without the bother associated with loading and executing an assemble language program.

Be sure to save a few copies of Spooler before trying it out; the Basic program automatically erases itself from memory via the new command in line 1610.

With copies of Spooler saved on tape or disk, sinply type RUN and Enter to execute the program. Spooler starts by reserving high memory and POKEing the assembly language program into the reserved memory area. Spooler them calls the initialization routine. Spooler will determine if the initialization was successful; if it failed, Spooler will display an appropriate error mesage. With a successful initialization completed, Spooler will prompt you for the baud. To make Spooler's multitasking operation as transparent as possible, select the highest baud your printer or serial-to-parallel interface is able to operate with. If a change in baud should ever be desired, a change can be accomplished by POKEing the appropriate baud code into memory location 149 ($0095). The baud codes used by Spooler are illustrated in Table 1.

Table 1. Baud Codes

Baud	Code (Stored at location 149)
500	
75	1
110	2
135	3
150	4
300	5
600	6
1200	7
1800	8
2000	9
2400	10
3600	11
4800	12
7200	13
9600	14

Because I wanted the ability to reset the spooler and turn it on or off from a Basic program, I incorporated these abilities into the spooler program. Listing 3 presents the code to turn the spooler off. Listing 4 is the code to turn the spooler on. Finally, Listing 5 presents the code for resetting the spooler. All three programs are short and cam be easily incorporated into a Basic program; these operations can be performed at the Basic command mode level by somply entering the code with a line number.

A word of caution is in order about these three short programs. To operate correctly, all three expect the Spooler program to be located directly above the Basic high memory pointer. Therefore, always load other high memory routines before running the Basic POKE version of Spooler. Failure to do so will not affect Spooler's operation, but the three utility programs will not function correctly.

Listing 1. Spooler

```
00100 *
00110 * 64K CoCo Printer Spooler 1.0.0
00120 * Copyright (c) 1984 By Mark D. Goodwin
00130 *
0095  00140 BAUD    EQU   $0095
006F  00150 DEVICE  EQU   $006F
010C  00160 IRQVEC  EQU   $010C
009C  00170 LPOS    EQU   $009C
0098  00180 MPOS    EQU   $0098
0167  00190 OUTVEC  EQU   $0167
FFDF  00200 SRAM    EQU   $FFDF
FFDE  00210 SROM    EQU   $FFDE
      00220         ORG   $7EBF
      00230 *
      00240 * Initialization Flag
      00250 *
7EBF  00 00260 IFLG   FCB   0
      00270 *
      00280 * Initialization Routine
      00290 *
7EC0 1A 50  00300 INIT   ORCC  #$50   Disable the interrupts
7EC2 B7 FFDF 00310       STA   SRAM   Select RAM bank
7EC5 B6 8000 00320       LDA   $8000  A=RAM byte
7EC8 43     00330       COMA         Invert it
7EC9 B7 8000 00340       STA   $8000  RAM bank
7ECC B0 8000 00350       SUBA  $8000   available?
7ECF B7 FFDE 00360       STA   SROM   Select ROM bank
7ED2 26 2B   00370       BNE   INIT0  Jump if not 64K CoCo
7ED4 6A 8C EB 00380      DEC   IFLG,PCR Flag 64K CoCo
7ED7 CE 0167 00390       LDU   #OUTVEC Save the old
7EDA 30 8D 003F 00400    LEAX  OOUT,PCR output character
7EDE 8D 22   00410       BSR   INIT1    routine's address
7EE0 CE 010C 00420       LDU   #IRQVEC Save the
7EE3 30 8D 00D3 00430    LEAX  OIRQ,PCR old IRQ
7EE7 8D 19   00440       BSR   INIT1    routine's address
7EE9 86 7E   00450       LDA   #$7E    Set
7EEB B7 0167 00460       STA   OUTVEC   the new
7EEE B7 010C 00470       STA   IRQVEC   routines'
7EF1 30 8D 001E 00480    LEAX  PRINT,PCR  addresses
7EF5 BF 0168 00490       STX   OUTVEC+1
7EF8 30 8D 0062 00500    LEAX  IRQ,PCR
7EFC BF 010D 00510       STX   IRQVEC+1
7EFF 1C AF   00520 INIT0 ANDCC #$AF    Enable the interrupts
7F01 39      00530       RTS           Return to Basic
7F02 C6 03   00540 INIT1 LDB   #3      B=Loop count
7F04 A6 C0   00550 INIT2 LDA   ,U+     Save
7F06 A7 80   00560       STA   ,X+      a byte
7F08 5A      00570       DECB          Loop
7F09 26 F9   00580       BNE   INIT2    till done
7F0B 39      00590       RTS           Return
             00600 *
             00610 * Spooler Variables and Pointers
             00620 *
7F0C 00      00630 SFLG  FCB   0
7F0D 0000    00640 SLEN  FDB   0
7F0F 8000    00650 CPTR  FDB   $8000
7F11 8000    00660 NPTR  FDB   $8000
             00670 *
             00680 * New Output Character Routine
             00690 *
7F13 34 04   00700 FRINT PSHS  B       Save B
7F15 D6 6F   00710       LDB   DEVICE  Is the device
7F17 C1 FE   00720       CMPB  #-2      the printer?
7F19 35 04   00730       PULS  B       Get B
```

