For your TANDY Color Computer

Registered by Australian Post — Publication No QBG 4009

NZ $3.95 PNG K5.95
$4.95

AUSTRALIAN RAINBOW

November, 1986

No.65

BETTER FOR MICROS

Now there's an exciting new world for Personal Computer owners to explore. The world of Goldlink 642 on Telecom Viatel.

All you need is a 1200/75 baud modem, the appropriate software, and a telephone line, and your PC will be ready to go.

Suddenly you'll be able to shop for software on your PC, and actually download* it directly through the Viatel system. You'll be able to get PC advice and tips. Even place messages on the system for other Viatel users to read and respond to — literally a PC talkback service that lets you have a say on almost any subject.

That's just part of what Goldlink 642 offers. And Goldlink 642 is just part of what Telecom Viatel offers. You can also bank with Viatel, place bets, buy and sell shares, book travel, and much more. Instantly, easily, economically. 24 hours a day.

Ask for a free brochure at any Telecom Business Office. And start using your micro in a whole new, better way.

* Coming

Telecom Australia
Better for Business

TM657

Telecom
VIATEL®
AUSTRALIA'S NATIONAL VIDEOTEX SERVICE

APPLICATION FORM

DATE OF APPLICATION / /

(BEFORE COMPLETING THIS APPLICATION, PLEASE READ REVERSE SIDE CAREFULLY)

section 1

PLEASE TICK APPROPRIATE BOX TO INDICATE SERVICE REQUIRED

BUSINESS SERVICE ☐ NON-BUSINESS SERVICE ☐

(CHARGES INCURRED ON BUSINESS SERVICES ARE USUALLY TAX DEDUCTIBLE)

SURNAME (OR BUSINESS NAME IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS NUMBER/STREET

SUBURB/CITY STATE POSTCODE

TELEPHONE NUMBER ON WHICH SERVICE IS REQUIRED (INCLUDING STD CODE)

section 2

CONTACT NAME (IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS FOR BILLING IF DIFFERENT FROM SECTION 1 ABOVE
NUMBER/STREET

SUBURB/CITY STATE POSTCODE

CONTACT TELEPHONE NUMBER (INCLUDING STD CODE)

section 3

PLEASE DESCRIBE NATURE OF BUSINESS (OR OCCUPATION IF NOT A BUSINESS SERVICE)

PLEASE INDICATE TYPE OF EQUIPMENT USED TO ACCESS VIATEL

special instructions

THIS FORM SHOULD BE HANDED IN AT ANY TELECOM BUSINESS OFFICE OR MAY BE MAILED WITHOUT A STAMP TO FREEPOST 20, VIATEL BOX 188C, GPO MELBOURNE, VICTORIA 3001.

PLEASE ALLOW TEN WORKING DAYS FOR PROCESSING OF APPLICATION AND RETURN MAIL ADVICE.

telecom use only

DTE PP VN

BG SC CI

REF

SPAN OVERLAY NO. 140

REGISTRATION AND SUBSCRIPTIONS

Customers must register as a Business Service if the telephone number nominated for the use of the VIATEL Service is a Business Service and/or VIATEL is to be used wholly or mainly for Business, Commercial, Industrial, Professional or Government purposes. (Charges incurred on Business Services are usually tax deductible.)

Where a Business Telephone Service is nominated for the use of VIATEL, but the use of VIATEL is wholly or mainly for Non-Business purposes, the Customer may be registered as a Non-Business VIATEL subscriber, providing the registration is taken out in the Customer's personal name and address and not a Business name.

Telecom Australia will register the Business or Individual named under Section 1 as a Customer of its VIATEL Service and will provide the Customer with a confidential Customer Identity Number and Personal Password by mail.

Where billing address is indicated, bills and bill related correspondence ONLY will be forwarded to that address. All other correspondence will be forwarded to address under Section 1.

Customers should advise VIATEL of any change of address as soon as possible.

If you lose your Customer Identity Number and/or Personal Password, you must advise VIATEL in writing before new numbers are issued. Our postal address is: Freepost 20, Box 188C, GPO Melbourne, Vic. 3001. FOR SECURITY REASONS REPLACEMENT NUMBERS AND PASSWORDS CANNOT BE PROVIDED OVER THE TELEPHONE.

Customers of VIATEL acknowledge that their name and registered VIATEL Number will appear on the VIATEL Mailbox Directory and that Service Providers and/or other registered VIATEL users may send messages to their VIATEL number.

Telecom Australia undertakes no responsibility in relation to the accuracy of the information or service provided by Service Providers on VIATEL. Telecom Australia will not be responsible for any loss or damage arising out of or in any way connected with the use of this information or service.

Attention is also drawn to the terms and conditions governing the provision of information and services by some Service Providers. These terms and conditions may, in some cases, include a disclaimer absolving the Service Provider from liability regarding information and services supplied on VIATEL. The means of accessing these terms and conditions is set out on the Service Provider's Index Page on VIATEL.

Should you require any changes to your existing telephone equipment (e.g. new exchange line, additional socket), please contact your local District Telecom Office.

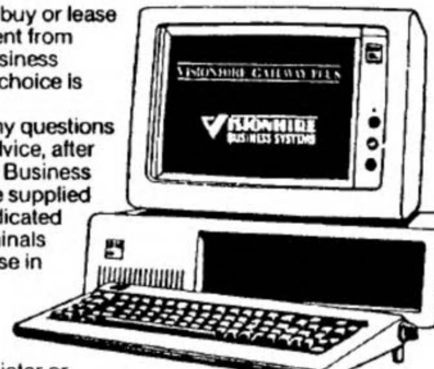
In a small number of cases VIATEL reception may be unsatisfactory. Correction may incur an additional charge.

VIDEOTEX EQUIPMENT SOLUTIONS

Anyone with a telephone can access Videotex and Visionhire Business Systems will tailor the hardware and software to suit your needs, ranging from an adaptor for only $18.00 per month up to IBM's PC of the year,the AT. Only the latest equipment from famous brands like IBM, Apple, Commodore, Tandata and Sony is used, all backed by Visioncare's Australia wide full service network.

You can rent, buy or lease your equipment from Visionhire Business Systems, the choice is yours.

If you have any questions just ask for advice, after all, Visionhire Business Systems have supplied 75% of all dedicated videotex terminals currently in use in Australia.



For friendly advice ring Will Ballschmieter or Don Sanderson on 52 5723.

VISIONHIRE BUSINESS SYSTEMS

A Division of Visionhire Australia Pty Ltd

36 Brookes Street, Bowen Hills

BAYNE & TREMBATH CHRISTMAS SPECIAL

CoCo 3 + DRIVE 0

$ 820

CALL NOW!!!

Ask for Bob Parkinson

SPECIAL CLOSES 24/12/86

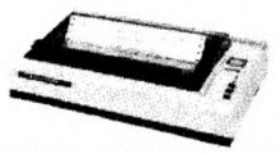
Sorry, this special price applies ONLY to sales of the CoCo3 + DRIVE 0

3 Boneo Rd., Rosebud, Victoria 3940

Ph: (059) 86 8288. A/H: (059) 85 4947

CPA - 80 (S)	$522.96
CPB - 80 (S)	$554.38

PERIPHERAL PRINTERS
IDEAL FOR CO-CO



- 100 CPS CPA - 80 (S) ● 130 CPS CPB - 80 (S)
- TRACTOR/FRICTION PAPER FEEDER STD.
- WITH SERIAL RS 232 C/CURRENT LOOP.
- DATA TRANSFER RATE - 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400 AND 4800 B.P.S. - PROGRAMMABLE.
- SYNCHRONIZATION - ASYNCHRONOUS
- HANDSHAKING - FLAG CONTROL X-ON/X-OFF PROTOCOL.
- LOGIC LEVEL - EIA LEVEL (RS 232 C) 20ma CURRENT LOOP.
- SOCKET PROVISION FOR USER FITTED 4K BUFFER (USES 6116)
- SERVICE MANUALS AND SPARE PARTS AVAILABLE.

SPECIAL OFFER — WHILE STOCKS LAST

- **DUAL DISC SYSTEM FOR CO-CO**
  DUAL SINGLE SIDED DOUBLE DENSITY DISC DRIVES.
  CONTROLLER CABLES.
  CASE POWER SUPPLY AND MANUALS DOS & ROM — $499 inc. Tax
- **SINGLE DRIVE SYSTEM FOR CO-CO**
  ONE DOUBLE SIDED DISC DRIVE (CASE HAS SPACE FOR 2 DRIVES) OTHERWISE AS ABOVE — $449 inc. Tax
- **DISKETTE SPECIAL**

DS DD GTR BLK PK25		$2.14 ea inc. Tax
CLUB OR SCHOOL BUY	200	$1.95 ea inc. Tax
	800	$1.85 ea inc. Tax

PRICES VALID WHILE STOCKS LAST — OTHERWISE SUBJECT TO EXCHANGE VARIATION.

energy CONTROL

ENERGY CONTROL PTY. LTD.
P.O. BOX 6502, GOODNA. QLD. 4300
BRISBANE, AUSTRALIA.
PHONE (07) 288 2455
TELEX AA 43778 ENECON
PHONE WELLINGTON N.Z. (64) 843 499
W.A. (09) 325 5722

Hardware/Software Specialists

For All Your CoCo Needs

AUTO ANSWER $399.00

INFO CENTRE

THE FIRST BULLETIN BOARD SYSTEM
for Tandy's computers
(02) 344 9511 — 300 BPS (24 Hours)
(02) 344 9600 — 1200/75 BPS
(After Hours Only)

SPECIAL!

Avtek Mini Modem + Cable + CoCo Tex Program — the total Viatel System — $279.00

We also have the largest range of Software for OS-9 and Flex operating systems.

PARIS RADIO ELECTRONICS

161 Bunnerong Rd., Kingsford. N.S.W. 2032

(02) 344 9111

RAINBOW Info

How To Read Rainbow

Please note that all the BASIC program listings you find in THE RAINBOW are formatted for a 32-character screen — so they show up just as they do on your CoCo screen. One easy way to check on the accuracy of your typing is to compare what character "goes under" what. If the characters match — and your line endings come out the same — you have a pretty good way of knowing that your typing is accurate.

We also have "key boxes" to show you the *minimum* system a program needs. But, *do* read the text before you start typing.

Finally, the little cassette symbol on the table of contents and at the beginning of articles indicates that the program is available through our RAINBOW ON TAPE service. An order form for this service is on the insert card bound in the magazine.

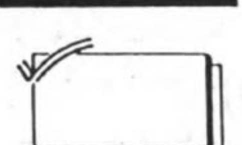
What's A CoCo

CoCo is an affectionate name that was first given to the Tandy Color Computer by its many fans, users and owners.

However, when we use the term CoCo, we refer to both the Tandy Color Computer and the TDP System-100 Computer. It is easier than using both of the "given" names throughout THE RAINBOW.

In most cases, when a specific computer is mentioned, the application is for that specific computer. However, since the TDP System-100 and Tandy Color are, for all purposes, the same computer in a different case, these terms are almost always interchangeable.

The Rainbow Check Plus



The small box you see accompanying a program listing in THE RAINBOW is a "check sum" system, which is designed to help you type in programs accurately.

Rainbow Check PLUS counts the number and values of characters you type in. You can then compare the number you get to those printed in THE RAINBOW. On longer programs, some benchmark lines are given. When you reach the end of one of those lines with your typing, simply check to see if the numbers match.

To use *Rainbow Check PLUS*, type in the program and CSAVE it for later use, then type in the command RUN and press ENTER. Once the program has run, type NEW and ENTER to remove it from the area where the program you're typing in will go

Now, while keying in a listing from THE RAINBOW, whenever you press the down-arrow key, your CoCo gives the check sum based on the length and content of the program in memory. This is to check against the numbers printed in THE RAINBOW. If your number is different, check the listing carefully to be sure you typed in the correct BASIC program code. For more details on this helpful utility, refer to H. Allen Curtis' article on Page 21 of the February 1984 RAINBOW.

Since *Rainbow Check PLUS* counts spaces and punctuation, be sure to type in the listing exactly the way it's given in the magazine.

```
10 CLS:X=256*PEEK(35)+178
20 CLEAR 25,X-1
30 X=256*PEEK (35)+178
40 FOR Z=X TO X+77
50 READ Y:W=W+Y:PRINT Z,Y;W
60 POKE Z,Y:NEXT
70 IFW=7985THEN80ELSEPRINT
   "DATA ERROR":STOP
80 EXEC X:END
90 DATA 182, 1, 106, 167, 140, 60, 134
100 DATA 126, 183, 1, 106, 190, 1, 107
110 DATA 175, 140, 50, 48, 140, 4, 191
120 DATA 1, 107, 57, 129, 10, 38, 38
130 DATA 52, 22, 79, 158, 25, 230, 129
140 DATA 39, 12, 171, 128, 171, 128
150 DATA 230, 132, 38, 250, 48, 1, 32
160 DATA 240, 183, 2, 222, 48, 140, 14
170 DATA 159, 166, 166, 132, 28, 254
180 DATA 189, 173, 198, 53, 22, 126, 0
190 DATA 0, 135, 255, 134, 40, 55
200 DATA 51, 52, 41, 0
```

Using Machine Language

Machine language programs are one of the features of THE RAINBOW. There are a number of ways to "get" these programs into memory so you can operate them.

The easiest way is by using an editor/assembler, a program you can purchase from a number of sources.

An editor/assembler allows you to enter mnemonics into your CoCo and then have the editor/assembler assemble them into specific instructions that are understood by the 6809 chip that controls your computer.

When you use an editor/assembler, all you have to do, essentially, is copy the relevant instructions from THE RAINBOW'S listing into CoCo.

Another method of getting an assembly language listing into CoCo is called "hand assembly." As the name implies, you do the assembly by hand This can *sometimes* cause problems when you have to set up an ORIGIN statement or an EQUATE. In short, you have to know something about assembly to hand-assemble some programs.

Use the following program if you wish to hand-assemble machine language listings:

```
10 CLEAR200,&H3F00:I=&H3F80
20 PRINT "ADDRESS:";HEX$(I);
30 INPUT "BYTE";B$
40 POKE I,VAL("&H"+B$)
50 I=I+1:GOTO 20
```

This program assumes you have a 16K CoCo. If you have 32K, change the &H3F00 in Line 10 to &H7F00 and change the value of I to &H7F80.

The Crew

Founder Greg Wilson
Publishers Graham & Annette Morphett
Managing Editor Graham Morphett
Accounts Annette Morphett
Assistant Editor Sonya Young
Advertising Graham Morphett
Art Jim Bentick
Sub Editors
Assembly Language: Kevin Mischewski
MC-10: Jim Rogers
softgold: Barry Cawley
Forth: John Pexon
OS-9: Jack Fricker
Special Thanks to
Brian Dougan, Paul Humphreys,
Alex Hartmann, Michael Horn,
Darcy O'Toole, Martha Gritwhistle,
Geoff Fiala, John Redmond
and Mike Turk.

Phones: **(075) 51 0577 Voice**
(075) 32 6370 CoCoLink
Deadlines:
7th of the procedding month.
Printed by:
Australian Rainbow Magazine
P.O. Box 1742
Southport. Qld. 4215.
Registered Publication QBG 4009.

AUSTRALIAN RAINBOW

CONTENTS

STOP GLARE !

Ever tried to de-bug your screen of Glare and Reflection ! Can't, can you? TELEMAT Anti-Glare sprays for Computer and T.V. screens can. Developed and manufactured in Australia to combat this very problem. It will reduce Reflection and Glare by up to 94% and will also solve the problems of eye strain and headaches. TELEMAT is available in VDU monitor Amber, Green, Neutral and TV Neutral. It is applied in a spray form and application is carried out by trained technicians. The coating has an indefinite life span and is guaranteed for 3 years. The coating is very durable and will not show fingerprints. Average coating time is approximately 10 to 15 minutes per unit. It can change a normal black and white VDU into an Amber or Green unit. To maintain the coating, we use as part of the process, and recommend, TELEMAT ANTI-STATIC POLISH. The coating has no detrimental effect on the screen or unit and can be removed very quickly if required. This form of Anti-Glare treatment is being used throughout Australia by large corporations right through to the home user. The product has been independently tested by the Queensland Institute of Technology and these results are available on request.