Coming in next issue what we have all been waiting for

the answer to who won Bill Barden

along with

- *the Data Gatherer Part VI*
- *Defuser*
- *Reviews*

and more!

```
7F1B 27   03       00740          BEQ     PRINTO  Jump if printer
7F1D               00750 OOUT     RMB     3       Use the old routine
7F20 32   62       00760 PRINTO   LEAS    2,S     Clean up the stack
7F22 34   13       00770          PSHS    CC,A,X  Save regs
7F24 1A   50       00780 PRINT1   ORCC    #$50    Disable the interrupts
7F26 AE   8C E4    00790          LDX     SLEN,PCR X=Number of characters
7F29 8C   7F00     00800          CMPX    #32512  Jump if the
7F2C 26   05       00810          BNE     PRINT2   buffer isn't full
7F2E 1C   AF       00820          ANDCC   #$AF    Enable the interrupts
7F30 13            00830          SYNC            Wait till IRQ
7F31 20   F1       00840          BRA     PRINT1  Loop
7F33 30   01       00850 PRINT2   LEAX    1,X     Bump number of characters
7F35 AF   8C D5    00860          STX     SLEN,PCR Save it
7F38 AE   8C D6    00870          LDX     NPTR,PCR X=Next location pointer
7F3B B7   FFDF     00880          STA     SRAM    Select RAM bank
7F3E A7   80       00890          STA     ,X+     Save the character
7F40 B7   FFDE     00900          STA     SROM    Select ROM bank
7F43 8C   FF00     00910          CMPX    #$FF00  Jump if not
7F46 26   03       00920          BNE     PRINT3   end of buffer
7F48 8E   8000     00930          LDX     #$8000  X=Start of buffer
7F4B AF   8C C3    00940 PRINT3   STX     NPTR,PCR Save the next location
7F4E 81   0D       00950          CMPA    #13     Jump if it's
7F50 27   08       00960          BEQ     PRINT4   a carriage return
7F52 0C   9C       00970          INC     LPOS    Bump line position
7F54 96   9C       00980          LDA     LPOS    A=Line position
7F56 91   9B       00990          CMPA    MPOS    Jump if not
7F58 26   02       01000          BNE     PRINT5   end of line
7F5A 0F   9C       01010 PRINT4   CLR     LPOS    Zero line position
7F5C 35   93       01020 PRINT5   PULS    CC,A,X,PC Get regs and return
                   01030 *
                   01040 * New IRQ Routine
                   01050 *
7F5E 6D   8C AB    01060 IRQ      TST     SFLG,PCR Jump if
7F61 26   57       01070          BNE     OIRQ     spooler is off
7F63 B6   FF22     01080          LDA     $FF22   Jump if
7F66 44            01090          LSRA             printer
7F67 25   51       01100          BCS     OIRQ     isn't ready
7F69 AE   8C A1    01110          LDX     SLEN,PCR X=Number of characters
7F6C 27   4C       01120          BEQ     OIRQ    Jump if no characters
7F6E 30   1F       01130          LEAX    -1,X    Dec number of characters
7F70 AF   8C 9A    01140          STX     SLEN,PCR Save it
7F73 AE   8C 99    01150          LDX     CPTR,PCR X=Character pointer
7F76 B7   FFDF     01160          STA     SRAM    Select RAM bank
7F79 A6   80       01170          LDA     ,X+     A=Character
7F7B B7   FFDE     01180          STA     SROM    Select ROM bank
7F7E 8C   FF00     01190          CMPX    #$FF00  Jump if not
7F81 26   03       01200          BNE     IRQO     end of buffer
7F83 8E   8000     01210          LDX     #$8000  X=Start of buffer
7F86 AF   8C 86    01220 IRQO     STX     CPTR,PCR Save character pointer
7F89 30   8D 0046  01230          LEAX    BTAB,PCR X=Baud rate counts table
7F8D D6   95       01240          LDB     BAUD    B=Baud rate code
7F8F 3A            01250          ABX             Adjust
7F90 3A            01260          ABX              the table
7F91 3A            01270          ABX               pointer
7F92 E6   80       01280          LDB     ,X+     B=Short count
7F94 E7   8D 002E  01290          STB     CNT2,PCR Save it
7F98 AE   84       01300          LDX     ,X      X=Long count
7F9A AF   8D 0025  01310          STX     CNT1,PCR Save it
7F9E 8D   1D       01320          BSR     IRQ2    Set RS232 line high
7FA0 5F            01330          CLRB            Send the
7FA1 8D   1C       01340          BSR     IRQ3     start bit
7FA3 21   FE       01350          BRN     *       Delay
7FA5 21   FE       01360          BRN     *        for the
7FA7 21   FE       01370          BRN     *         baud rate
7FA9 C6   08       01380          LDB     #8      B=Number of bits
7FAB 34   04       01390 IRQ1     PSHS    B       Save count
7FAD 5F            01400          CLRB            Zero B
7FAE 44            01410          LSRA            Carry=Next bit
7FAF 59            01420          ROLB            Bit 0 B=Next bit
7FB0 58            01430          ASLB            Bit 1 B=Next bit
7FB1 8D   0C       01440          BSR     IRQ3    Send the bit value
7FB3 35   04       01450          PULS    B       Get the count
7FB5 5A            01460          DECB            Loop
7FB6 26   F3       01470          BNE     IRQ1     till done
7FB8 8D   03       01480          BSR     IRQ2    Send stop bit
7FBA               01490 OIRQ     RMB     3       Use old IRQ routine
7FBD C6   02       01500 IRQ2     LDB     #2      B=High value
7FBF F7   FF20     01510 IRQ3     STB     $FF20   Send the bit value
7FC2 8E   0000     01520          LDX     #0      X=Long count
          7FC3     01530 CNT1     EQU     *-2
7FC5 C6   00       01540          LDB     #0      B=Short count
          7FC6     01550 CNT2     EQU     *-1
7FC7 26   05       01560          BNE     IRQ5    Jump if short count
7FC9 30   1F       01570 IRQ4     LEAX    -1,X    Delay for
7FCB 26   FC       01580          BNE     IRQ4     the baud rate
7FCD 39            01590          RTS             Return
7FCE 5A            01600 IRQ5     DECB            Delay
7FCF 12            01610          NOP              for the
7FD0 26   FC       01620          BNE     IRQ5      baud rate
7FD2 39            01630          RTS             Return
                   01640 *
                   01650 * Baud Rate Counts Table
                   01660 *
7FD3       00      01670 BTAB     FCB     0
7FD4       08B7    01680          FDB     2231
7FD6       00      01690          FCB     0
7FD7       05CD    01700          FDB     1485
7FD9       00      01710          FCB     0
7FDA       03F2    01720          FDB     1010
7FDC       00      01730          FCB     0
7FDD       0339    01740          FDB     825
7FDF       00      01750          FCB     0
7FE0       02E3    01760          FDB     739
7FE2       00      01770          FCB     0
7FE3       016E    01780          FDB     366
7FE5       00      01790          FCB     0
7FE6       00B4    01800          FDB     180
7FE8       63      01810          FCB     99
7FE9       0000    01820          FDB     0
7FEB       3F      01830          FCB     63
7FEC       0000    01840          FDB     0
7FEE       38      01850          FCB     56
7FEF       0000    01860          FDB     0
7FF1       2E      01870          FCB     46
7FF2       0000    01880          FDB     0
7FF4       1C      01890          FCB     28
7FF5       0000    01900          FDB     0
7FF7       00      01910          FCB     0
7FF8       0011    01920          FDB     17
7FFA       00      01930          FCB     0
7FFB       0009    01940          FDB     9
7FFD       06      01950          FCB     6
7FFE       0000    01960          FDB     0
           7EC0    01970          END     INIT
00000 TOTAL ERRORS
```