For more detailed information write,
TELEMAT Anti-Glare Pty. Ltd.,
3 Fortril Drive, Springwood,
Qld. 4127. or Phone: **(07) 208 3557**

GOLDLINK COM. STATION

642 on VIATEL

DISTRIBUTORS

Distributors are required in all states throughout Australia for a recently developed product designed to combat the problems of reflection and glare on Computer and Television screens. This form of anti-glare is a proven product and is manufactured in Australia. The low initial purchase price for the distributorship and the extremely attractive returns make this a very viable proposition to an individual or company. There is unlimited sales potential in the Australian market. Initial assistance in setting-up is provided and on-going aid etc assured.

For further details write to **TELEMAT Anti-Glare Pty. Ltd.**
3 Fortril Drive, Springwood, Qld. 4127
Phone (07) 208 3557 or N.S.W. (065) 54 1441

PRINT#-2

FINALLY I get the new CoCo to work with! The Colour Computer 3 arrives this afternoon (I hope)! After weeks of waiting and screaming we finally get the words, "Your new CoCo will arrive sometime today". This will replace the CoCo I work with now. It is a long white case (the one directly after the grey case before the short white case) and has served me well. I'm going to miss that keyboard, though. It's the one after the grey button-type one before the raised ones with the alphabet placed in the top left.

What do I do with this new machine? Well the white case I've been working with does this primary job: I receive your program(s) and process them; ie I load them into the computer. If they don't load I send them back. If they do load, I see if they work. If they do I then LLIST them on a printer and write up the instructions for them. They are then sorted into their proper categories and at the end of the month, are put onto Australian Rainbow On Tape or CoCoOz.

But sometimes this process doesn't always work. There are cases of incompatability which are presently on the increase.

And there will be more incompatability in times to come. Just last week, 500 new CoCo's came into Tandy's warehouse and out again like a hot knife through butter.

But what really impresses me is the addition of the different screen widths, more colours available for graphics, characters and backgrounds and not to mention underlining and blinking. Much better (eh? it already has been the best for the past 4-5 years!) ... alright, superior screen presentation. And all accessable using BASIC!! I mean, we've had the best computer ever built; now we've got one that is even better than the best!

That's why I've been saying for the past few weeks I must have the new CoCo 3!

Anyway, to those 500 people with the new Colour Computer: Congratulations! You have made an excellent buy, and for a measly $400!! You wouldn't find that price tag on an Amiga, Atari or IBM PC!

Which leads me onto another point. There is very little software at present! So how about you 500 programmers get something in? We would LOVE to see new material come in!

In fact we herewith announce a short sharp competition for programs - CoCo 3 BASIC. The best program of any sort submitted strictly by 7th February 1987 will win one pair joysticks, 1 box Tandy disks, 1 Koala Pad and 2 months of Rainbow On Tape or Disk.

I'm all ready and waiting for your program(s). It doesn't matter what it is or how long it is as long as it shows some of the new CoCo's abilities.

We will be bring you more inside information on the new CoCo 3 the moment it arrives! Come to think of it, we've been swamped by questions about the CoCo 3. One of the most popular questions was (in not those exact words):

"I've just read the new Tandy catalogue. It says in there that you have to have a monitor to access the hi-resolution screens with the new CoCo 3. Is that true?"

Well the answer is this: You don't need a monitor to access the hi-res screens. You can still use your TV set, although it's advisable getting a monitor, or even better, an RGB monitor. If you have a TV set, the border tends to move to the left by two characters. Also, the graphics are just a little fuzzy in the hi-res screen and having a TV set is quite acceptable, as long as it's of good quality.

If you have any questions, just give me a ring and ask for Alex or contact your local user group meet.

Viatel (or Com Station 642 as its known)

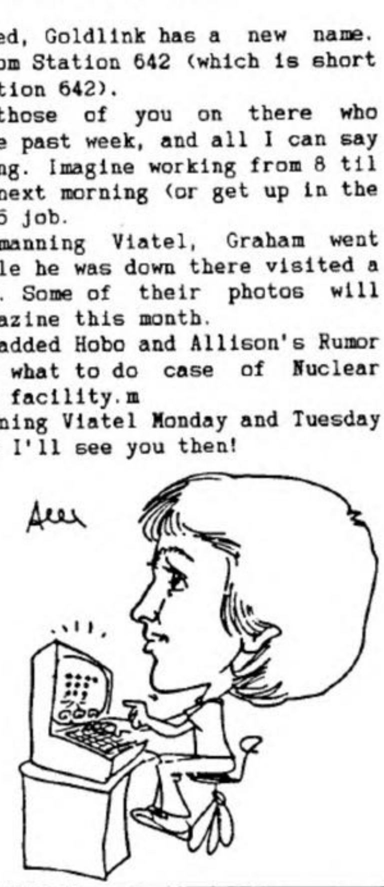
If you haven't noticed, Goldlink has a new name. It is now known as Com Station 642 (which is short for Communications Station 642).

I've manned it (for those of you on there who hadn't noticed) for the past week, and all I can say is that it's very tiring. Imagine working from 8 til 12 at night to get up next morning (or get up in the morning) and do a 9 - 5 job.

Anyway while I was manning Viatel, Graham went down to Sydney and while he was down there visited a few of the Viatellians. Some of their photos will appear in the CoCo Magazine this month.

This month we have added Hobo and Allison's Rumor and Winge Board. Also, what to do case of Nuclear attack and a letterbox facility.m

And I will be manning Viatel Monday and Tuesday nights from now on. So I'll see you then!



MK1 SERIAL/PARALLEL PRINTER INTERFACE

- Connect COCO 1 ,2 or 3 to any PARALLEL PRINTER eg: EPSON,GEMINI, BMC,CP 80 ,TANDY
- EXTRA SERIAL PORT for MODEM no need to UNPLUG CABLES.
- INCLUDES all CABLES and Connectors
- SIX BAUD RATES: 300,600, 1200, 2400,4800,9600.
- FASTER PRINTING for SCREEN DUMPS
- Power Pack required for Printers not supplying power at pin 18 on on Printer connector eg:Epson,BMC

ONLY : $ 110.00 (includes postage)

add $9.00 for Power Pack if required.

VDI+ VIDEO DRIVER

- Connect to a Color or Monochrome Composite Video Monitor
- Solderless Installation.
- Includes Sound Output.

only: $35.00

Available from : G.&G. FIALA P.O. BOX 46. THORNLEIGH.2120 PHONE:02-843172

REVIEWS OF PRODUCTS JUST RELEASED IN THE USA

Software Review

Unkill: Help for Lost Programs

Sooner or later it will happen. You get that long-awaited disk drive and are thrilled with the power and speed it offers. You'll get so carried away that sooner or later you'll accidentally kill a file you didn't mean to. Some of the more expensive utilities in use today have an unkill feature coupled with a lot of other commands. Now a dedicated program called simply *Unkill* can be used if you make that dreaded mistake.

In order to put this program through its paces, I killed a program on one of my disks and then ran *Unkill*. Prompts guide the user through a series of steps to recover the lost program as long as it has not been overwritten in the meantime. When a file is killed, all that really happens is that the first character of the filename is deleted from the directory on Track 17. The rest of the file is still on the disk, but it's no longer retrievable.

This program requires two disk drives and a printer. The printer is used to read the remaining portion of the filename, as well as the file format, program length in bytes and starting granule. The program examines the entire disk for other killed files. All 67 granules are examined and information displayed as to their use. Using the information displayed on the screen and printer you can make some decisions, with the aid of the program prompts, to recover the lost program.

It is not foolproof, however. The user needs to have some working knowledge of the disk file allocation table and how files are written to the disk. Remember, when a file is killed, the file allocation table is reset.

I was impressed with *Unkill*'s ease of use. It's easy on the wallet, too. It may be worth its weight in gold if it saves even one favorite program.

(Proper Programs, P.O. Box 681, Garner, NC 27529, $9.95 plus $2 S/H)

— Jerry Semones

Software Review

C Compiler: All the Features at Half the Price

The *C Compiler* program may be used to compile C language source code programs into executable machine language OS-9 modules. The *C Compiler* reads C language source files from one or more disk files, compiles them into assembly language source files, optimizes the assembly language source code for speed and compactness, assembles the output into relocatable object code modules, and links the object modules to the library functions and other compiled object code modules to produce executable machine language OS-9 modules.

The Radio Shack *C Compiler* comes on two 35-track OS-9 disks. A light blue, spiral-bound manual of more than 100 pages accompanies the disks.

One disk contains the programs needed to compile a C source program to an executable assembly language program. The second disk contains the compiled library functions, definition files, three C source program examples and some assembly source programs.

The first disk contains files for the following: two-pass compiler executive program, macro pre-processor program, compiler pass 1 program, compiler pass 2 program, relocating assembler program, assembly code output optimizer program and assembly code output linker program.

The second disk contains directories for: source header files containing definitions for various applications, the compiled C library functions, three sample C source code programs and a subdirectory of various assembly source listings.

System calls are provided which perform functions such as open or close a file, change execution or data directory, create a new file and so on. These are provided to extend the portability of the language and to save the user from writing the functions. They are needed because these functions deal with the hardware of the system. With these calls, you won't have to know assembly language code to write programs to perform these functions. In fact, there's a system call (_os9()), that lets a C programmer access any OS-9 system call by passing the function code and a pointer to a register structure as an argument.

The pre-processor directive, #ASM, is supported for applications where you must have the speed of embedded assembly language. I found no stacking order of function arguments, so we'll have to experiment if we need this capability.

The optimizer can be suppressed via an option at compile time. This speeds up the compilation. The optimizer shortens the code about 11 percent with a comparable increase in speed, according to Microware. I compared the output of the *line.c* program with and without the optimizer and found that it replaced long branches with short ones where possible (saving one byte each time) and rearranged some code to tighten up things (saving a few more bytes). Of course, a good assembly language programmer could have done the same or better, but for long programs or inexperienced assembly language programmers it's a real boon!

A profiler program is included that keeps track of how many times each function is executed while the program is running. If your program appears to be slow, the profiler can help you find the most-used functions that might require extra effort to speed up the execution time. If it's a memory hog, then the profiler could be used to find functions that are not used and can be omitted.

Something else I really like is the option to output C source code on the assembly output listing. This assists in debugging and/or massaging areas of code that need special assembly language attention to speed up the execution. It helps locate bugs or idiosyncrasies of the compiler, too.

The company that wrote the marvelous, modular OS-9 operating system hard coded the drive number for the library drive in two of the passes! D1 is coded into CC1 at offset $EE5 and in C.PREP at offset $135C (Microware, how could you?). If you have a RAM disk or a hard disk and want to speed up the compiler, you'll need to patch the descriptor name into these locations.

A good source-level debugger would have been great; maybe we'll get one when enough CoCos have 512K of RAM! We need powerful tools so we can produce good software in less time.

A C source code library for the system calls and library functions would be helpful to beginners and software developers.

This is Version 1.00.00; the last revision was done in 1983. Either they did it right the first time, or nobody's spoken loudly enough to make them update. I do suspect though, that this version was adapted from a version supporting one of the earlier OS-9 systems in existence before CoCo OS-9 was born. How about a new version that doesn't hard code the library drive? Throw in bit fields, too, while you're at it.

The *C Compiler* is a good value at $99.95. I recommend it for anyone wanting to learn the C language, and for those who know C and don't want to program OS-9 application programs in assembly language. The features included are comparable to C compilers costing twice as much on other computers.

(Tandy Corp; available in Radio Shack stores nationwide, Cat. No. 26-3038, $99.95)

— Jesse W. Jackson

Software Review

Rapid Action, Good Graphics Highlight *Pump Man*

Man the joysticks! Aliens are attacking underground and it is your job, as Pump Man, to stop them.

PumpMan is Saguaro Software's version of the arcade game *Dig Dug*. In it, you must dig tunnels under the ground in an attempt to trap and blow up aliens using a high pressure air pump. Or, lead the aliens up a tunnel and drop a rock on them. But beware, these aliens can transport themselves through solid rock, and pop into one of your tunnels. Watch your step near the fire-breathing dragon!

Quite honestly, I'll confess to being an arcade *Dig Dug* addict. It offers a wonderful diversion from shoot-'em-ups and provides a delightful challenge. This is why, when I received *PumpMan* to review, I immediately pounced on my joystick and got ready for hours of fun.

Is *PumpMan* fun? It sure is. But, I must admit to having mixed feelings about the game. While the game was certainly welcome and well-used around my house, I couldn't help but feel that it could have been better.

PumpMan comes on a copy-protected disk, or is available on cassette. While I generally dislike copy-protection schemes, I can't fuss too much about the one used for disk, since it does let you make a non-executable backup of the program. If something happens to the original, simply back up the copy (which won't work by itself) onto the original disk and it should work. Loading the game is as easy as typing `LOADM "PUMPMAN"`.

The game is played with one joystick to move Pump Man in one of four directions. The firebutton activates the pump. I sometimes had trouble turning right or left with the older joysticks. It was a frustration to me that the firebutton did not autofire. Holding down the button does not keep the pump activated; you must constantly press it, and even then it sometimes doesn't register.

The graphics are good, and the action rapid. The animation, however, is often flickery which, while not affecting the game play, can be mildly annoying. On the plus side, *PumpMan* keeps your interest, has a game pause feature and comes with 15 board variations. The game's sound effects are adequate, but I do miss the catchy background music found in the arcade version. Most all the other features of the arcade game are included, though.

The two-page documentation covers all the necessary features of *PumpMan*.

On a rating scale of one to five, I'd rate *PumpMan* as follows: playability, three; keeps interest, four; documentation, three; graphics, three; sound, two and price vs. value, three.

PumpMan is a fun game, and it is evident that Dave Dies, the author, has a great deal of talent. As it is, it's very good, but a few more weeks of work could have made this game outstanding.

(Saguaro Software, P.O. Box 1864, Telluride, CO 81435, $24.95 tape, $29.95 disk; $1 S/H. Requires 32K and joystick)

— Eric W. Tilenius

Software Review

Mission: F-16 Assault: A Must for Serious Game Players

You're flying over enemy terrain at about Mach 2. Suddenly you see the first target, a group of unprotected buildings. You speed up as you drop bombs to destroy this target. But before the bombs hit the ground, a blip appears on the radar. You know what that means, so you get ready for combat. An enemy helicopter glides onto the screen and you fire furiously at the craft for a few seconds before destroying it. You wipe your brow and continue with the mission.

No, you haven't joined the Air Force, you're playing *Mission: F-16 Assault*, one of Diecom's newest 64K arrivals. Of all of the games I've played on my CoCo, this new release from Diecom has to be one of the best. Its graphics and animation are the best I have seen in an action game, and it has down-to-earth logic.

The scenario is as follows: You are in control of a white jet fighter. The ground scrolls beneath the plane. You have full control of the jet's movement around the screen. Your missiles fire forward and the bombs are dropped to the ground.

The entire playing field is situated over enemy territory which is dotted with different types of defenses. There are several tactical areas, including refineries, airports and other vital locations. Points are awarded for destroying these targets. Not only are these land sites targets, but some also serve the enemy. Airports, for example are used to launch airplanes, and once destroyed, aircraft may no longer take off from that site. There are other sites which are unique in this way, such as missile silos and helicopter pads.

The enemy uses several kinds of defense against your aircraft. These include missiles, jet aircraft, helicopters, boats and tanks. They may all be destroyed except for the surface-to-air missiles and the enemy's missiles which fire from the jet aircraft and helicopters.

All flying aircraft take off from a ground site such as an airport or helicopter pad. Therefore, these craft may be destroyed both on the ground and in the air. To destroy anything on the ground, you must drop a bomb. To destroy an enemy in the air, you must use missiles which fire forward from your aircraft.

Radar is on the right side of the screen and indicates most enemy craft positions. There is one special enemy aircraft which can be used to jam your radar. This aircraft must be destroyed before your radar is destroyed.

Some other features in this game are: a bonus aircraft for every 10,000 points, a pause and restart feature, and a high score board.

I love this game, and give it a full five-star rating. *Mission: F-16 Assault* is a must for all serious CoCo game players.

(Diecom Products, 6715 Fifth Line, Milton, Ontario, Canada L9T 2X8. Tape or disk $28.95 U.S., $38.95 Can.)

— Pat Downard

Software Review

Wall Street Keeps the Interest Flowing

Wall Street is a game which can be played by one to eight people, and the strategy involved does not change with the number of participants. Each player begins the game with $1,000 in cash, and tries, through the purchase and sale of company shares, to increase his or her holdings to a winning amount which can be set at any number between $2,000 and $999,999,999. *Wall Street* can even be played noncompetitively, i.e., by setting the winning amount of money at a sufficiently high level, the players can enjoy refining their tactics and mastering the idiosyncrasies of the program for several hours without even coming close to a victory.