BAUD	0095	IFLG	7EBF	IRQ0	7F86	IRQVEC	010C	OUTVEC	0167	PRINTO	7F20
BTAB	7FD3	INIT	7EC0	IRQ1	7FAB	LPOS	009C	PRINT	7F13	PRINT1	7F24
CNT1	7FC3	INIT0	7EFF	IRQ2	7FBD	MPOS	009B	PRINT2	7F33	PRINT3	7F4B
CNT2	7FC6	INIT1	7F02	IRQ3	7FBF	NPTR	7F11	PRINT4	7F5A	PRINT5	7F5C
CPTR	7F0F	INIT2	7F04	IRQ4	7FC9	OIRQ	7FBA	SFLG	7F0C	SLEN	7F0D
DEVICE	006F	IRQ	7F5E	IRQ5	7FCE	OOUT	7F1D	SRAM	FFDF	SROM	FFDE

Listing 2. Basic Version

```
0 '
1 ' 64K COCO SPOOLER 1.0.0
2 ' BASIC POKE VERSION
3 ' COPYRIGHT (C) 1984 BY MARK D. GOODWIN
4 '
5 ' SET UP THE DISPLAY
6 '
10 CLS
20 PRINT"      64K COCO SPOOLER 1.0.0"
30 PRINT" COPYRIGHT 1984 BY MARK GOODWIN"
40 PRINT"--------------------------------"
50 PRINT"               PLEASE"
60 PRINT"                WAIT"
70 PRINT"               FOR ME"
80 PRINT"                 TO"
90 PRINT"            INITIALIZE"
100 PRINT"           THE  SPOOLER"
197 '
198 ' RESERVE HIGH MEMORY
199 '
200 CLEAR200,PEEK(39)*256+PEEK(40)-320
297 '
298 ' POKE THE SPOOLER INTO HIGH MEMORY
299 '
300 M=PEEK(39)*256+PEEK(40)+1
310 FORI=0TO320:READJ:POKEM+I,J:NEXTI
397 '
398 ' CALL THE INITIALIZATION ROUTINE
399 '
400 EXECM+1
497 '
498 ' CLEAR PARTIAL SCREEN
499 '
500 PRINT@128,;:FORI=1TO6:PRINT:NEXT:PRINT@128,;
597 '
598 ' BRANCH IF 64K MACHINE
599 '
600 IFPEEK(M)<>0THEN900
697 '
698 ' DISPLAY ABORT MESSAGE
699 '
700 PRINT"          UPPER RAM BANK"
710 PRINT"         ISN'T AVAILABLE!"
720 PRINT
730 PRINT"          INITIALIZATION"
740 PRINT"         HAS BEEN ABORTED"
797 '
798 ' RESTORE HIGH MEMORY
799 '
800 CLEAR200,PEEK(39)*256+PEEK(40)+322
810 GOTO1600
897 '
898 ' SET UP BAUD RATES ARRAY
899 '
900 DIMB(14):FORI=0TO14:READB(I):NEXT
997 '
998 ' DISPLAY THE BAUD RATES
999 '
1000 FORI=0TO7:PRINT"      ";:IFB(I)<100THENPRINT"  ";
     ELSEIFB(I)<1000THENPRINT" ";
1010 PRINTB(I);:IFI+8=15THENPRINTELSEPRINT"          "
     B(I+8)
1020 NEXT
1097 '
1098 ' GET THE BAUD RATE
1099 '
1100 PRINT@416,CHR$(13);:PRINT@416,;:INPUT" ENTER
     THE BAUD RATE";B
1197 '
1198 ' IS IT A LEGITIMATE RATE?
1199 '
1200 L=0:H=14
1210 M=(H+L)/2
1220 IFB(M)=B THEN1300
1230 IFB<B(M)THENH=M-1ELSEL=M+1
1240 IFL<=H THEN1210ELSE1100
1297 '
1298 ' SET THE BAUD RATE
1299 '
1300 POKE149,M
1397 '
1398 ' REMOVE INITIALIZATION ROUTINE
1399 '
1400 CLEAR200,PEEK(39)*256+PEEK(40)+78
1497 '
1498 ' DISPLAY COMPLETION MESSAGE
1499 '
1500 PRINT@128,;:FORI=1TO10:PRINT:NEXT:PRINT@128,;
1510 PRINT"          INITIALIZATION"
1520 PRINT"           IS COMPLETE!"
1597 '
1598 ' ERASE THIS PROGRAM
1599 '
1600 PRINT
1610 NEW
1697 '
1698 ' SPOOLER POKE VALUES
1699 '
1700 DATA0,26,80,183,255,223,182,128
1701 DATA0,67,183,128,0,176,128,0
1702 DATA183,255,222,38,43,106,140,232
1703 DATA206,1,103,48,141,0,63,141
1704 DATA34,206,1,12,48,141,0,211
1705 DATA141,25,134,126,183,1,103,183
1706 DATA1,12,48,141,0,30,191,1
1707 DATA104,48,141,0,98,191,1,13
1708 DATA28,175,57,198,3,166,192,167
1709 DATA128,90,38,249,57,0,0,0
1710 DATA128,0,128,0,52,4,214,111
1711 DATA193,254,53,4,39,3,61,162
1712 DATA30,50,98,52,19,26,80,174
1713 DATA140,228,140,127,0,38,5,28
1714 DATA175,19,32,241,48,1,175,140
1715 DATA213,174,140,214,183,255,223,167
1716 DATA128,183,255,222,140,255,0,38
1717 DATA3,142,128,0,175,140,195,129
1718 DATA13,39,8,12,156,150,156,145
1719 DATA155,38,2,15,156,53,147,109
1720 DATA140,171,38,87,182,255,34,68
1721 DATA37,81,174,140,161,39,76,48
1722 DATA31,175,140,154,174,140,153,183
1723 DATA255,223,166,128,183,255,222,140
1724 DATA255,0,38,3,142,128,0,175
1725 DATA140,134,48,141,0,70,214,149
1726 DATA58,58,58,230,128,231,141,0
1727 DATA46,174,132,175,141,0,37,141
1728 DATA29,95,141,28,33,254,33,254
1729 DATA33,254,198,8,52,4,95,68
1730 DATA89,88,141,12,53,4,90,38
1731 DATA243,141,3,255,0,255,198,2
1732 DATA247,255,32,142,0,0,198,0
1733 DATA38,5,48,31,38,252,57,90
1734 DATA18,38,252,57,0,8,183,0
1735 DATA5,205,0,3,242,0,3,57
1736 DATA0,2,227,0,1,110,0,0
1737 DATA180,99,0,0,63,0,0,56
1738 DATA0,0,46,0,0,28,0,0
1739 DATA0,0,17,0,0,9,6,0
1740 DATA0
1797 '
1798 ' BAUD RATE VALUES
1799 '
1800 DATA 50,75,110,135,150,300,600,1200,1800,
     2000,2400,3600,4800,7200,9600
```