There are eight American companies to choose from whose high, average and low stock prices are correlated with a stock indicator somewhat analogous to the famous Dow Jones Average. The object is to maximize profits by buying low and selling high as in real stock exchanges. At first I thought this was a Simulation of the stock market and anticipated some realistic market action. However, this program is a game, and its departures from realism make the proceedings swifter, more exciting, and for those who don't get too greedy, more profitable. There is an old saying in the stock market, "The Bulls make money, even the Bears make money, but the pigs . . . they don't make any money!"

For example, there was never a stockholders' meeting like the ones in this game where you go in with X shares of a company's stock and emerge with 2X, 3X or 4X and usually more shares. This is an exhilarating way to live the good life if you can resist the urge to hold the stock in the hope that it will double and triple some more, and instead, convert your shares to cash before the broker's fee is assessed at $10 per share!

We never did discover the relationship between the simulated rolls of the dice and the ups and downs of the game. We guessed that perhaps the makers of the program first created it on a physical board with squares to determine one's fate for each turn.

As we continued to play we became increasingly convinced that *Wall Street* is to the stock market what Monopoly is to real estate. *Wall Street* is written in BASIC and comes on an unprotected tape which is easily loaded and converted to disk. The documentation is adequate, although it's hard to understand the game until you play it. The game would be improved by writing the current player's name on every screen; we occasionally had some controversy over whose turn it was. Also, some folks wished the program would allow them to liquidate their shares while automatically computing scores before reaching the designated winning amount.

We are happy to recommend this program. Unlike most games, it has held our interest through several playings, and at a cost of $6 per tape, you'll still have money left over for investing in the real thing.

(Drayon Software, P.O. Box 2516, Renton, WA 98056. Requires 16K ECB, $6.)

— Patricia Arrington

Software Review

Max Fonts: Valuable Add-On

Derringer Software has produced a useful add-on for *CoCo Max* called *Max Fonts*. The existing *CoCo Max* and *CoCo Max II* provide 14 fonts. With *Max Fonts* you can add up to 72 more. There are three disks, each with 24 fonts. You can buy one, two or all three.

Each of the fonts can be modified using the Style pull-down menu. For those who are not familiar with *CoCo Max*, the Style menu allows you to alter the default printing of the fonts by making them bold, outlined, shadowed or italicized. You can also use any of the styles in combination with each other.

Max Fonts comes with five half-pages of documentation printed in very small type. Two pages are all it takes to explain how to use the product. The remaining three pages are used for showing samples of all the fonts. The instructions are clear and easy to follow.

The only drawback I could find is the disk handling required to use *Max Fonts*. After starting *CoCo Max* you must remove the system disk and replace it with the font disk you want to use. The Load Page command on the file menu is used to load the fonts. Due to the limitation of the size of the pull-down menu, only half (12) of the fonts are accessible at a time.

Should you want to use the original fonts provided by *CoCo Max*, Derringer Software has provided them on each of the three *Max Fonts* disks. This prevents having to remove a *Max Fonts* disk and replace the system disk.

Max Fonts is compatible with the original *CoCo Max* as well as the new *CoCo Max II*. I tested the fonts on both systems and had absolutely no problems. *Max Fonts* is easy to use, well-documented and performs the job as designed and advertised. I recommend this software as a valuable add-on for the *CoCo Max* system user.

(Derringer Software Inc., P.O. Box 5300, Florence, SC 29502-5300, $24.95 each, all three for $64.95)

— Rick L. Earsley

Book Review

Disk BASIC Unraveled Is a Valuable Library Addition

Here's a book you software hackers out there can really sink your teeth into. *Disk BASIC Unraveled* is a fully detailed and documented disassembled listing of Disk BASIC Versions 1.0 and 1.1. The book is not a tutorial or a how-to manual on machine language, but rather a detailed look at the assembly listings.

The reader needs to have beginning knowledge of 6809 assembly language programming to be able to take full advantage of the opportunities this book offers. It is also assumed that the reader is familiar with the contents of the disk system owner's manual which contains a general description of the overall operation of Disk BASIC and other useful information concerning the physical and logical format of the tracks and sectors.

Properly studied and used, *Disk BASIC Unraveled* should help the serious reader understand the theory behind Color DOS, and to modify it for his own purposes or add extra commands or functions.

The book is nicely bound, magazine size and 154 pages long. It's full of useful information for the serious CoCo hacker. There is even a nice section that deals with the 1793 Floppy Disk Controller, which I found to be very useful. If you are inclined toward machine language programming, you will benefit from this publication.

(Spectrum Projects Inc., P.O. Box 21272, Woodhaven, NY 11421, $19.95 plus $3 S/H)

— Jerry Semones

Hardware Review

Colorchestra Lets the Music Flow

Since I am in a country rock band, write my own music, and own a computer and a synthesizer, I was thrilled when I received Colorchestra, a MIDI sequencer for the Color Computer. There are many MIDI sequencers for other computers but not for the Color Computer. Now those of us who own CoCos and synthesizers won't be tempted to buy another computer to use with our musical instruments. Before I review Colorchestra though, I had better give a little history and explain what MIDI equipment is supposed to do.

MIDI (Musical Instrument Digital Interface) is a universal language adopted by most musical instrument manufacturers so that MIDI-equipped instruments can communicate and control each other, regardless of which company makes the instrument.

When electronic music was first being developed it was possible to interconnect or interface to monophonic synthesizers by using simple voltage signals. But as computerized polyphonic synthesizers became the norm, the old techniques of interfacing became too cumbersome. So MIDIs were developed. They use eight-bit signals to serially transmit all kinds of information from one instrument to another.

Colorchestra is a very attractive product. There is even a ROM pack made of walnut! The documentation is packaged in a sturdy binder and there is a cassette version as well as two disk versions. The software is not copy protected, so make and use copies. You must have the hardware and the software to make it work.

The many features include: works with any MIDI-equipped synthesizer or rhythm machine; 16 polyphonic multifunction tracks; 8,000 note storage not dedicated to any specific track; user friendly Hi-Res graphics interface; real time write mode; solo capability on any track; varying tempo range from 30 to 250 beats per minute; audible and visual metronome; programmable measure locator; sequencer records from any MIDI channel (1-16); each track can output to any MIDI channel (1-16); records full spectrum of MIDI data including program changes, pitch bends and all 128 available MIDI controllers; accepts or transmits MIDI synchronization for rhythm or drum machines; and programmable time signature plus many others.

To use Colorchestra you need a 64K Color Computer and any MIDI-equipped device. With disk drive you also need a Y Cable or a multipack. It does not matter what version of DOS you have because the program makes no ROM calls.

Colorchestra is the beginning of a series of programs from Horizon that allows you to work with the CoCo and a MIDI.

It is not a problem to boot up Colorchestra in either cassette or disk format. In the main menu you will see a well-designed screen of icons giving eight choices. They are Multitrack Recording, Track Editing, Toolbox Menu, Control Panel Menu, Help, Disk I/O, Cassette I/O and System Trash. To choose any of the options is simply a matter of using the arrows to move and pressing ENTER.

Multitrack recording is the heart of the program. Here you can record in any or all of the 16 tracks after selecting recording or playback options. Your options include Recording Resolution (used to clean up timing errors), Time Signature, Tempo, Metronome Mode (to control timing and sychronization with other machines) and Track Selection. When ENTER is pressed, recording begins after one measure. To end recording, press BREAK, which returns you to the main menu.

Track editing and the Tool kit give you the ability to change what you have recorded. You can change individual notes or completely transpose your composition from one key to another.

The control panel is used to change Colorchestra's general options: MIDI echo controls, velocity controls, sequence title, real time filter settings, MIDI in channel selection and MIDI out channel controls .

The options for Help, Disk I/O, Cassette I/O and System Trash are all self-explanatory and easy to use. Each option uses icons and the arrow keys for selection. In the Disk I/O menu you can Load, Save, Kill, Rename and get a directory. In the Cassette I/O you can Save, Load and Verify.

If you have MIDI equipment and a CoCo, now there is a product to let the creative juices flow. Colorchestra is the beginning of a new world for CoCo musicians.

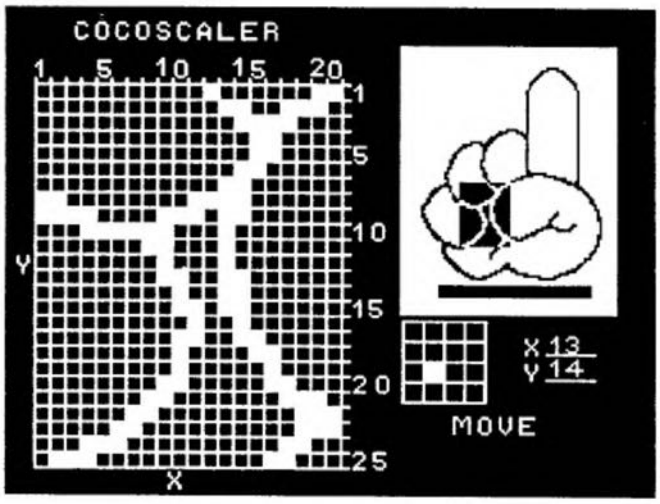
(Horizon Software Corporation, P.O. Box 289, Opelousas, LA 70570, 64K required, $149.95)

— Thomas E. Nedreberg

32K
Disk

An easy-to-use detailer for CoCo drawings

The CoCo Scaler



By Wayne Womack

The CoCo Scaler can be used to get more detail in your drawings. While you draw on a low resolution screen, your drawing is displayed in high resolution in the upper-right corner of the screen. The commands available in *CoCo Scaler* are Move, End, Set and Reset.

Each section of the high resolution screen can be worked on, one section at a time, in the low resolution section. First, load an old drawing or create a new one. If you want an old drawing, it will be loaded and will show up in the high resolution grid. Press 'M' to move a section of the picture to the editing grid. Use the arrows to move the block marker to the correct section. Then press 'S' or 'R' and that section is moved to the editing grid.

Once the section is in place, use the joystick to move to the correct square. Pressing the firebutton fills the square with white (if you press 'R') or black (if you press 'S'). You can toggle between 'R' and 'S' as needed. As you edit in the low resolution grid, the picture in the high resolution grid is also changed. When finished with one section, press 'M' to get another section. When you are finished changing the picture, press 'E' and give your drawing a name. It will be saved with the extension /DRW.

(You may direct questions about this program to the author at 12738 Gist Road, Bridgeton, MO 63044. Please enclose an SASE for a reply.) □

Wayne Womack has been a commercial artist for 15 years and lives in Bridgeton, Missouri. In the evenings he teaches BASIC programming at a local high school.

Line	Function
10-30	Remark statements.
40-50	Initialize program.
60-320	Create each letter of the alphabet.
233-610	Create the numbers.
620-630	More program initialization.
640-1130	Draw the Title Screens.
1140	Timing Loop.
1150	Clears the screen.
1160-1200	Let you choose an old drawing or start a new drawing.
1210-1220	Draw frame for the high resolution screen in the upper right corner of the screen.
1230-1250	More program initialization.
1260	Writes *CoCo Scaler* at top of screen.
1270-1330	Draw the grid used for the low resolution screen.
1340-1490	Draw numbers and letters on the screen.
1500-1520	Draw the move screen grid.
1530-1710	Joystick and Command processing lines.
1720-1780	Draw X and Y position on the screen.
1790-1920	Show you where you are on the high resolution screen.
1930-2100	Scan the high resolution area and transfer it to the low resolution side for editing.
2110	Clears everything off screen except the drawing area for saving the picture to disk.
2110	Timing loop.
2130-2200	Save the drawing to disk.
2210-2280	Load a drawing.
2290-2310	This is a temporary block to show you where you are while in moving mode.

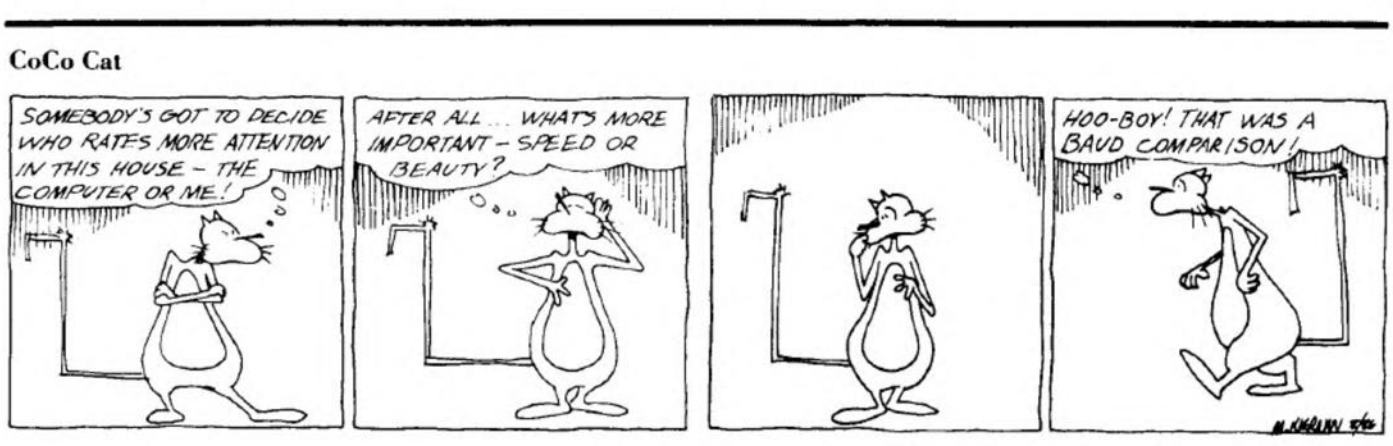
180	11	1620	133
360	126	1700	22
580	82	1860	136
840	232	2000	73
1030	196	2170	173
1240	174	END	22
1480	176		

The listing: SCALER

```
10 'THE COCO SCALER
20 'BY VV SOFTWARE
30 '10/21/84
40 CLEAR1500:PCLS
50 DIM NO$(25),A(25),SC(25)
60 '***LETTERS***
70 LA$="BM+1,0U4E2F2D2BL4R4D2BR3
"
80 LB$="BM+1,0U6R3F1D1G1BL3R3F1D
1G1BL3R3BR4"
90 LC$="BM+1,0BR3E1BU4H1L2G1D4F1
R2BR4"
100 LD$="BM+1,0BR3E1U4H1L3D6R3BR
4"
110 LE$="BM+1,0BR5BU6L5D3R3BL3D3
R5BR3"
120 LF$="BM+1,0BR5BU6L5D3R3BL3D3
BR6"
130 LG$="BM+1,0BR5E1U2L2BR2BU2H1
L2G1D4F1R2BR4"
140 LH$="BM+1,0U6BD3R4BU3D6BR4"
150 LI$="BM+1,0BU6R4BL2D6BL2R4BR
3"
160 LJ$="BM+1,0BU1F1R2E1U5BD6BR4
"
170 LK$="BM+1,0U6BD3R2E2U1BL2BD3
F2D1BR3"
180 LL$="BM+1,0U6BD6R5BR3"
190 LM$="BM+1,0U6R1F2D1U1E2R1D6B
R3"
200 LN$="BM+1,0U6F5BU5D6BR3"
210 LO$="BM+1,0BR3L2H1U4E1R2F1D4
G1BR4"
220 LP$="BM+1,0U6R3F1D1G1L3D3BR7
"
230 LQ$="BM+1,0BR3L2H1U4E1R2F1D4
G1BH1F2BU1BR3"
240 LR$="BM+1,0U6R3F1D1G1BL3R3F1
D2BR3"
250 LS$="BM+1,0BU1F1R2E1U1H1L2H1
U1E1R2F1BD5BR3"
260 LT$="BM+1,0BR6BU6L6BR3D6BR5"
270 LU$="BM+1,0BU6D5F1R3E1U5BD6B
R3"
280 LV$="BM+1,0BU6D4F2E2U4BD6BR3
"
290 LW$="BM+1,0BU6D6R1E2U2D2F2R1
BU6D6BR3"
300 LX$="BM+1,0U1E4U1BL4D1F4D1BR
3"
310 LY$="BM+1,0BU6D2F2E2U2BL2BD4
D2BR5"
320 LZ$="BM+1,0BU6R4D1G4D1R4BR3"
330 '***NUMBERS***
340 NO$(0)="BM+1,0BR1R2E1U4H1L2G
1D4F1BH1E4BD5BR3"
350 NO$(1)="BM+1,0BU4E2D6BR3"
360 NO$(2)="BM+1,0BU5E1R2F1D1G4R
4BR3"
370 NO$(3)="BM+1,0BU5E1R2F1D1G1L
1BR1F1D1G1L2H1BF1BR6"
380 NO$(4)="BM+1,0BU2E4D6BL4BU2R
6BD2BR3"
390 NO$(5)="BM+1,0BR4BU6L4D3R3F1
D1G1BL3BU1F1R2BR3"
400 NO$(6)="BM+1,0BR5BU5H1L2G1D4
F1R2E1U1H1L3BD3BR6"
410 NO$(7)="BM+1,0BR3U2E3U1L5BD6
BR8"
420 NO$(8)="BM+1,0BR3L2H1U1E1H1U
1E1R2F1D1G1L2BR2F1D1G1BR4"
430 NO$(9)="BM+1,0BU1F1R2E1U4H1L
2G1D1F1R3BD3BR4"
440 NO$(0)="BM+6,-1U4H1L2G1D4F1R
2E1G1BR4"
450 NO$(10)=NO$(1)+NO$(0)
460 NO$(11)=NO$(1)+NO$(1)
470 NO$(12)=NO$(1)+NO$(2)
480 NO$(13)=NO$(1)+NO$(3)
490 NO$(14)=NO$(1)+NO$(4)
500 NO$(15)=NO$(1)+NO$(5)
510 NO$(16)=NO$(1)+NO$(6)
520 NO$(17)=NO$(1)+NO$(7)
530 NO$(18)=NO$(1)+NO$(8)
540 NO$(19)=NO$(1)+NO$(9)
550 NO$(20)=NO$(2)+NO$(0)
560 NO$(21)=NO$(2)+NO$(1)
570 NO$(22)=NO$(2)+NO$(2)
580 NO$(23)=NO$(2)+NO$(3)
590 NO$(24)=NO$(2)+NO$(4)
600 NO$(25)=NO$(2)+NO$(5)
610 MI$="BM+2,-3R3BD3BR2"
620 A=5:B=20
630 PMODE 4,1: PCLS: SCREEN 1,1
640 DRAW "BM81,177;XLV$;": DRAW
"BM80,177;XLV$;": DRAW "BM90,177
;XLW$;": DRAW "BM91,177;XLW$;"
650 DRAW "BM105,177;XLS$;XLO$;XL
F$;XLT$;XLW$;XLA$;XLR$;XLE$;"
560 PMODE 4,1
670 LINE(4,4)-(250,166),PSET,B
680 LINE(4,182)-(250,182),PSET
690 LINE(4,185)-(250,185),PSET
700 XX=254:YY=166
710 NT$="L255O4V3T255;4;4;"
720 FORX=4TO250 STEP5
730 XX=XX-5
740 LINE(X,4)-(XX,YY),PSET
750 PLAY NT$
760 NEXT
770 FORY=8TO166 STEP5
780 YY=YY-5
790 LINE(4,YY)-(250,Y),PSET
800 PLAY NT$
810 NEXT
820 FORI=1TO1000:NEXTI
830 LINE(4,4)-(250,165),PRESET,B
F
840 LINE(6,167)-(250,181),PRESET
,BF
850 NT$="L200O2;1;2;3;4;"
860 VN$="V1T15"
870 DRAW "BM95,177;XLP$;XLR$;XLE
$;XLS$;XLE$;XLN$;XLT$;XLS$;"
880 X=4
890 PLAY VN$
900 FORY=5TO52 STEP4: X=X+6
910 LINE(X,Y)-(X*2.75,Y*2.75),PS
ET,B
920 FORI=1TO2
930 PLAY "V+T+"+NT$
940 NEXTI
950 LINE(X+1,Y*2.75)-(X*2.75,Y*2
.75),PRESET
960 LINE(X*2.75,Y+1)-(X*2.75,Y*2
.75),PRESET
970 NEXTY
980 LINE(82,52)-(82*2.75,52*2.75
),PSET,B
990 FORI=1TO2:PLAY "V+T+"+NT$:NE
XTI
1000 LINE(6,167)-(250,181),PRESE
T,BF
1010 DRAW"BM95,95;S16;XLC$;XLO$;
XLC$;XLO$;":DRAW"BM96,94;XLC$;XL
O$;XLC$;XLO$;S12;"
1020 DRAW"BM102,119;XLS$;"
1030 DRAW"BM119,119;XLC$;"
1040 DRAW"BM135,119;XLA$;"
1050 DRAW"BM152,119;XLL$;"
1060 DRAW"BM171,119;XLE$;"
1070 DRAW"BM190,119;XLR$;"
1080 DRAW"BM+0,0;S4;"
1090 LINE(4,163)-(250,163),PSET
1100 DRAW "BM50,177;XLC$;XLO$;XL
P$;XLY$;XLR$;XLI$;XLG$;XLH$;XLT$
;"
1110 DRAW "BM135,177;XNO$(1);XNO
$(0);XMI$;XNO$(2);XNO$(1);XMI$;X
NO$(8);XNO$(4);"
1120 PLAY "T4O4"
1130 PLAY "V2L8CFAO+CP8O-AL4.O+C
P4P8O-"
1140 FORI=1TO1000:NEXTI
1150 PCLS
1160 CLS:PRINT:PRINT:PRINT:PRINT
"        NEW OR OLD SCREEN":PRINT
:PRINT"           PICK N OR O"
1170 X$=INKEY$
1180 IF X$="O"THEN2210
1190 IF X$="N"THEN1220
1200 GOTO1170
1210 '***SMALL DRAWING BOARD***
1220 LINE(155,15)-(239,119),PSET
,BF
1230 '***INIT PROGRAM***
1240 PMODE 4,1:SCREEN1,1:RS=0:X1
=1:Y1=1:X2=15:Y2=33:X3=160:Y3=12
6:X4=157:Y4=17
1250 DIM M(21,26)
1260 DRAW "BM27,12;XLC$;XLO$;XLC
$;XLO$;XBK$;XLS$;XLC$;XLA$;XLL$;
XLE$;XLR$;"
1270 '***LARGE GRID***
1280 FORX=12 TO 132 STEP6
1290 LINE(X,28)-(X,180),PSET
1300 NEXT X
1310 FORY=30 TO 180 STEP6
1320 LINE(12,Y)-(135,Y),PSET
1330 NEXT Y
1340 '***NUMER LARGE GRID***
1350 DRAW "BM 11,27;XNO$(1);"
1360 DRAW "BM 36,27;XNO$(5);"
1370 DRAW "BM 59,26;XNO$(1);":DR
AW "BM 65,27;XNO$(0);"
1380 DRAW "BM 89,26;XNO$(1);":DR
AW "BM 96,27;XNO$(5);"
1390 DRAW "BM 119,26;XNO$(2);":D
RAW "BM 125,27;XNO$(0);"
1400 DRAW "BM 4,103;XLY$;"
1410 DRAW "BM 64,188;XLX$;"
1420 DRAW "BM 136,36;XNO$(1);"
1430 DRAW "BM 136,60;XNO$(5);"
1440 DRAW "BM 136,90;XNO$(10);"
1450 DRAW "BM 136,120;XNO$(15);"
1460 DRAW "BM 136,150;XNO$(20);
1470 DRAW "BM 136,180;XNO$(25);"
1480 DRAW "BM 203,135;XLX$;":LIN
E(212,135)-(231,135),PSET
1490 DRAW "BM 203,145;XLY$;":LIN
E(212,145)-(231,145),PSET
1500 '***MOVE GRID***
1510 FORY=122 TO 154 STEP 8: LIN
E(156,Y)-(188,Y),PSET:NEXTY
1520 FOR X=156 TO 188 STEP 8:LIN
```

```
E(X,122)-(X,154),PSET:NEXTX:PAIN
T(X3,Y3):GOSUB1730:GOSUB1760:DRA
W "BM 175,166;XBK$;XLS$;XLE$;XLT
$;"
1530 '***COMMANDS***
1540 X$=INKEY$
1550 J=JOYSTK(0):K=JOYSTK(1):P=P
EEK(65280)
1560 IF J=0 THEN X1=X1-1: X2=X2-
6: IF X1<=1 THEN X1=1: X2=15
1570 IF J=63 THEN X1=X1+1: X2=X2
+6: IF X1=>20 THEN X1=20: X2=129
1580 IF J=0 OR J=63 THEN GOSUB17
30
1590 IF K=0 THEN Y1=Y1-1: Y2=Y2-
6: IF Y1<=1 THEN Y1=1: Y2=33
1600 IF K=63 THEN Y1=Y1+1: Y2=Y2
+6: IF Y1=>25 THEN Y1=25: Y2=177
1610 IF K=0 OR K=63 THEN GOSUB17
60
1620 IF PPOINT(X2-2,Y2-2)=0 THEN
 PSET(X2,Y2):PSET(X2-1,Y2):PSET(
X2+1,Y2):PSET(X2,Y2+1):PSET(X2,Y
2-1):PRESET(X2,Y2):PRESET(X2-1,Y
2):PRESET(X2+1,Y2):PRESET(X2,Y2+
1):PRESET(X2,Y2-1):GOTO 1640
1630 PRESET(X2,Y2):PRESET(X2-1,Y
2):PRESET(X2+1,Y2):PRESET(X2,Y2+
1):PRESET(X2,Y2-1):PSET(X2,Y2):P
SET(X2-1,Y2):PSET(X2+1,Y2):PSET(
X2,Y2-1):PSET(X2,Y2+1)
1640 IF P=126 AND RS=0 OR P=254
AND RS=0 THEN PRESET(X1+X4,Y1+Y4
):PAINT(X2,Y2):GOTO1540
1650 IF P=126 AND RS=1 OR P=254
AND RS=1 THEN PSET(X1+X4,Y1+Y4):
 LINE(X2-2,Y2-2)-(X2+2,Y2+2),PRE
SET,BF:GOTO1540
1660 IFX$=""THEN1540
1670 IFX$="E"THEN LINE(174,156)-
(230,166),PRESET,BF:DRAW"BM175,1
66;XBK$;XLE$;XLN$;XLD$;":GOTO211
0
1680 IFX$="R"THEN SOUND1,1:RS=1:
LINE(174,156)-(230,166),PRESET,B
F:DRAW "BM 175,166;XLR$;XLE$;XLS
$;XLE$;XLT$;":GOTO1540
1690 IFX$="S"THEN SOUND1,1:RS=0:
LINE(174,156)-(230,166),PRESET,B
F:DRAW "BM 175,166;XBK$;XLS$;XLE
$;XLT$;":GOTO1540
1700 IF X$="M"THEN SOUND1,1:LINE
(174,156)-(230,166),PRESET,BF:DR
AW"BM175,166;XLM$;XLO$;XLV$;XLE$
;":GOSUB1800:GOTO1540
1710 GOTO1540
1720 '***POS. NUMBERS***
1730 LINE(209,123)-(234,134),PRE
SET,BF
1740 DRAW "BM213,133;XNO$(X1);"
1750 RETURN
1760 LINE(209,136)-(234,144),PRE
SET,BF
1770 DRAW "BM213,143;XNO$(Y1);"
1780 RETURN
1790 ***LOC. ON MOVE SCREEN***
1800 GOSUB2290
1810 X$=INKEY$: IF X$=""GOTO 181
0
1820 IF X$<>CHR$(10) AND X$<>CHR
$(9) AND X$<>CHR$(8) AND X$<>CHR
$(94) AND X$<>"S" AND X$<>"R" TH
EN 1810
1830 LINE(X3-3,Y3-3)-(X3+3,Y3+3)
,PRESET,BF
1840 IF X$=CHR$(10) THEN SOUND1,
1: GOSUB 2290: Y4=Y4+25: Y3=Y3+8
: IF Y4>92THEN Y4=92: Y3=150: SO
UND10,5
1850 IF X$=CHR$(9) THEN SOUND1,1
: GOSUB 2290: X4=X4+20: X3=X3+8:
 IF X4>217THEN X4=217: X3=184: S
OUND10,5
1860 IF X$=CHR$(8) THEN SOUND1,1
: GOSUB 2290: X4=X4-20: X3=X3-8:
 IF X4<157THEN X4=157: X3=160:SO
UND10,5
1870 IF X$=CHR$(94) THEN SOUND1,
1: GOSUB 2290:Y4=Y4-25: Y3=Y3-8:
 IF Y4<17THEN Y4=17: Y3=126:SOUN
D10,1
1880 PAINT(X3,Y3)
1890 IF X$="S" THEN SOUND1,1: GO
SUB2290: RS=0: LINE(174,156)-(23
0,166),PRESET,BF:GOSUB1940:DRAW
"BM175,166;XBK$;XLS$;XLE$;XLT$;"
:RETURN
1900 IF X$="R"THENSOUND1,1: GOSU
B2290: RS=1:LINE(174,156)-(230,1
66),PRESET,BF:GOSUB1940:DRAW "BM
175,166;XLR$;XLE$;XLS$;XLE$;XLT$
;":RETURN
1910 GOSUB2290
1920 GOTO1810
1930 '***SCAN SCREEN***
1940 LINE(209,123)-(234,134),PRE
SET,BF
1950 POKE 65495,0
1960 XS=9:YS=27
1970 FORI=1TO25:SY=I+Y4:YS=YS+6
1980 LINE(209,136)-(234,144),PRE
SET,BF
1990 DRAW "BM213,143;XNO$(I);"
2000 FORN=1TO20:SX=N+X4
2010 XS=XS+6
2020 LINE(209,123)-(234,134),PRE
SET,BF
2030 DRAW "BM213,133;XNO$(N);"
2040 A(N)=PPOINT(SX,SY)
2050 IF A(N)=0 THEN PAINT(XS,YS)
 ELSE LINE(XS-2,YS-2)-(XS+2,YS+2
),PRESET,BF
2060 NEXTN:XS=9:NEXTI
2070 LINE(209,136)-(234,144),PRE
SET,BF:DRAW "BM213,143;XNO$(Y1);
"
2080 LINE(209,123)-(234,134),PRE
SET,BF:DRAW "BM213,134;XNO$(X1);
"
2090 POKE 65494,0
2100 RETURN
2110 LINE(1,1)-(256,14),PRESET,B
F:LINE(1,15)-(154,120),PRESET,BF
:LINE(1,120)-(256,192),PRESET,BF
2120 FORI=1TO2000:NEXTI
2130 CLS:PRINT:PRINT:PRINT:PRINT
"        SAVE GRAPHICS"
2140 PRINT"          =============
"
2150 PRINT"          DRAWINGS NAME
"
2160 PRINT"               UP TO":PR
INT"        8 CHARACTERS"
2170 PRINT:PRINT"          ------
--/DRW";
2180 PRINT@294," ";:INPUTNA$
2190 IF NA$="" THEN2150
2200 SAVEM NA$+"/DRW",3584,9727,
3584:END
2210 CLS:PRINT:PRINT:PRINT:PRINT
"        LOAD GRAPHICS"
2220 PRINT"          =============
"
2230 PRINT"          DRAWINGS NAME
"
2240 PRINT"               UP TO":PR
INT"        8 CHARACTERS"
2250 PRINT:PRINT"          ------
--/DRW";
2260 PRINT@294," ";:INPUTNA$
2270 IF NA$=""THEN2230
2280 LOADM NA$+"/DRW":SCREEN1,1:
GOTO1240
2290 GET(X4+1,Y4+1)-(X4+20,Y4+25
),M,G
2300 PUT(X4+1,Y4+1)-(X4+20,Y4+25
),M,NOT
2310 RETURN
```

CoCo Cat


SOMEBODY'S GOT TO DECIDE WHO RATES MORE ATTENTION IN THIS HOUSE – THE COMPUTER OR ME!
AFTER ALL ... WHATS MORE IMPORTANT – SPEED OR BEAUTY?
HOO-BOY! THAT WAS A BAUD COMPARISON!

Stretch, reduce and enlarge your drawing creations

ZOOM —

FOR those of you who love playing with graphics, here's a simple, short and almost infinitely versatile program.

Using Zoom-Stretch you can change a small image to a larger image or vice versa. But you can also stretch or squeeze the image like Silly Putty. Does that sound like what you've been waiting for? Well start typing.