Listing 3. Turn Off

```
0 '
1 ' TURN SPOOLER ON
CODE
2 '
10
POKEPEEK(39)*256$PEEK(
40)$1,0
```

Listing 4. Turn On

```
0 '
1 ' TURN SPOOLER OFF
CODE
2 '
10
POKEPEEK(39)*256$PEEK(
40)$1,1
```

Listing 5. Reset

```
0 '
1 ' RESET SPOOLER CODE
2 '
10
M$PEEK(39)*256$PEEK(40)$
1:POKEM,1:POKEM$1,0:PO-
KEM$2,0:POKEM$3,128:PO-
KEM$4,0:POKEM$5,128:PO-
KEM$6,0:POKEM,0
```

****
*** NEWS FROM UNDER COLOR **** NEWS FOR UNDER THE COLORS ***
********** Compiled May 23, 1985 **********
** Send news to: Under Color, Box 6809, Roxbury, VT 05669 **
********** CompuServe 70136,1257 **********
****

WHERE HAVE WE BEEN? At Rainbowfest and other places. Trying to meet our every-two-weeks schedule has been tough, especially with summer travel. Our apologies. WE'LL TRY TO DO BETTER. More Rainbowfest info will be printed next time.

* * * * *

TSK, TSK DEPARTMENT: Colorware is UPSET WITH UNDER COLOR for publishing the integrated circuit types used in its COCOMAX CARTRIDGE (see Issue #6). It seems they're outraged that we've VIOLATED THEIR RIGHTS and HURT THEIR BUSINESS. Well, UNDER COLOR is outraged that companies still keep HIDING FROM THEIR OWN CUSTOMERS, and business at Rainbowfest was brisk, hardly hurting at all. Nevertheless, Under Color has asked Colorware to COME OUT FIGHTING if they think they're right ... and we're OFFERING SPACE right in these pages. So far? SILENCE. Perhaps Under Color is right in calling HARDWARE AND SOFTWARE PROTECTION TOTALLY INDEFENSIBLE. Well, Colorware? Can you defend it?

* * * * *

A curious letter has been published recently in other CoCo mags about the 6847-T1 LOWERCASE VIDEO GENERATOR in the new Korean Color Computers. The letter states the -T1 is a pin-compatible, drop-in replacement for your present 6847. IT ISN'T ... DON'T TRY IT (that's why we didn't publish that letter!). By the way, want to know if you've got the new lowercase generator in your machine? Look for an integrated circuit with TANDY'S CUSTOM NUMBER XC80652P. (Read on for other KoKo changes). Under Color has a "Rev B" machine IN ITS HANDS NOW, and will report soon.

* * * * *

WANT FULL DATA ON THAT 6847-T1 LOWERCASE VDG? Write to Motorola Semiconductors, 3501 Ed Bluestein Blvd., Austin TX 78721 and ask for PUBLICATION #ADI 1048 for product type MC6847T1. It's called "Advance Information", and IT'S FREE.

* * * * *

In a BADLY SHRINKING COCO MARKET, many companies are ABANDONING THE BABY of the Tandy line. Under Color spoke to two firms soon to be DROPPING COCO SUPPORT, and it's certain that RGS MICRO IS OUT OF THE SELLING BUSINESS too. Under Color received a form letter from RGS indicating that they are KEEPING THEIR RESEARCH AND DEVELOPMENT department, but scuttling the rest. Where does this leave RGS customers? We'll report if there is anything to worry about. SKYLINE MARKETING has disappeared and its stock is being closed out by another company.

* * * * *

****
*** NEWS FROM UNDER COLOR **** NEWS FOR UNDER THE COLORS ***
********** Compiled May 23, 1985 **********
** Send news to: Under Color, Box 6809, Roxbury, VT 05669 **
********** CompuServe 70136,1257 **********
****

HARD DISK FANS -- especially those worried about the retreat of RGS Micro -- may be in for relief from P.R.O. Systems of Kansas City. Their new, FULLY SUPPORTED HARD DISK SUB-SYSTEM is about to be introduced, and was seen at Rainbowfest. The brainchild of the well-respected Steve Odneal, the first units survived their ordeal as airline baggage and performed flawlessly.

* * * * *

PORTABLE COCO APPEARS! Again thanks to Steve Odneal, veteran improver and first with a CoCo-based Flex operating system, there has appeared a FULLY PORTABLE COLOR COMPUTER COMPLETE WITH DISK DRIVES AND MONITOR. The portable was on view at the May 17-19 Rainbowfest in Chicago, but it's ONLY A PROTOTYPE at this stage. Odneal will judge user interest from the show.

* * * * *

WHAT?? A COCO KIT?? Radio Shack rumors continue, and the latest development involves a CoCo2 in kit form. News is sketchy at this point, but certain ELECTRONIC DROPPINGS LEFT IN A TANDY GARBAGE BIN suggest a CoCo kit is coming.

* * * * *

Looking for SERVICE MANUALS? Many manuals can be ordered from Radio Shack National Parts by using an "MS" prefix. Korean KoKo 26-3134A is MS2603134A, for example. Try it. Speaking of service, among the MINOR CHANGES in the "A" KoKos is the use of a processor marked MX5560, which is a Motorola 6809EP, a Hitachi 6809EP or 68A09EP. Color Basic is TC1001 and Extended Basic is TC1002; unknown ROM TC1004 is also marked. Diodes previously used with IC7 (the SALT) are internal to the part after date code 8417 (17th week of 1984).

* * * * *

It's CHECKSUM TIME. Here's an update of the checksums for the various versions of the Basic and Extended Basic ROMs:

Color Basic 1.0 ... 9505 Color Basic 1.1 ... E54B
Color Basic 1.2 ... 5917 Color Basic 1.3 ... 5342
Extended Basic 1.0 ... 2206 Extended Basic 1.1 ... 3338
Extended Basic 1.1 ... 9906 (some versions give this one!)

Know the DISK BASIC checksums for various versions and alternate DOSes? Under Color would like to publish them here.

* * * * *

Off Color: Off Color: Off Color: When In Doubt, Try Prayer

By Jeffrey S. Parker

So, someone finally went and did it. They manufactured a device which can make my CoCo run CP/M. And (will wonders never cease?) my editor wants me to review it. I can just imagine it now—Little H's for ladders being run up and down by the letter "p," while X's try madly to capture them.

It arrives a few days later—with the COD label crimsoning at me (who says reviewers get the best software free?). Plugging it in is no problem; it goes into the cartridge port with only a minimum of force. It doesn't break, so we are off to a good start. On goes the disk drive controller, and my CoCo now looks as if it has hit a tree and broken off a stub wing attempting to land.

Okay, Wordstar 2.2—here we go! Whirrah, whirrah, chunk, kachunk, kachunkachunk, datippidah, datippidah . . .

There is the cursor: flashing from the middle of the screen (which is now black and white). No Wordstar. Not even a 1.1 version. A large tear forms in my right eye and courses down my cheek.

This is not an uncommon event. Someone claims to have developed the ultimate, and sends it out for review. "Sounds interesting," I remember saying to my Editor, "send it right along." And he does, with a note saying, "I need this desperately in three days or less. Help me, Computer Reviewer, you're my only hope." The image of the note fades, and I realize that the program does not work. Of course, it is not the manufacturer's fault. Lesson number 1 from computer reviewer school: the manufacturer is always right.

Then I go to do a little word processing, and the computer is now functionless. Motionless. Dark, silent, and depressing. I guess it is just a law of the Universe that my computer must break at the crucial moment. The CPU goes "phut," and that's that. Gonzo, kablooie, out to lunch. And the review due in the morning. My IBM Selectric will have nothing to do with me. "You wanted word processing," it smirks at me, "you got word processing."

My wife walks in and sees my face. Whoosh, she's gone. Moments later, she returns with a blanket, a pillow, my slippers, a cold compress, and a hot toddy . . .

"It crashed," I whisper.

"There, there, dear. I am sure your CoCo will feel better soon. Now just relax, and I'll get you something yummy to eat." She knows the storm is brewing. She can feel the ozone in the air. She knows it will break at any second.

"What kind of machine . . . "

She groans, covering her eyes.

"Do they think they can *hype* on the American public . . . "

Oh God . . . "

"With this outrageous pricetag attached . . . "

She cranes her neck wildly, searching for cover.

"And the damn thing doesn't even run!"

My wife has always been inventive and decisive, particularly in crisis situations. She walks over and kicks la machine, and whirrah, whirrah, datippidah, zippideedoodah, zipideeday, there's Wordstar.

The sun comes out. Flowers begin to bud and burst into bloom. The cat comes out of hiding. Wordstar 2.2 has arrived.

My troubles, according to the powers that be, appear to be just beginning. A program that works the way it should during the day, but refuses to work at night. How odd.

The blank grey monitor screen seems smug. The red light of my power switch is on. I have contact. The clutch is engaged, the monitor should be monitoring, the disk should be driving.

"Got any ideas?" I ask the cat, who smiles nervously, gulps, and backs out of the room.

Let us pause here to consider philosophy. I am referring, of course, to Murphy's Law. If something can go wrong, it will, at the worst possible moment, and to the greatest possible extent. Why me? Why my computer? I am now considering turning my CoCo into a SCUBA, as my eyes light on the

neighbor's pool. "Hmm. Mysterious sunken CoCo baffles authorities. Sounds good." I can hear my Selectric snickering quietly. "Fine," I say, after all the usual double-checkings of interfaces, power cords and switches. I pick up the phone and call my local Radio Shack, one of several thousand Tandy Help Centers.

"Good evening, Radio Shack."

"Hello. I think I broke my computer."

(Pause) "What computer do you have, sir, and what seems to be the trouble?"