Listing 1 shows the main program in a simple menu format. It is self prompting and uses eight graphics pages. The original rectangle will be taken from the image in PMODE 4,1 and OR-ed with the image in PMODE 4,5 (within the rectangle specified).

First, draw or load the original image to PMODE 4,1. Determine what part of that image you would like to transfer to the destination screen using the top left (X1, Y1) and bottom right (X2, Y2) coordinates (they must be entered in that order). Next, determine where and what size the end result will be on the destination screen in the same manner (X3, Y3) (X4, Y4).

Load and run Zoom-Stretch. When the menu appears, you may want to clear the destination screen (unless you are adding to what's already there). Next enter Zoom-Stretch Mode 1. The computer asks for the origin and destination coordinates. Enter them as instructed and watch the CoCo do its work. Press any key to return to the menu. It's as simple as that.

To help you understand how it works and to show some examples of the variety of possibilities of this program, I have included three modifications which use Zoom-Stretch to create interesting graphics effects.

Listing 2 is a modification which draws a flower on the original screen (PMODE 4,1) and deposits 12 flowers of various sizes onto the destination screen (PMODE 4,5) to produce a pretty flowered pattern on the screen. Add these lines to the original screen as shown. When run, this starts producing the flower pattern, then returns to the menu. Enter, save, then run it and see it do its stuff.

Listing 3 is another modification to Listing 1. This one draws five-pointed stars of decreasing size on the destination screen from one star drawn on the original screen.

Listing 4, another modification, draws the word "RAINBOW" on the original screen and stretches and squeezes the letters onto the destination screen. This illustrates the Silly Putty effect.

If you like to work with machine language, run Zoom-Stretch and save the machine language to shorten the program. The ML routine is fully position independent. The value of PC and the cleared memory area must be changed to accommodate the new position.

Run the program and enter:-

```
CSAVEM"ZOOM/ML", PC+16, PC+186, PC+26
```

Delete the data statements and change Line 15 to read:-

```
15 CLOADM"ZOOM/ML"
```

If you have a disk system, change CSAVEM to SAVEM and CLOADM to LOADM. If you don't understand machine language, don't worry. Just type in the program as it is and don't change anything.

Does this give you ideas or new possibilities? Maybe you can load the latest drawings of your friends and stretch them a bit. How about wallpaper patterns with more variety? You are limited only by your imagination and 49,152 pixels. Have fun!

Listing 1: ZOOM1

```
0 'ZOOM/STRETCH,1985 BY RON ROPS
ON
5 PCLEAR8:PMODE4,5:CLEAR200,&H7F
00
10 PC=&H7F01
15 FORA=PC+16 TO PC+186:READB$:P
OKEA,VAL("&H"+B$):NEXTA
20 DATA 80,40,20,10,8,4,2,1,0,0,
31,8C,F3,5F,B6,0,19,83,18,0,1F,3
,83,18,0,1F,1,A6,32,C6,20,3D,30,
8B,A6,3E,A7,3A,C6,20,3D,33,CB,A6
,30,E6,3C,E7,38,ED,28,8D,28,A6,3
E,A1,3A,27,F,8D,20,A6,3E,A1,3F,2
7
25 DATA 67,6C,3E,33,C8,20,20,EB,
A6,32,A1,33,27,5A,6C,32,30,88,20
,EC,36,E3,3A,ED,3A,20,D6,5F,A6,2
8,A7,30,A6,29,A7,3C,ED,38,8D,22,
A6,3C,A1,38,27,C,8D,1A,A6,3C,A1
30 DATA 3D,27,32,6C,3C,20,EE,A6,
30,A1,31,27,28,6C,30,EC,34,E3,38
,ED,38,20,DC,A6,30,1F,89,44,44,4
4,C4,7,E6,A5,E4,86,27,F,A6,3C,1F
,89,44,44,44,C4,7,E6,A5,EA,C6,E7
,C6,39
45 GOTO100
50 A$=INKEY$:IFA$=""THEN50ELSERE
TURN
100 CLS:PRINT@13,"MENUE","<1> ZO
OM/STRETCH","<2> CLEAR DEST. SCR
EEN","<3> VIEW ORIGIN SCREEN","<
4> VIEW DESTINATION SCREEN","<5>
 TRANSFER DEST TO ORIG":INPUTA:O
NA GOSUB200,300,400,500,600
110 GOSUB50:GOTO100
200 CLS:INPUT"ORIGINAL RECTANGLE
: X1,Y1,X2,Y2";X1,Y1,X2,Y2
210 IFX1>255ORY1>191ORX2>255ORY2
>191THEN200
220 INPUT"DESTINATION RECTANGLE:
 X1,Y1,X2,Y2";X3,Y3,X4,Y4
230 IFX3>255ORY3>191ORX4>255ORY4
>191THEN220
240 RH=(X4-X3)/(X2-X1):H2=(RH-IN
T(RH))*256:RV=(Y4-Y3)/(Y2-Y1):V2
=(RV-INT(RV))*256
250 POKEPC,X1:POKEPC+1,X2:POKEPC
+2,Y1:POKEPC+3,Y2:POKEPC+4,RH:PO
KEPC+5,H2:POKEPC+6,RV:POKEPC+7,V
2:POKEPC+12,X3:POKEPC+13,X4:POKE
PC+14,Y3:POKEPC+15,Y4
260 PMODE4,5:SCREEN1,1:EXEC(PC+2
6):RETURN
300 PMODE4,5:SCREEN1,1:PCLS0:RET
URN
400 PMODE4,1:SCREEN1,1:RETURN
500 PMODE4,5:SCREEN1,1:RETURN
600 PMODE4,1:SCREEN1,1:FORA=1TO4
:PCOPY (A+4) TO (A):NEXTA:GOTO40
0
```

32K ECB

STRETCH

by Ronald T. Ropson

Listing 2: ZOOM2

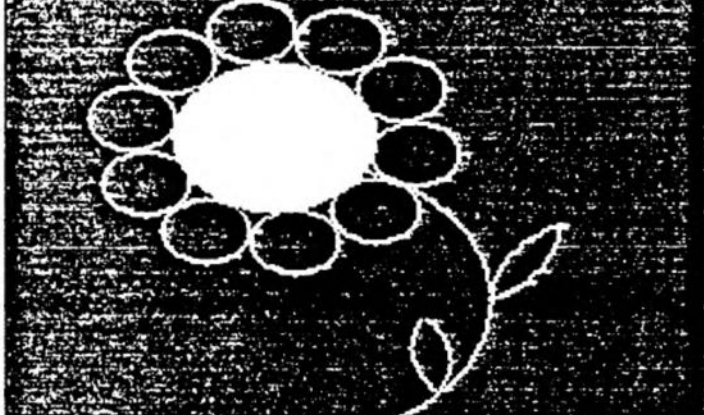
```
45 GOSUB2000:GOTO100
2000 GOSUB400:PCLS0
2005 'DRAW FLOWER
2010 CIRCLE(100,70),36,,.75:PAIN
T(100,70),1,1
2020 FORT=.15TO6.29STEP.6283:X=5
0*SIN(T)+100:Y=37.5*COS(T)+70:CI
RCLE(X,Y),16,,.75:NEXTT
2030 CIRCLE(110,128),66,,.75,.75
,.2
2040 CIRCLE(215,128),40,,.75,.5,
.7
2050 CIRCLE(160,101),40,,.75,0,.
2
2060 CIRCLE(132,150),30,,.75,.88
,.08
2070 CIRCLE(177,145),30,,.75,.38
,.58
2090 'COPY FLOWERS TO SCREEN
2100 GOSUB300:FORA=1TO12:X1=0:Y1
=0:X2=255:Y2=191:READX3,Y3,X4,Y4
:GOSUB240:NEXTA:GOSUB50:RETURN
2110 DATA0,0,48,36,32,12,132,87,
118,0,198,60,208,16,255,52,12,64
,72,112,144,48,255,140,112,56,16
0,92,64,80,144,152,16,132,64,168
,132,116,212,176,208,144,255,180
,84,156,132,191
```

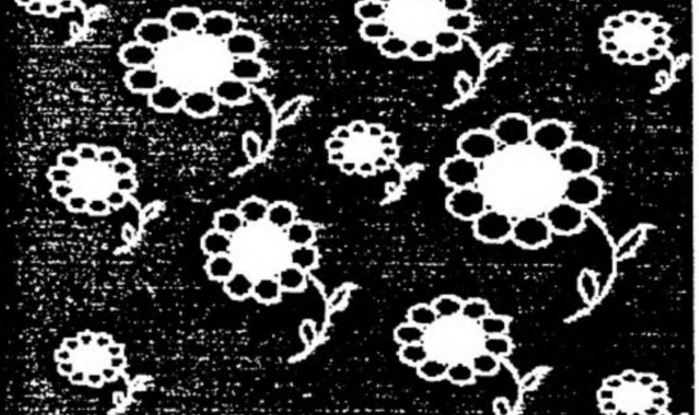
Listing 3: ZOOM3

```
45 GOSUB3000:GOTO100
3000 GOSUB400:PCLS0'5STAR
3010 C=1:X=128:Y=5:FORT=0TO6.3ST
EP.6283:A=X:B=Y:X=128-SIN(T)*121
*C:Y=96-COS(T)*91*C
3020 IFC=1THENC=.39ELSEC=1
3030 LINE(A,B)-(X,Y),PSET:NEXTT
3040 GOSUB300:FORA=0TO100STEP25:
X1=0:Y1=0:X2=255:Y2=191:X3=A:Y3=
A*.75:X4=255-X3:Y4=191-Y3:GOSUB2
40:NEXTA:GOSUB50:RETURN
```

Listing 4: ZOOM4

```
45 GOSUB4000:GOTO100
4000 GOSUB400:PCLS0'RAINBOW
4010 DRAW"BM32,106U20R10D10L10F1
0BR12U20R10D10NL10D10BR12R5NR5U2
0L5R10BR12ND20F20U20BR12ND20R10D
8G2L8R8F2D8L10"
4020 CIRCLE(162,96),11
4030 LINE(182,86)-(192,106),PSET
:LINE-(202,86),PSET:LINE-(212,10
6),PSET:LINE-(222,86),PSET
4040 GOSUB300:FORA=1TO11:X1=16:Y
1=85:X2=240:Y2=107:READX3,Y3,X4,
Y4:GOSUB240:NEXTA:GOSUB50:RETURN
4050 DATA0,10,255,18,0,30,255,90
,0,100,128,120,128,100,255,120,0
,130,85,160,85,130,160,160,160,1
30,255,160,0,170,64,180,64,170,1
28,180,128,170,192,180,192,170,2
55,180
```





The Challenge Returns: Driller II Is a Thriller, Too

By Fred B. Scerbo

Editor's Note: If you have an idea for the "Wishing Well," submit it to Fred c/o THE RAINBOW. Remember, keep your ideas specific, and don't forget that this is BASIC. All programs resulting from your wishes are for your use but remain the property of the author.

Let's face it. Sequels are a major part of our everyday life. We've had *Rocky VI*, *Psycho III* and *Poltergeist II*. If something works, we are tempted to try for a repeat of the success story. The "Wishing Well" is no exception. Some of the best programs in these pages have often been the inspiration for newer and better versions, or better yet, sequels. This month's "Wishing Well" offers a newly written sequel to a program that first appeared in these pages over three years ago: *Multi Math Driller*. So, here it is! The wait is over! You asked for it! The saga continues . . . *Multi Math Driller II*.

The Wish

The prime motivating force behind *Driller I* was a desire to counter the effects of the math software glut which had the "let's see how many aliens you can kill" approach. As a teacher, I have a bit of a problem with the idea that zapping, blasting and killing are the best ways to teach our youngsters. Not only that, but the novelty of zapping soon wears off and actually serves as a block to our learning efforts. As a student progresses, the game aspect of such software only slows down further progress.

Driller I took a different approach. Instead of a spaceship, we have a large oil rig ready to drill into the ground. Sitting above the drill was a multipication problem. Running in an underground stream below the drill was a river of moving answers which, of course, included the correct response. When the correct answer ran under the drill, pressing the spacebar or the firebutton on the right joystick caused the drill bit to sink into the ground and detect the correct answer below it.

Sound different? It was, and I received very favorable responses from parents and teachers who found the program a welcome alternative to the violent software their youngsters were too often confronted with. To be perfectly honest, this sequel is a bit overdue. However, that is one of the problems that a column like this runs into trying to grant so many wishes. To my patient readers I offer my apologies. Better than that, however, *Driller II* is now a reality.

The Program

Driller II is designed to fit into a 16K Color BASIC CoCo and the MC-10 with the 20K expansion unit. To use the program in a 16K Extended CoCo, you will need to clear the graphics memory. You may do this in two ways: PCLEAR1 or POKE25,6:NEW.

Do not use this POKE if you are using a disk drive. Use the PCLEAR1 method instead. If you have 32K or 64K, you already have all the memory you need.

Driller II, like its companion program, uses CHR$ graphics to give us an attractive nine-color screen. Since we do not need to use the Hi-Res graphics, using CHR$ colors gives a much more dramatic effect. It also gives a much larger image since the largest number we will be working with is a two digit number.

The program contains a large number of DATA statements at the end of the listing. These must be keyed in exactly, since they form the basis for our enlarged letters and numbers displayed in color. Notice that several lines have only a string of commas. Be sure to key these in just as you see them.

Several parts of the listing also use lowercase letters. Be sure to use the SHIFT/'0' to type these into your CoCo. They will appear as inversed letters on your screen. The inverse video characters give us a nice effect. Use the SHIFT/'0' to return to normal again.

Fred Scerbo is a special needs instructor for the North Adams Public Schools in North Adams, Massachusetts. He holds a master's in education and has published some of the first software available for the Color Computer through his software firm, Illustrated Memory Banks.