"Are you a technician?"

"No sir, I'm the computer department manager. How can I help you?"

I explained my predicament.

"Are you sure it's plugged in, Sir?" I count to ten silently. "Hello, are you still there?"

"Yes, yes, I'm here, it's my computer that's dying! It is very definately, most assuredly, beyond a possible shadow of a reasonable doubt, most certainly, absolutely, positively, plugged in."

"You're sure now? Check again." I count to ten. In Arabic. "Hello? Hello? Are you there, Sir?"

"Yes," I sigh. "I'm still here. It's plugged in."

"Hmm. Hmm. Well then, it could be a fuse problem."

I perk up. "A fuse, you say?"

"Yes, it could be that. Of course, it isn't user serviceable, in any event . . . "

"No. Of course not."

"If you'd like to bring it in, I'd be happy to take a look at it for you, Sir."

"It's out of warranty."

" . . . for a twenty-five dollar diagnostic fee, applicable towards the work, of course."

"I'm kind of in a desperate hurry . . . "

"Well, of course, Sir, I understand. I'll put it on rush, and I'm sure that if it doesn't need any back-ordered parts, I could have it back for you pronto. Three weeks, at the outside."

"Three weeks . . . "

"Yes. Now, your name, Sir?"

"Click."

So, there it sits, and here I stand, an impasse. What's a reviewer to do? A bright idea flashes on in my head, and I dig into a lower desk drawer for the manual. On consulting the problem section, I see two suggestions:

1. Try plugging it in.
2. Try prayer.

I figure, what the heck, and about the time that my wife walks in, I have dropped to my knees before my computer.

"Oh no."

"Yup."

"Boom?"

"Boom."

"Prayer?"

"Yes, I've come to that."

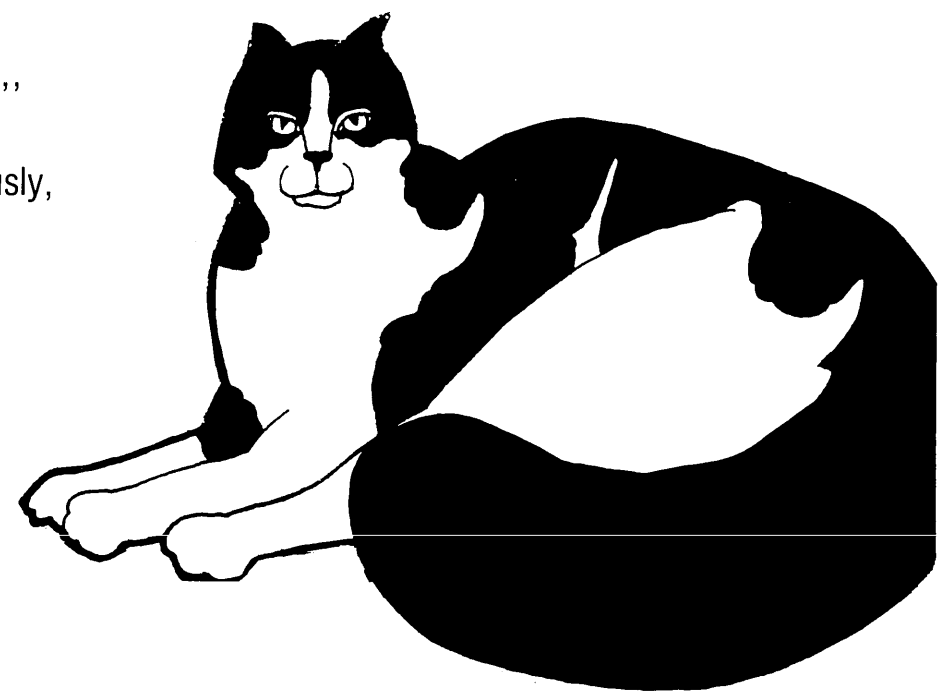
My wife walks over and kicks the machine.

The fireworks from *Love, American Style,* and *Walt Disney World* on Sunday nights go off in my forehead. Up comes the drive, on comes the printer, and my screen declares its allegiance to Microsoft. "Honey, would you like a job at Tandy?" I say struggling to my feet.

She laughs, as much from relief as from humor. One more storm averted.

In the end, it has all worked out pretty well. I am continuing my fortnightly computer ordeals. My wife, having just completed her first book, Step by Step Programming, *is hard at work on her second volume,* Kickstarting Your Computer, *or* Zen and the Art of Computer Maintenance. *Of course, my wife's kick solution does not work in every instance. There was the one time when I kicked the table, and it fell over . . . (end)*

"Got any ideas?"
I ask the cat,
who smiles nervously,
gulps,
and backs out
of the
room



• To all *TCCM* Summer Programming Project contest finalists:

I am looking for the addresses of the finalists in the programming project contest.

I am looking for your winning entries in *TCCM*'s contest. I *will* pay for your listings if you would only send me your address and price for listing. Please send addresses to: Chris Foster #8 Morris Lane, Texarkana, TX 75503

• In the February 1, 1985 DEFUSR column Terry Kepner stated that the only way to get truly random numbersfrom RND is to make repeated calls to RND for a random length of time or to write your own RND in assembler. While both of these solutions are workable, it is much easier to use an undocumented feature of RND. RND may be seeded with a starting point by passing it a negative argument. Thus it is posible to get truly random numbers by passing a special Timer variable as a negative argument. Try running the following and you will see a different number between 1 and 10 printed after each machine power-up:

```
10 DUMMY = RND(-TIMER)
20 PRINT RND(10)
```

Like Terry's suggestion of calling RND repeatedly for a random length of time, Timer will be the random length of time between machine power-up and the call to RND (-TIMER).

In addition to using a negative argument to seed RND to a random value, RND may be seeded to a specific value to get repeatable sequences of random numbers sequence of five numbers chosen between 1 and 100:

```
10 FOR I=1 TO 5          50 NEXT J
20 DUMMY=RND(-1)         60 PRINT
30 FOR J=1 TO 5          70 NEXT I
40 PRINT RND(100);
```

In summary:

RND(0)—Return a random number between 0 and 1.

RND(n)—n – 1 return a random number between 1 and n.

RND(n)—n – 0 set the random seed. Will cause subsequent calls to RND (0) or RND (n) with n – 0 to return the same sequence.

RND(–Timer)—Set the random seed to a random value causing susequent calls to RND(0) or RND(n) with n – 0 to return a true randum sequence. **Bill Welch**

• Here is a Poke that I read about somewhere and that makes the RND truely random and does not appear to alter the other operations of Color Computer:

```
POKE 280, PEEK (275).
```

This may be of especial use to Richard M. Auclair, Westfield, MA. **Paulette Grantham**

A couple of Saturdays ago I phoned you concerning 27128 EPROM, and the use of the Color Burning with disk operation.

As you instructed me, I burned two different DOS's into the 27128. Then, using a 28-pin adapter to interface the EPROM to the 24 pin disk controller ROM socket, I either switched a +5V (thru a 10K resistor) to access the upper 8K or Gnd. to access the lower 8K of the 27128 to pin 26. This I did with a SPDT mini-toggle switch. With this arrangement I can switch between two different 8K DOS's.

My other problem was to be able to access the complete 16K of the 27128 as one continuous memory. In order to access all 16K in a row, you only have to bend up pin 26 of the 27128 so that it does not go into the 28 pin adapter socket. Solder one end of a short piece of wire to the raised pin 26 and the other end to contact 37 of the connector which plugs into the CoCo. This simple modification gives you access to the full 16K required by Spectrum DOS or KDOS.

I am now using my Green Mountain Color Burner with and at the same time as my disk controller and DOS. All this is possible by using a buffer called Buffer Shack made in Laval, Quebec by Infocodec Inc..The transformation is very simple and requires modification to the three parts of the Color Burner package.

Hardware By cutting the trace on contact 36 of the burner and soldering the cut lead to contact 40. This is all that is required for the burner itself.

Assembly Language Program In the A/L Blast program all that needs changing are the three $FF40's to $FF80 (lines 890, 1000 and 1160).

Basic Program In the Basic Burner program, all the cassette I/O commands are changed to their equivalent disk I/O commands. Other commands such as DIR, RENAME, etc. can easily be added as one chooses.

With these simple modifications, which took about one hour, I have completely eliminated the slow and dull task of having to load and save all object codes to cassette before being able to use my burner. I do not, however, recommend these mods be undertaken by anyone who is not completely familiar with hardware, Assembly Language, and Basic hacking. **Edgar H. Poulin**

Clubs . . .

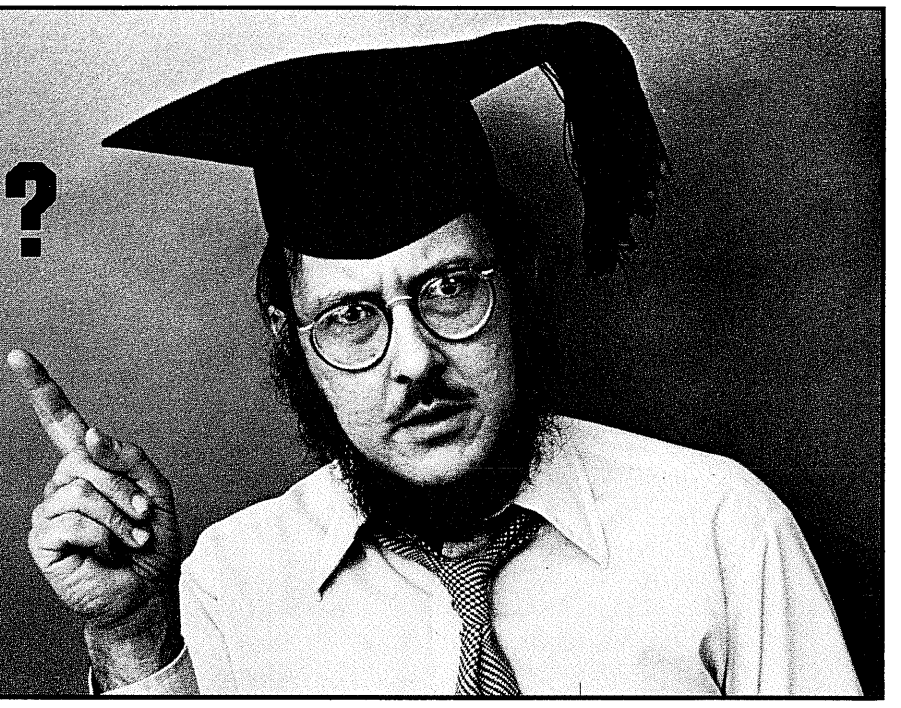
• The Mercer County Color Computer Club (MCC) meets the first Wednesday of every month at the Mercer Country Public Library, Main Branch, U.S. Route 1, Lawrenceville, N.J. Dues are $12 per year. On 12/4/84 we had our third meating with over 30 members.

• The New England Coconuts Color Computer Club is permanent mailing address is: New England Coconuts, P.O. Box 6604, Providence, RI 02940-6604. Contract Robert J. Sullivan, at (401) 739-8743, weeknights or weekends only. We ask all inquiries to include a stamped, self-addressed envelope, and no collect calls, please!

Manufacturer's Notes:

A year-long site/location study had proven to us that to better service the CoCo marketplace a consolidation of Spectrum Projects Inc. facilities and personel is in the best interests of all. Therefore, all business operations, orders, and correspondence will be located at our original business address: Spectrum Projects Inc., 93-15 86th Drive, P.O. Box 21272, Woodhaven, New York 11421, phone number (718)441-2807

FEELING HELPLESS? YOU NEED TO LEARN A LESSON!



MICRO LANGUAGE LAB: LEARNING THE 6809

Feeling at the mercy of a programmer somewhere? Mystified by a machine? My **Micro Language Lab** will give you the power to end those helpless feelings.