80	109	600	191
160	242	685	193
245	104	775	79
345	69	850	202
425	209	END	70
485	175		

The listing: DRILLER2

```
1 REM*****************************
2 REM*   MULTI MATH DRILLER 2   *
3 REM*     BY FRED B.SCERBO     *
4 REM*   COPYRIGHT (C) 1986     *
5 REM*   60  HARDING AVENUE     *
6 REM*  NORTH ADAMS, MA 01247   *
7 REM*****************************
10 CLS0
15 CLEAR500
20 FOR ZZ=1TO96:BB$=BB$+CHR$(128
):NEXTZZ
25 BR=30:YS=20
30 REM IF MC-10 THEN MC=15360
35 MC=0
40 DIM A(45,9),B(4,12)
45 FORI=2TO11:FORY=1TO9:READ A(I
,Y):NEXTY,I
50 FORI=19TO44:FORY=1TO9
55 READ A(I,Y)
60 NEXTY,I
65 FORI=1TO4:FORY=1TO12:READ B(I
,Y):NEXTY,I
70 FOR ZZ=0TO31:PRINT@ZZ,CHR$(18
8);:NEXT ZZ:FOR ZZ=320TO351:PRIN
T@ZZ,CHR$(179);:NEXT ZZ:FORI=0TO
21:SET(0,I,4):SET(63,I,4):NEXT
75 W$="MULTI":C=32:L=38:GOSUB545
:W$="MATH":C=16:L=136:GOSUB545:W
$="DRILLER":C=64:L=225:GOSUB545
80 FORI=57TO61:SET(I,14,5):SET(I
,18,5):NEXT:FORI=15TO17:SET(58,I
,5):SET(60,I,5):NEXT
85 REM <SHIFT><0> FOR LOWERCASE
90 R$=CHR$(128):PRINT@417,"by"+R
$+"fred"+R$+"scerbo"+R$+R$+"copy
right";
95 POKE1467+MC,49:POKE1468+MC,57
:POKE1469+MC,56:POKE1470+MC,54
100 GOSUB685:FORI=417TO480:PRINT
@I,CHR$(128);:NEXTI
105 PRINT@353,"select"R$"speed"R
$"from"R$"fast"R$"to"R$"slow";:G
OSUB115
110 GOTO120
115 W$="1 TO 9":C=112:L=422:GOSU
B545:RETURN
120 X$=INKEY$:IFX$=""THEN120
125 X=ASC(X$):IFX<49THEN120
130 IFX>57THEN120
135 K=VAL(X$):DL=K*8
140 CLS0:W$="SELECT":C=32:L=4:GO
SUB545:W$="DESIRED":C=48:L=98:GO
SUB545
145 W$="LEVELS":C=16:L=196:GOSUB
545:W$="FROM":C=64:L=296:GOSUB54
5:GOSUB115
150 X$=INKEY$:IFX$=""THEN150
155 X=ASC(X$):IFX<49THEN150
160 IFX>57THEN150
165 K=VAL(X$)
170 CLS0:W$="DO YOU":C=80:L=5:GO
SUB545:W$="WANT THE":C=112:L=96:
GOSUB545:W$="LEVELS":L=196:C=64:
GOSUB545
175 W$="ASSORTED":C=32:L=288:GOS
UB545:W$="Y ":C=16:L=386:GOSUB54
5:W$="OR ":C=0:GOSUB545:W$="N ":
C=16:GOSUB545
180 PRINT@L+1,CHR$(190);CHR$(188
)CHR$(191);:PRINT@L+33,CHR$(128)
CHR$(188)CHR$(188);:SET(54,28,4)
185 X$=INKEY$:IFX$="Y"THEN200
190 IFX$="N"THEN205
195 GOTO185
200 AJ=1:GOTO205
205 CLS0:GOSUB215
210 GOTO220
215 W$="WHAT IS":C=16:L=3:GOSUB5
45:RETURN
220 FOR ZZ=416TO447:PRINT@ZZ,CHR
$(188);:NEXT ZZ:FOR ZZ=480TO510:
PRINT@ZZ,CHR$(179);:NEXT ZZ
225 POKE1535+MC,179
230 E=29:F=34:FORG=10TO24STEP2
235 FORI=E TO F:SET(I,G,5):NEXTI
240 SET(E-1,G+1,6):SET(F+1,G+1,6
)
245 E=E-1:F=F+1:NEXTG
250 FORI=12TO26:SET(31,I,3):SET(
32,I,3):NEXTI
255 PRINT@108,"divided"R$"by";:F
OR TP=1TO YS:NP=0:IF TR=>BR THEN
490
260 F=RND(9):IF F=LN THEN260
265 IF AJ=0 THEN E=K ·
270 IF AJ=1 THEN E=RND(K)
275 FORLL=132TO196STEP32:PRINT@L
L,R$R$R$R$R$R$R$;:NEXTLL
280 E=E*F:F=E/F:E2=E
285 IFE<10THEN300
290 EE$=STR$(E):E1=VAL(MID$(EE$,
2,1)):E2=VAL(RIGHT$(EE$,1))
295 I=E1+2:L=132:C=112:GOSUB635
300 LN=F:I=E2+2:L=136:C=112:GOSU
B635:I=F+2:L=151:GOSUB635
305 AN=E/F:F$=STR$(AN)
310 FORI=1TO6:G=RND(9):H=RND(9):
H$=STR$(G*H):F$=F$+"     "+H$:NEX
TI
315 J$=LEFT$(F$,32)
320 PRINT@448,J$;
325 L$=RIGHT$(J$,31):M$=LEFT$(J$
,1):J$=L$+M$
330 FORP=1TO DL:NEXTP:IFTR=>BR T
HEN490
335 IFINKEY$=CHR$(32)THEN370
340 REM IF MC-10 DELETE LINE345
345 POKE339,255:IFPEEK(339)=254T
HEN370
350 NP=NP+1:IFNP=150THEN360
355 GOTO320
360 PRINT@0,BB$;:W$="THINK":C=64
:L=6:GOSUB545:SOUND1,2:SOUND1,2:
SOUND1,2
365 GOTO320
370 TR=TR+1:PRINT@431,CHR$(186)C
HR$(181);:PRINT@463,CHR$(138)CHR
$(133);
375 IF AN=VAL(MID$(J$,15,4))THEN
385
380 GOTO425
385 PRINT@0,BB$;
390 FORC=16TO112STEP32:W$="CORRE
CT":L=3:GOSUB545:SOUNDC+1,1:NEXT
C
395 L=0:PRINT@0,BB$;:IFE<10THEN4
05
400 I=E1+2:L=0:C=48:GOSUB635
405 L=4:I=E2+2:C=48:GOSUB635:L=L
+4:FORI=15TO22:SET(I,2,6):NEXT:S
ET(18,0,6):SET(18,4,6):I=F+2:C=4
8:GOSUB635
410 W$=" IS":C=16:GOSUB545:W$=ST
R$(AN):C=32:GOSUB545
415 GOSUB685:PRINT@0,BB$;:GOSUB2
15:CR=CR+1:NEXT TP
420 GOTO445
425 PRINT@0,BB$;:W$="WRONG":C=64
:L=6:GOSUB545:SOUND20,1:SOUND2,1
:SOUND20,1:SOUND2,1
430 WR=WR+1:PRINT@0,BB$;:W$="TRY
":C=32:L=10:GOSUB545:FORI=1TO300
:NEXT:PRINT@0,BB$;:W$="AGAIN"
435 C=96:L=6:GOSUB545:FORI=1TO30
0:NEXT:PRINT@0,BB$;:GOSUB215:IF
NP>100THEN NP=0
440 GOTO320
445 IFTR<>YS THEN490
450 PRINT@0,BB$;:FOR JJ=448TO479
:PRINT@JJ,CHR$(159);:NEXTJJ
455 FORI=28TO10STEP-1:SET(31,I,2
):SET(32,I,2):SOUND230,1:NEXTI:F
ORI=1TO7:SET(30-I*2,10-I,2)
460 SET(33+I*2,10-I,2):SOUND230,
1:NEXTI:SET(30-I*2,11-I,2):SET(3
3+I*2,11-I,2)
465 FORI=1TO12:SET(13-I,2+I*2,2)
:SET(50+I,2+I*2,2):SOUND230,1:NE
XTI:FORI=1TO20:SOUNDRND(230),1:N
EXT
470 CLS0:W$="YOU  HIT":C=32:L=2:
GOSUB545:W$="PAYDIRT":C=64:L=98:
GOSUB545
475 W$="WITH A":C=48:L=196:GOSUB
545:W$="PERFECT":C=16:L=290:GOSU
B545
480 W$="SCORE":C=112:L=390:GOSUB
545
485 GOSUB685
490 CLS0:W$="OUT OF":C=16:L=6:GO
SUB545:W$=STR$(TR)+" TRIES":C=48
:L=96:GOSUB545:W$="YOU HAD"
495 C=32:L=196:GOSUB545:W$=STR$(
WR):C=64:L=307-(LEN(W$)*3):GOSUB
545
500 W$="MISSES":IF WR=1 THEN W$=
"  MISS"
505 C=112:L=388:GOSUB545
510 IFINKEY$=CHR$(13)THEN520
515 GOTO510
520 PRINT@483,"press"R$"enter"R$
"for"R$"another"R$"try";
525 FORI=1TO1000:NEXT
530 IFINKEY$=CHR$(13)THEN540
535 GOTO530
540 RUN
545 P=LEN(W$):FORZ=1TOP:I=ASC(MI
D$(W$,Z,1))-46
550 IFI=31THEN585
555 IFI=32THEN595
560 IFI=41THEN605
565 IFI=42THEN615
570 IFI=-14THEN625
575 GOSUB635
580 GOTO630
585 I=1:GOSUB660
590 GOTO630
595 I=2:GOSUB660
600 GOTO630
605 I=3:GOSUB660
610 GOTO630
615 I=4:GOSUB660
620 GOTO630
625 L=L+2
630 NEXT:RETURN
635 PRINT@0+L,CHR$(A(I,1)+C)CHR$
(A(I,2)+C)CHR$(A(I,3)+C);
640 PRINT@32+L,CHR$(A(I,4)+C)CHR
$(A(I,5)+C)CHR$(A(I,6)+C);
645 PRINT@64+L,CHR$(A(I,7)+C)CHR
$(A(I,8)+C)CHR$(A(I,9)+C);
650 L=L+4:RETURN
655 GOTO655
660 PRINT@0+L,CHR$(B(I,1)+C)CHR$
(B(I,2)+C)CHR$(B(I,3)+C)CHR$(B(I
,4)+C);
665 PRINT@32+L,CHR$(B(I,5)+C)CHR
$(B(I,6)+C)CHR$(B(I,7)+C)CHR$(B(
I,8)+C);
670 PRINT@64+L,CHR$(B(I,9)+C)CHR
$(B(I,10)+C)CHR$(B(I,11)+C)CHR$(
B(I,12)+C);:L=L+5:RETURN
675 PRINT@Q,CHR$(154);:PRINT@Q+3
0,CHR$(145)CHR$(128)CHR$(154)CHR
$(145);
680 PRINT@Q+63,CHR$(153)CHR$(155
)CHR$(152);:PRINT@Q+96,CHR$(152)
;:RETURN
685 FORI=1TO1500:TU=RND(9999)
690 REM MC-10 DELETE LINE695
695 IFPEEK(339)=254THEN705
```

```
700 IFINKEY$=""THENNEXT
705 RETURN
710 DATA135,140,139,143,128,143,
132,140,136
715 DATA129,143,128,128,143,128,
132,140,136
720 DATA142,140,139,131,140,129,
140,140,140
725 DATA140,140,139,140,140,143,
140,140,136
730 DATA143,133,138,140,141,142,
128,132,136
735 DATA143,140,140,140,140,143,
140,140,140
740 DATA143,140,140,143,140,143,
140,140,140
745 DATA142,140,143,128,135,136,
132,136,128
750 DATA143,140,143,143,140,143,
140,140,140
755 DATA143,140,143,140,140,143,
140,140,140
760 DATA135,140,139,143,140,143,
140,128,140
765 DATA143,140,139,143,140,139,
140,140,136
770 DATA143,140,140,143,128,128,
140,140,140
775 DATA143,140,139,143,128,143,
140,140,136
780 DATA143,140,140,143,140,140,
140,140,140
785 DATA143,140,140,143,140,140,
140,128,128
790 DATA143,140,140,143,132,143,
140,140,140
795 DATA143,128,143,143,140,143,
140,128,140
800 DATA132,143,136,128,143,128,
132,140,136
805 DATA140,141,142,128,133,138,
140,140,136
810 DATA143,129,142,143,141,130,
140,128,140
815 DATA143,128,128,143,128,128,
140,140,140
820 DATA,,,,,,,,,,,,,,,,,
825 DATA 143,140,143,143,128,143
,140,140,140
830 DATA143,140,143,143,140,140,
140,128,128
835 DATA143,140,143,143,129,143,
140,140,142
840 DATA143,140,143,143,141,130,
140,128,140
845 DATA143,140,140,140,140,143,
140,140,140
850 DATA140,143,140,128,143,128,
128,140,128
855 DATA143,128,143,143,128,143,
140,140,140
860 DATA139,128,135,141,131,142,
128,140,128
865 DATA,,,,,,,,,,,,,,,,,
870 DATA139,128,135,132,143,136,
128,140,128
875 DATA140,140,143,131,140,128,
140,140,140
880 DATA143,130,129,143,143,132,
136,143,140,128,128,140
885 DATA143,139,128,143,143,132,
139,143,140,128,132,140
890 DATA143,128,128,143,143,134,
137,143,132,136,132,136
895 DATA141,130,129,142,129,134,
137,130,140,128,128,140
```

Get better graphics while using less memory

OPTIMUM ANIMATION

By Steven R. Polsz

Have you ever designed a program using animation graphics only to find, after long hours of work, that the graphics were too elaborate and they ate up so much memory there wasn't enough room for the entire program? Have you had to settle for spaceships that look like little crosses rather than the beautiful graphics display you wanted? Perhaps your explosions had to be a series of blinking colors instead of the real thing? If you have tried any extensive animation or game design, this or some similar difficulty has probably clouded your efforts. But there is a solution to this problem — a solution that frees at least 95 percent of the memory previously reserved for graphics storage.

Let us first consider the process of creating the animation scenes: the GET statement. The usual syntax of this statement is `GET (X1,Y1)-(X2,Y2),A` where X1,Y1 are the upper-left corner coordinates and X2,Y2 are the lower-right corner coordinates of the graphics scene to be stored. The variable 'A' is the target array where the scene is stored. This array is dimensioned to match graphics scene point to array member, in a one-to-one correspondence.

Steve Polsz lives in Philadelphia and is a free-lance programmer, writer and artist. The discovery of optimum animation is due to his impatient 3-year-old, Adam, and an undiscovered typo.

Thus, the number of members in the target array is the same as the number of points in the animation scene. If we create two animation scenes that are 8 by 16 points in size, we need two 8-by-16 arrays to store them. Each of these arrays contains 128 members, and each array member consists of five bytes — a total of 1280 bytes. This is slightly less than 300 bytes short of one graphics page.

Yet if we use `PMODE 0`, the entire graphics screen (128 by 96 points) is stored on this very same page. By using the GET statement, the equivalent of two rows of the video screen is stored in the identical space utilized by the computer to store the entire screen. It seems that the GET statement is very inefficient. Then again, is it?

Our next step is to examine the actual contents of the array 'A' after the graphics scene has been stored in it. The following short program does just that:

```
10 PMODE1,1:SCREEN1,0:COLOR
4,3:P CLS
30 DRAW"S8 C1 BM 2,6 UER3F
2DRUHR 2ER3FD BM4,4 C4 R3FBR3ER3"
50 DIM A(15,5):GET(0,0)-(30,10),
A:PUT(32,32)-(62,42),A
70 GOSUB210
90 FORJ=0TO5
110 FORI=0TO15
130 PRINTI;J;A(I,J)
150 NEXTI
170 GOSUB210
190 NEXTJ
210 A$=""
230 A$=INKEY$
250 IFA$="" THEN 230 ELSE RETURN
```

Lines 10 through 50 create the spaceship graphics scene shown in the Figure, then reproduces it elsewhere on the screen by use of the PUT statement. This scene is 16 by six, therefore the associated array consists of 480 bytes — slightly more than one-fourth of one graphics page.

By pressing any key, the contents of the array 'A' are displayed, one screenful at a time. To view the next screenful, push any key.

The first two numbers are the indices of the array member, the third, its value. As you can see, the great majority of these values is zero. The non-zero members are as follows:

```
A(0,0) -2.932031 E+12
A(1,0) -1.88127596 E-36
A(2,0) -9.12340439 E-35
A(3,0) -1.14532461 E+10
A(4,0) -2.93203083 E+12
```

How should we interpret this? The first five members of 'A' are non-zero and distinct, while the remaining cells of 'A' are empty.

According to the Extended BASIC manual, the GET statement transfers the animation scene pointwise in a one-to-one mapping into the array 'A'. Therefore, we should expect, for example, the point (0,0) to be coded into A(0,0), the point (2,0) into A(1,0), and so forth until point (30,10) is mapped into A(15,5). Obviously this cannot be the case. If it were, A(0,0) through A(4,0) would be identical and none of the members of 'A' would have a zero value.

The logical assumption is that the entire graphics scene is coded into the first five members of 'A'. Let us add the following lines to our program:

```
60 DIM B(4): GET(0,0)-(30,10),
B: PUT(64,32)-(94,42),B
```

Replace lines 90 through 130 with:

```
90 FOR I= 0 TO 4
110 PRINT A(I,0); B(I)
```

Delete Line 190 and run the program.

The original spaceship appears in the upper-left corner along with two duplicates. Examine the two closely; you will find them to be identical. Push any key and the screen displays the values of A(0,0) through A(4,0) with its counterpart from the array 'B' next to it. As you see, these too are identical.

So the two-dimensional array supposedly required for the GET/PUT statement pair can be replaced by a one-dimensional array of considerably shorter length. But how can we determine the minimum size of this array?

This answer may be deduced from the information found in the BASIC manual *Getting Started with Color BASIC*. Pages 264 to 266 detail the computer's method of storing graphics information (see Table 1).

Table 1

Graphics 2R is the same as PMODE 0
Graphics 3C is the same as PMODE 1
Graphics 3R is the same as PMODE 2
Graphics 6C is the same as PMODE 3
Graphics 6R is the same as PMODE 4

As we can see from the table, there are eight points of graphics display per byte for even-numbered PMODES, and four points per byte for odd-numbered PMODES.

Taking all this into consideration, the 96 points of the spaceship graphics in PMODE 1 can be contained in 24 bytes of information. Since each member of an array contains five bytes, this scene can be fitted into an array of length five.

The GET parameters define exactly how the scene is fitted into the target array. If either more of less than five bytes are used in any row of the graphics scene, there will be a "wrap around" so that each member of the target array will be filled before the next one is written into.

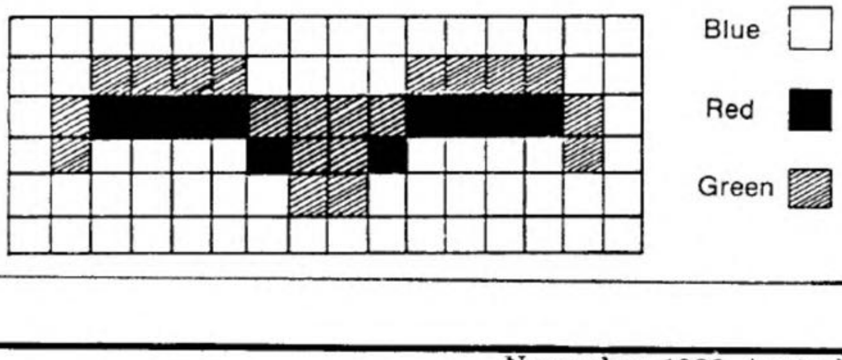
The PUT statement reverses this process, translating the scene within the confines of its parameters. Any trailing members of the information array have a zero value and are ignored.

Therefore, to most efficiently use the GET/PUT statement pair, count the number of graphics points used in the scene and divide this number by 20, if in an odd PMODE, or 40, if in an even PMODE, then subtract one.

The result is the necessary length of a one-dimensional array needed to store the scene. As you see, by use of this method, the array space is drastically reduced — a total of at least 95 percent over the usual method.

(Questions about this program may be directed to the author at 6739 Regent Street, Philadelphia, PA 19142, 215-727-7562. Please enclose an SASE when writing.)

Figure 1: Spaceship in PMODE 1



32K Disk

This program makes it easy to use several different graphics programs without all the extension changing hassle

Picture File Extension Changer

By Jeff White

In recent years many new graphics programs have come on the market for the Color Computer, each with its own unique qualities. I find myself going from one to another quite often: using one program for drawing the basic outline of the picture, another to paint it and yet another to edit my mistakes.

The problem with this is that they all use different extensions to load and save the files. It is a hassle to rename the file each time you go from one to another, particularly when you have a whole disk of files and each one must be renamed. I went to work to solve this problem and *Picture File Extension Changer* was the result.

This program changes individual file extensions or it can do an entire disk at once. It is menu driven and easy to use. The first thing that appears when running the program is the title screen. I used *Maxcmp* to convert the file to ASCII so I could merge the file with the title. You are asked if you need instructions.

The main menu appears next and there are seven options to choose from. The most popular extensions are used for options one through six. Option seven allows you to enter whatever extension you want. After selecting one of the choices, you are asked whether you want to rename all of the files or to rename individual files. If you choose all files, every file with that extension will be changed. Be careful when in this mode; if you are changing files with BIN extensions, *all* files with BIN extensions will be changed, not just picture files.

Next, you are asked to enter the drive number of the disk you want to rename. If you chose to rename all files earlier, it renames all selected extensions to the new extension. If you chose to rename individual files, a menu of all the files appears. Just enter the number next to the file you want to rename and it does it. When finished, press 'Q' to quit.

After quitting, you are asked if you want to do another disk. If so, it returns to the main menu. If not, the program ends.

Jeff White is a self-taught programmer and has had a CoCo for three years. He is president of the Carrollwood CoCo Club and owner of Merlin's Software. Jeff lives in Tampa, Florida.

Editor's Note: The following listing must be entered exactly as it appears in the magazine. To generate the underscore(_) use SHIFT-up arrow. To generate the backslash (\) use SHIFT-CLEAR.

		1260	136	1910	81
260	192	1360	133	1990	197
420	203	1520	213	2080	189
600	137	1640	3	2170	134
840	24	1730	203	2260	182
1070	178	1820	195	END	126

The listing: EXTCHNGR

```
10 'PICTURE FILE
   EXTENTION CHANGER
20 'BY JEFF WHITE
30 '(C) 1986
40 '
50 '
60 'MERLIN'S SOFTWARE
70 '1304 FOUR SEASONS BLVD.
80 'TAMPA, FLA. 33613
90 '(813) 971-4451
100 B=3:CLS(B)
110 PMODE4,1
120 POKE179,1
130 PCLS
140 SCREEN1,1
150 GOSUB1510
160 FORT=1TO900:NEXTT
170 PMODE4,1:SCREEN1,1
```

```
180 FORI=1 TO 152:LINE(0,I)-(256
,I),PRESET:NEXT
190 A$=INKEY$:IF A$="" THEN 190
200 FORI=192 TO 152 STEP-1:LINE(
0,I)-(256,I),PRESET:NEXT
210 IF A$="Y" THEN 1240
220 CLEAR2000
230 B=3
240 DIM C$(11),PIC$(68)
250 CLS(B)
260 PRINT" picture file extentio
n changer"
270 POKE1024,32:POKE1032,32:POKE
1037,32:POKE1047,32:POKE1055,32
280 PRINT@96,"1. RENAME <BIN> TO
 <MAX>"
290 PRINT"2. RENAME <MAX> TO <BI
N>"
300 PRINT"3. RENAME <BIN> TO <PI
C>"
310 PRINT"4. RENAME <PIC> TO <BI
N>"
320 PRINT"5. RENAME <BIN> TO <PI
X>"
330 PRINT"6. RENAME <PIX> TO <BI
N>"
340 PRINT"7. RENAME <ANOTHER <EX
T>"
350 PRINT@185,"pick a";:PRINT@21
7,"number"
360 POKE1213,32
370 O$=INKEY$:IF O$="" THEN 370
380 IF O$="1" THEN OE$="BIN":NE$
="MAX"
390 IF O$="2" THEN OE$="MAX":NE$
="BIN"
400 IF O$="3" THEN OE$="BIN":NE$
="PIC"
410 IF O$="4" THEN OE$="PIC":NE$
="BIN"
420 IF O$="5" THEN OE$="BIN":NE$
="PIX"
430 IF O$="6" THEN OE$="PIX":NE$
="BIN"
440 IF O$="7" THEN PRINT@352,"";
:INPUT"OLD EXTENTION";OE$:PRINT@
352,"                     ":PRIN
T@352,"";:INPUT"NEW EXTENTION";N
E$
450 O=VAL(O$)
460 IF O<1 OR O>7 THEN 370
470 PRINT@352,"RENAME (1)ALL OR
(2) INDIVIDUAL?";
480 R$=INKEY$:IF R$="" THEN480
490 IF R$="1" THEN A=1:GOTO520
500 IF R$="2" THEN A=2:GOTO520
510 GOTO480
520 PRINT@448,"           (ENTER
)= 0"
530 POKE1471,95
540 PRINT@416,"";:INPUT"ENTER DR
IVE NUMBER (0,1,2,3)";K
550 IF K<0 OR K>3 THEN 530
560 DRIVE K
570 CLS(B)
580 GOSUB800
590 'RENAME INDIVIDUAL FILES
600 PRINT@392,"enter the number"
;
610 PRINT@425,"of the picture";
620 POKE1425,32:POKE1421,32
630 POKE1448,32:POKE1451,32:POKE
1455,32:POKE1463,32
640 PRINT@456,"to be renamed";
650 POKE1482,32:POKE1485,32:POKE
1493,62
660 POKE1494,32:POKE1495,32
670 PRINT@488,"type (q) to quit"
;
680 POKE1516,32:POKE1517,60:POKE
1519,62:POKE1520,32:POKE1523,32
690 PRINT@470,"";:LINE INPUT"";F
$
700 FORT=1496TO1503:POKE T,62:NE
XT
710 IF F$="Q" THEN 1140
720 F=VAL(F$)
730 IF F<1 OR F>C THEN 660
740 CLS(B)
750 PRINT@192," ";PIC$(F);" IS
NOW BEING RENAMED"
760 P$=PIC$(F)+"/"+NE$
770 RENAME PIC$(F)+"/"+OE$ TO P$
780 GOSUB800
790 GOTO600
800 'GET FILE NAMES
810 FOR X = 3 TO 11
820 DSKI$ K,17,X,A$,B$
830 IF (LEFT$(A$,1)=CHR$(&HFF))
THEN 850
840 C$(X)=A$+LEFT$(B$,127):NEXT
X
850 X=X+1:C=1
860 FOR Y = 3 TO X:FOR Z=0 TO 7
870 IF MID$(C$(Y),Z*32+9,3)<> OE
$ THEN 920
880 PIC$(C)=MID$(C$(Y),Z*32+1,8)
890 L$=LEFT$(PIC$(C),1)
900 IF (L$=CHR$(0) OR L$=CHR$(&H
FF)) THEN 920
910 C=C+1
920 NEXT Z:NEXT Y
930 IF A=1 THEN GOSUB1030
940 C=C-1
950 IF C=0 THEN 1470
960 MID=INT(C/2)+1
970 CLS(B):TAB=1
980 FOR D = 1 TO C
990 PRINT@TAB,USING"##";D;:PRINT
".--> ";PIC$(D);
1000 TAB=TAB+32:IF D=MID THEN TA
B=16
1010 NEXT D
1020 RETURN
1030 'RENAME ALL FILES
1040 FORD=1 TO C-1
1050 IF C=1 THEN 1470
1060 PRINT@224,"<"+OE$+"> FILES
NOW BEING RENAMED TO<"+NE$+"> FI
LES."
1070 P$=PIC$(D)+"/"+NE$
1080 RENAME PIC$(D)+"/"+OE$ TO P
$
1090 NEXTD
1100 CLS
1110 DIR
1120 PRINT@384,"ALL <"+OE$+"> FI
LES HAVE NOW BEEN   RENAMED TO <
"+NE$+"> FILES."
1130 GOTO1160
1140 CLS:DIR
1150 PRINT@384,"ALL SELECTED <"+
OE$+"> FILES HAVE   NOW BEEN CHA
NGED TO <"+NE$+"> FILES."
1160 PRINT@448,"DO YOU WISH TO D
O ANOTHER DISK?            (yes
/NO)            ";
1170 FORT=1TO300:NEXT
1180 PRINT@448,"DO YOU WISH TO D
O ANOTHER DISK?            (YES
/no)            ";
1190 FORT=1TO300:NEXT
1200 A$=INKEY$:IF A$="" THEN1160
1210 IF A$="Y" THEN 220
1220 IF A$="N" THEN 1230 ELSE 12
00
1230 POKE113,0:EXEC40999
1240 'INSTRUCTIONS
1250 CLS(B)
1260 PRINT"            INSTRUCTION
S         ";
1270 PRINT" MANY NEW GRAPHIC PRO
GRAMS HAVE COME ON THE MARKET RE
CENTLY THATUSE EXTENTIONS OTHER
THAN THE   STANDARD <BIN>. THERE
 MAY BE    ";
1280 PRINT"TIMES WHEN YOU WOULD
LIKE TO USEA FILE FROM ONE BUT I
T HAS TO BERENAMED BECAUSE OF TH
E EXTENTION";
1290 PRINT"DIFFERENCE. MANY TIME
S YOU MAY  WISH TO WORK ON A FUL
L DISK OF  FILES BUT YOU WOULD H
AVE TO GO  ";
1300 PRINT"AND RENAME EVERYONE.
WELL THIS  PROGRAM WILL HELP YOU
 OUT.      ";
1310 PRINT@448,"   PRESS SPACEBA
R TO CONTINUE    ";
1320 EXEC44539
1330 CLS(B)
1340 PRINT" PICTURE FILE EXTENTI
ON CHANGER WILL CHANGE THOSE EXT
ENTIONS FORYOU. THE PROGRAM IS M
ENU DRIVEN AND VERY EASY TO USE.
 YOU HAVE A";
1350 PRINT"CHOICE OF RENAMING AL
L OF THE   FILES OR SOME OF THE
FILES. "
1360 PRINT" IF YOU CHOOSE TO REN
AME ALL THEFILES IT WILL CHANGE
EVERY <BIN>FILE ON THE DISK SO C
HECK TO    MAKE SURE ALL THE FIL
ES ON THE  DISK ARE PICTURES."
1370 PRINT:PRINT
1380 PRINT@448,"   PRESS SPACEBA
R TO CONTINUE "
1390 EXEC44539
1400 CLS(B)
1410 PRINT" IF YOU CHOOSE INDIVI
DUAL YOU   CAN PICK WHICH ONES Y
OU WANT TO CHANGE. BASIC FILES A
RE IGNORED IN BOTH CASES."
1420 PRINT" THAT IS ABOUT ALL YO
U NEED TO  KNOW. I HOPE THIS PRO
GRAM IS AS HANDY TO YOU HAS IT H
AS BEEN FORME."
1430 PRINT:PRINT:PRINT:PRINT:PRI
NT
1440 PRINT@448,"   PRESS SPACEBA
R TO CONTINUE "
1450 EXEC44539
1460 GOTO 220
1470 CLS(B):DIR
1480 PRINT:PRINT"THERE ARE NO <"
+OE$+"> FILES ON DISK"
1490 FORT=1TO4000:NEXT:GOTO220
1500 'TITLE PAGE DATA
1510 FOR X=&H7E00 TO &H7E95
1520 READ H$:POKE X,VAL("&H"+H$)
:NEXT
1530 DATA 9E,33,30,6,10,8E,5E,0,
86,8,A7,8C,3A,86,6,A7,8C,34
1540 DATA A6,80,80,30,48,48,48,5
9,6A,8C,29,27,E,6A,8C,25,26
1550 DATA F4,E7,A0,C6,8,E7,8C,1C
,20,EB,86,6,A7,8C,14,A6,80,26,A
1560 DATA A6,4,81,22,26,C,30,5,A
6,80,80,30,48,48,20,D9,0,0
1570 DATA 8E,5E,0,10,8E,E,0,A6,8
0,A7,8C,42,6F,8C,40,A6,80,A1,8C,
3A
1580 DATA 26,F,E6,80,A6,80,A7,A4
,8D,10,8D,1D,5A,26,F7,20,4,A7,A4
1590 DATA 8D,5,8D,12,27,E2,39,10
,8C,25,E0,24,4,31,A8,20,39,31,A9
1600 DATA E8,21,39,6D,8C,F,26,B,
10,8C,25,FF,26,3,6C,8C,4,1A,4,39
1610 READ Z:EXEC&H7E00
1620 RETURN
1630 DATA1
1640 "j^X40>ZB3nX40>XR3nX80>X1oo
cmo?cZ30olo0cljPgojS01jP[oo?Wbm^
C
1650 "Z1>cTi_;an?cZ1ooZ103Z8_oZ2
03Z6010of1PIVMVI65oHF1TIVMWIV3o0
0
1660 "3Z30100>X:onX4o^X0ooc``>X5
0>X:oc10?V5QHV1PI6EQ?000?nX7onX4
0
1670 "?ooo_cmo0clooooo?glo>XConX
80>XIo`o7ifLW00NAT9bNWQ17YRIThLL
0
1680 "0>X=o`00jP[o7`110020h?3`1>
3Ph<2``;RHh?clo0gno_oon?SoookPP>
X
1690 "90>X:o`m78c<A6AUI6A4a8d<73
nX7onX40?n78?iSHF7Z16EQoP00jQ?oj
P
1700 "P0jQko0=n4a<QXLC5aFIR<3=10
0>X=o`00h>X41>X8n60H10?Z100LOWUg
K
1710 "f1_=cTmo0cm3NGSL0031`D11PL
71`010@000>X;ookln0_in0gmo0Wkn?S
Z
1720 "2?oZ103ooglo?`20jPBa<ah0@>
XConX80>XKoo3di0`o30cZ2>`_9b0`jP
g
1730 "ojQ<0`20N0@000=k_knoL?oom0
I_S`80o0ooolg>c@Z60Kooi0000jP_o0
8
1740 "oZ19VKSH6AcP00jPSojP@0jPGo
?aoZ19107cmojQ?ojPP0jQko?SPc=^K\
k
1750 "nc\k0I7P`0H?^X=o`0000P8420
Q@T;Z1H@40024Q8@4onX40=b109RHn?S
m
1760 "o0]i<023k1R69UShn7RHh?P000
3Z2o107[>c\ijc\k<c7P10jPSojP@0oo
n
1770 "2<W8b<[:b\;6bW831k^3PjQ3oj
PP0oon0_hb<jPK\k^gZ1Nb<_h20jP0o0
o
1780 "0L30H6oP00N28<n>011nX=o`00
0210P80000410`<71`1?3a107ogZ1@00
7
1790 "fm_ccoZ1?10?oooo`;Z303Z2om
0729PK7KZ16H1003Z2?oZ103oo`o?c13
?
1800 "jPCHfLn0011o0nX@onX80?oo0o
0<1^;cno_clPk11003oooZ13oZ1ooPbA
]
1810 "161SZ1UTH0P00jPgo000?3alo0
goZ3_mo7`1?7nX@oclo0NX90>X<onX;0
n
1820 "X9onX40>X5o`2<g<c<c8d6033Z
4ooZ203oon3\K60P<2@<70`<K6cZ1>b<
_
1830 "h20jP0o1n?a>AVIjPKI67h00>X
=o`41jRci6@Ta@D7Z107Z8?oZ103Z101
7
1840 "IfMWHNUaH@<7jQ?ojPP0jP0on>
?<f9Ra\K>c\IRL3`<P>>X4on7YbAQn61
S
1850 "Z11WHf5m>@63Z3ooll1NO[jn?Sn
o_khnoPh>Y7onX40?ool>O[jn;Rn_[jh
^
1860 "cPh>XConX80>X6ooh0hg0`4X;2
o_k2TS;RPPH0ooo`m_K`17QbIVkVi^X4
m
1870 "QJF1Q10@>X?oc14hC4`<N?Z11W
a003ZAooZ103ooc0FQ<CTM3`L34ATL?3
o
1880 "o_cino;Z1?KbloWho?kooooZ20
3Z1?1?Cdo21\;2jPK?`<ChL043oooZ1G
m
1890 "1LFI\C5QHFEUHC4`6PA0LjQ3o?
clo48J>QhJFY\H00>Y7onX40?oo0gmo0
7
1900 "c5bM7SaGP00?10Wa10?cmo0glo
?an07aooooooZ203Z1o16iVIVjPIWHW0o
7
1910 "X30jP0o1n4H34cVi_Kfi^a<6>0
11nXCo`=c\c<`=3P`0@?ZAooZ103Z101
0
1920 "nH^CX1N;1@00oon2<W8b<[:b\;
6bW831k^3PonX80>X7o`KTHF=QHNX4IF
7
1930 "aHD<02>X7on396g`H6>X6FAQn0
03Z8_oln0?gjPKc1?SZ=_oZ103oh<kHf
1
1940 "Wi6ATH>EZD003oo`oZ2\n?3alo
0ooZ203Z1o10W;k6P83Z1Yn7i`L7jP0o
1
1950 "n?a>AVIjPKI67h00>XRoclo0<L
<30`\C8L04>XQon68?eUI6AoHf=PH0`0
0
1960 "jP0ojP@0ogmn07gmo?cojP@o0o
oon?[bm>CXb=6AXj>7QnX5onX80>X7oo
1
1970 "7630`H7mPH6<a>9n7`?3Z4ooZ1
7oZ9?10cnX4K1n?3aoZ10oin0[jn?Sno
_
1980 "khnoShjQ3o80bLS8`<jPC<Q`0@
noShjPCojP@0oo131c<b<36`\;:d^820
o
1990 "gmn07emo?cooooolo0cljPCojPP
0jP0o7h0WHc<cl`<3`a=S``L??nX7onX
;
2000 "o^Xho`clhoFiTH61WIdM73cZ4?
n3jPJc<g>c<b=3Q`10jPCojP@0jPGo0<
k
2010 "Fi\K6alH00?ool^9`N5a>Ad=iB
8`>7^X4onX80>Xloo37V31QH107jPG?a
f
2020 "MS<AR7`?3Z<ol?chn0Qhb<S9b]
aP00jQ3ol>OYjN7QjPGmnN;\h>3Z1?oZ
1
2030 "03Z1010?69P?8I6?000ool?chn
?jPN07aoZ1?oZ203oon3_hn?Z30_Skn3
P
2040 "ool7hW0a4832jP0n`Y8bL><00>
Xfo`3SIF1Pj75R003Z3oo`19n\[8b?jP
B
2050 "<_`00?goZ10oZ103ooon79f3iH
f5QIFUa003oQj>AVLS<k>c\b=RAXH>7o
o
2060 "oojPP0oo10oPL3<CP`90c130@P
<31o?`o?3`ooo`m?clo?`<o0hL34jPK>
`
2070 "?<00>Xfob49TN5S\1X<063Z3oo
10NL[2`>36`>C81h00>X7onX40>X5o`2
c
2080 "ah>3R1`921cZ4ooZ203oof0\;2
0P<2C<g<c<jPK\S;h0P>X7oclLRL_;b<
S
2090 "<bL_;b1?a003Z30oZ;_cZ17cZ1
0cZ2olo0?T:0P3P1PHn>3\hN>X7onX40
>
2100 "X5o`0H6ATI>efH003Z4ooZ203o
oo3Tc=c<c>X;k0`o003Z1o1008h61Phf
a
2110 "XH63Qkg``08jPgo003Z17X2o`8
2NWYjN03100;Z17X2o`82jPAj0_120^X
4
2120 "NP;o0P;Z17X2o^X70>X;o`m0o\
;0`?W1`Lo?S`10jP0ojP@0ooooo_`1nh
V
2130 "1PHVAh@04jQ?ojPP0jP0on>?<f
9R`_k2`\IRL3`<P>>X7ooQbAUm6A^X5M
Q
2140 "J67`00jPgo003Z17X2o`82NWYj
>P830000@61P0<00001PL70``000Ng]i
N
2150 "0;o001jNWQ0jPT0jPco0Wam07c
Z2?kZ1ooZ103ooo0DU83VSXJ6UZK7003
Z
2160 "4ooZ203Z1o1?`g<a6AWi0@7QRC
7QP`L0jP0o0GS13PHVjPIf1Y100>X=o`
0
2170 "0061PH03P003Z1700100`20H10
000>?cbkmo0GfmbNo_hna\;a`001fL;0
P
2180 "d>3`100`<00>X<o`o>c\k0a<cL
kLn>003Z1ooZ103Z10m07c4a<G6c7P00
j
2190 "Q?ojPP0jQkon>=<F5QIF4a7IVa
_1h`H6<c7h?3Z20oZ2`3Z1@400001PD4
1
2200 "0`000;c0gmji0ooknc07P01noo
ooinIVQD<0gooc003Z3?10An][HfM[`h
<
2210 "33R1PjP0ojP@0jPGo01VjW9RH6
1P0@>XConX80>XNo`0P?1RHf9P`h0821
?
2220 "PL3<`Hh047jPWo0008210@8D52
Q8CZ1PP000P820WnjP@0^7Sh<37`1?[j
1
2230 "_9P00;JT0a<\?3`13301000jP_
oh0kZ11Sh6API6Q`0@>X7onX40>X5o`3
7
2240 "c<c7d>S7003Z4ooZ203Z7_oPSc
5PH<3o`<37HW0o3X3PjPgo000P@420jP
@
2250 "0jP@23`826QXj>P;kjPD0>P;0`
X;Z17X2?`82NW[Z303Z2omo0GRHV1PHf
A
2260 "PI6Qa0`>X7onX40>X5oan?C`n?
cln?3aoZ4ooZ203Z7_112L?7Hf?S0`>3
8
2270 "1>31a1`jPgo00022QX20`82jPA
j0_120^X4NP;o0P8J6SYj0_120^X4NP;
o
2280 "0P;Z17X20nX80>X<o`1oS8`<3>
`\;0`h087Z1ooZ103oo12N8d<31^cXh>
7
2290 "YhN7Z4ooZ203Z7_11>7b<003Z1
S1?c`1?jPgo0NX4N@7m0@7Z17T1o@41j
P
2300 "Ai0Od10NX4N@7m0@7Z17T1o@41
NGUiL@7a0@5QPH410@7Z2ooQ2;mHF1P0
f
2310 "=SI67100>X7onX40?oo0nX6?go
Z6?oZ203Z0_1?cd1?3clolbM7Q`10jP0
o
2320 "jP@0jR;ojPP0jY;`jP@0jR;`jP
@0Y
```