Here's why. The heart of *any* computer program *anywhere* is machine language. Every piece of software uses it — your favorite game, spreadsheet, word processor, data base manager, or recipe file. Every time you hit "Enter", it's working. All the languages and operating systems are created from it. Basic, Fortran, Pascal, Flex, OS-9, TRSDOS... they're 6809 machine language, the whole lot!

And you can learn the language of the 6809, the programming heart of your Color Computer, with my **Micro Language Lab.**

Not everyone can program. Writing a few lines in Basic now and then doesn't make you a programmer. But if you *can* program, then my **Micro Language Lab** will teach you — the right stuff, the right way.

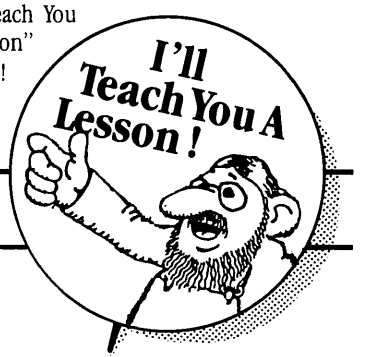
I'll teach you in 24 half-hour lessons on 12 audio cassettes, with a 220-page textbook, with data booklets, with 35 sample programs, and with a programming reference card. You'll spend 50 hours or more with my course, listening, watching, and working. And when you're done, you'll be programming your Color Computer in the 6809's language.

Stop ignoring those gnawing feelings of helplessness. The 6809 is where the power lies in your computer. Turn on that power with the **Micro Language Lab.**

- **Micro Language Lab,** $99.00 (plus $3.50 shipping and handling)

  Requires 16K Extended Basic EDTASM +

Not sure? Write or call for a Table of Contents and sample pages — and your "I'll Teach You A Lesson" button!



ALSO AVAILABLE FROM GREEN MOUNTAIN MICRO

(Add $2.50 shipping and handling to your order)

Lowerkit III

- Full-time upper and lowercase installs in 15 minutes.
- Normal and reverse video standard
- Fully compatible with all Alpha and Graphic modes

$79.95 assembled and tested
$49.95 complete kit of parts
Important! Specify Color Computer or Color Computer II

TV Buff II, clear image for video monitors, $19.95 (specify CoCo or CoCo2)

CoCoPort parallel interface (now includes plastic case), $54.95 / $44.95 kit

ColorPack ROM/RAM pack, $29.95 (specify configuration) / $19.95 kit / case $3.00 extra

64K memory upgrade kit with full instructions, $49.95 / with memory test on tape, $54.95

Color Quaver Software Music Synthesizer on tape (requires 32/64K), $19.95

Color Burner EPROM Programmer (2716/32/32A/64/64A/128, 68764/66) with software, $69.95 / $56.95 kit

User Group, Educational, Club and Dealer Discounts are available.

Green Mountain Micro
Bathory Road, Box U
Roxbury, Vermont 05669
802 485-6112
Hours: 9am - 5pm EST, Monday - Friday
COD/VISA/MASTERCARD

DYNACALC®

CoCo's Best & Fastest Spreadsheet System

ACCLAIMED BY THE EXPERTS

"DYNACALC is my choice for a CoCo spreadsheet."
Dan Downard, RAINBOW, September, 1984.

"Eat your heart out, Lotus 1-2-3!"
Scott Norman, HOT CoCo, October, 1984.

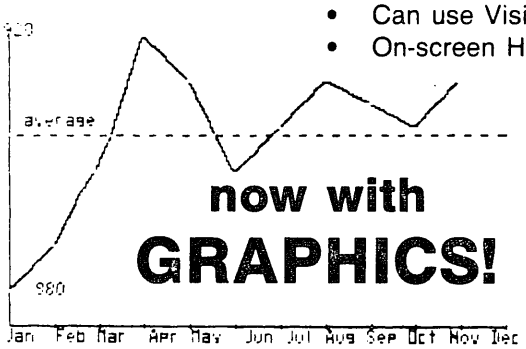
NOW ONLY $99.95

Built-in Features:

- 51 x 24 Display with Lower Case
- Super-fast Smart Screen Refresh
- Auto-Repeat Keyboard Driver
- Keysaver (TM) repeats last command x times
- Disk Operating System (works just like ROM DOS)
- Easy communication with BASIC/DOS programs
- Two-way communications with PRO-COLOR-FILE ★Enhanced★
- Outputs to ASCII Word Processors like Telewriter-64
- Fast 16-Digit Arithmetic with Scientific Functions
- Summation, Mean, and Standard Deviation Functions
- Logical Functions with String & Numeric Comparison
- String locate command to navigate large worksheets
- Sort full or partial worksheet by columns or rows
- Line, Bar, Hi/Lo/Close, Circle Graphs
- Full Graphics captioning and overlay facility
- Graphics Drivers for all popular Printers
- Joystick/Mouse Driver for Cursor Movement
- Works with any ROM versions, even JDOS
- 33k Available Worksheet Space
- Up to 256 Columns or 256 Rows
- Can use VisiCalc worksheets & training materials
- On-screen Help Messages

RAINBOW CERTIFICATION SEAL

OS-9 version available from Tandy/Radio Shack #26-3275



now with GRAPHICS!

FOR 64K DISK SYSTEMS

available from

COMPUTER SYSTEMS CENTER

NEW ADDRESS

42 Four Seasons Center #122
Chesterfield, MO 63017 USA
(314) 576-5020
10 to 6 Mon.-Fri.

CSC

or your local DYNACALC dealer

Price US$99.95
Outside North America add $10 postage

CANADA-CDN $129.95
DATAMAN INTERNATIONAL
420 FERGUSON AVE. N.
HAMILTON, ONT. L8L 4Y9
(416) 529-1319

AUSTRALIA
PARIS RADIO
161 BUNNERONG RD.
KINGSFORD 2032 NSW
(612) 344-9111

MasterCard VISA