Learning How To Function in Basic

By Joseph Kolar
Rainbow Contributing Editor

Last month, we used the MID$ and LEN functions on the inverse, black screen, but we didn't hurt ourselves explaining them in detail.

To make amends, we shall repair this neglect and work with LEN, LEFT$ and RIGHT$. We are going to use the MID$ that complements LEFT$ and RIGHT$. Keep in mind that there is another form of MID$ (a statement as opposed to a function).

We'll toy around with the regular green screen and create some interesting effects that may be of use in your programming future.

The first order of business is to give an overview of LEN, LEFT$, RIGHT$ and MID$. Look at Listing 1. Key in lines 10, 20 and 100. The meat is in Line 20. We plan to display a title on the text screen - a centered name and address heading.

The entire text was enclosed, within quote marks, in one long string of letters, numerals and blank spaces. The strung-out line was assigned a name, string variable A$. The three lines of the title were scrambled and blank spaces separating the lines were omitted by personal choice.

Florida-based Joseph Kolar is a veteran writer and programmer and specializes in introducing beginners to the powers of CoCo.

Key in Line 30. L is the variable assigned to LEN(A$), the length or number of characters/spaces in the string A$.

It was chosen to display this value to achieve a dual purpose. First, to locate it in the center of the screen as a centering guide. The two-digit value begins on the 15th space. (Remember the first line is 0, not 1.) Secondly, I was curious to know how many characters/spaces there were in A$.

The top line of desired text is buried in the middle of string A$, so we may as well fetch it, using MID$ as our appropriate tool.

Key in Line 40. Picking a location on the second row, I unimaginatively chose 32 at the left margin. Later, it would be centered. MID$, the target text, was the first of three values to be enclosed within parentheses. Counting from the first character in the string until reaching J (the beginning of the segment of text to be plucked out of A$), gave the second value to be added to A$, and separated from it by a comma. Next, counting from the first letter J, the number of characters/spaces to be included (totaling 12), became the third value, again separated from the second value by a comma. Don't forget to tack on the closing parenthesis.

Now run it. Notice that it lines up along the left margin.

The address is next and, since it is at the right end of string A$, it is a candidate for RIGHT$. Key in Line 50. The locating value, 64, was chosen, although any reasonable value near the left margin would have been fine, say from 64 through 70. RIGHT$ contains two items enclosed within parentheses. The first is the target string, A$. Since all of the characters/spaces at the right end of the string would be utilized to create the second line of text, the total number of characters/spaces making up the second entry would be found by counting backwards, beginning with T, up to and including one. If you prefer, count from one to T, but it would be best to work from right to left. The value is separated by a comma from A$. In other words, the last 21 characters/spaces will be displayed on the row. Run this.

Now, since the balance of our text appears at the beginning of Line 20, the LEFT$ was called upon for help. Key in Line 60. A value of 96 was chosen as the trial location of the third row. LEFT$ is used about the same as RIGHT$, except it works from the left end, or beginning, of A$. The number of letters to be included in this row were tallied, from I through zero and this total, 19, became the other value included in LEFT$. If you look at the number of characters/spaces used in the last items in MID$ and RIGHT$, and subtract the total from L, you can see that every character/space in A$ was accounted for. This doesn't always follow if you have unnecessary spaces or unused characters in the string. Again, run your work.

Take a few minutes and adjust the lines to center them. Take a moment to change the 52-33 in Line 60 to 19, since the point has been made.

Key in lines 70 and 80. Line 70 waits for any key to be pressed and then Line 80 zaps the top row and in the process says goodbye to 52.

Ordinarily, having no further use for

Line 30, it could be deleted — but then there goes the tutorial! Of course, Line 30 could be masked with a REM marker, but that too alters the listing.

Look at lines 40 through 60. Are your PRINT@ values 46, 69, 102, respectively? They need not be exactly the same. So long as the title appears reasonably well-centered to you, that is what counts.

Who wants to practice? Using MID$, how would you put on lines 50 and 60? Better still, put your name and address into a single long string and make a nicely centered heading to demonstrate your grasp of the functions.

If you plan to use the material in string A$ more than once, you could assign a variable to the substrings in lines 40, 50 and 60. They will be ready for instant use elsewhere in your program. Insert and run the following:

```
35 A1$=MID$(A$,20,12):A2$=RIGHT
   $(A$,21):A3$=LEFT$(A$,19)
90 PRINT@170,A1$:PRINT@197,A2
   $:PRINT@230,A3$
```

Naturally, you could then substitute the three variables, A1$, A2$ and A3$ for the function statements they represent in lines 40, 50 and 60, respectively. At that point, A$ becomes a dinosaur.

Key in Listing 2 and run it to get an overview. You will note that the text was printed one complete word at a time, repeated monotonously to the screen. The original objective was to afford you practice using LEFT$, RIGHT$ and MID$, the idea being to figure out many different ways to accomplish the mission. Seven examples were sufficient to create the text panel to keep the tutorial short and succinct. No doubt, you will be able to find other techniques to get the job done. I can think of about 20 variations on this theme. The acid test is your ability to display what you intended in the manner intended by drawing upon CoCo's versatility.

List lines 5 to 50. A word about the CLEAR 500. If you masked it with a REM, you would have quickly determined that the program works OK. Masked or not, CoCo already cleared 500 memory locations. Change Line 5 to CLEAR0 and run it. Again, change Line 5 to CLEAR1 and run. Repeat this through CLEAR4. An OS Error (out of string space) in Line 30 message appears, because there are five letters in each string of D$ used. Change Line 5 to CLEAR5. The program is in good shape because, coincidentally, every string is composed of five letters.

Restore Line 5 to CLEAR 500, if you like. On power up, CoCo automatically reserves 200 string spaces.

Whenever working with strings, it is good practice to CLEAR 500. If you work with many strings or lengthy strings up to about 255 characters/spaces, CoCo may have no places allocated to store them. It cries for guidance with an OS message. Don't panic! Increase the CLEAR 500 to CLEAR 600 and run. If it still isn't enough space, try a larger figure, until CoCo has enough memory reserved to handle the load you thrust upon it. You will see an example of this in the third tutorial of this series.

Now, let us return to lines 5 through 50. In Line 30, we decided to print BETTY using LEFT$ to pick out of string D$ the first five letters and print them beginning at location eight. Since I am lazy, I used the old reliable semicolon ploy to allow me to butt up the next segment without taxing my brain figuring out PRINT@ locations. A small pause fetched from a GOSUB routine allows time to digest the display momentarily. Then, using MID$, from the same string, beginning with the sixth character/space and going up to and including the 10th character/space, ANN was appended to BETTY, followed by a semicolon and another pause. Finally, utilizing RIGHT$, the balance of the letters were put on to complete the name followed by a pause of longer duration.

You could have broken up D$ to use the first six characters/spaces in Line 30 and four characters/spaces in Line 40 and still maintain the integrity of the three segment plan of attack. Only one problem. If you left Line 5 at CLEAR5, you got the OS message because there are six characters/spaces in Line 30. OK. Make sure Line 5 reads CLEAR 500. Now run. Can you pick up the error? Failure to change the starting letter in Line 40 from six to seven generated the problem.

Can you break up D$ into some other groups without destroying the presentation, using the same functions? Now is a good time to work something out and become more familiar with the three functions.

Your fertile mind tells you that this is a lot of work to put three equal segments on the screen. True, true.

List lines 70 to 100. To save all the fuss of counting and using LEFT$, etc., in Line 70, we prefabricated the three building blocks and assigned them to separate string variables. Still being naturally lazy, to locate the starting position of the second row of text, it was simple to add 32 to the PRINT@ location usurped from Line 30. In lines 80 to 100, each name was placed exactly as in lines 30 through 50, but with less effort. List Line 100 to compare. Run this. If you were a glutton for punishment, you could revise the strings in Line 70 without altering the presentation in this part of the tutorial. Be careful: The following presentations may get thrown out of kilter.

List lines 120 to 150. Here PRINTTAB was used to get the same results. Note the necessity of the semicolon. If you don't know what will happen when you run without it, pull out the semicolon and run it.

I can't stand that last line at the bottom of the panel. Find the correct program line and edit to center it!

List lines 160 to 180. To place the text in the correct spaces on the next row, without the semicolon ploy, each string must be located individually. OK, now run. Too much calculating! Better that CoCo does the work as in the previous presentation.

In order to return to the subject at hand and clown around with LEFT$, etc., list lines 200 to 220 and see how only MID$ was used to work out the same arrangement. List lines 240 to 300 to see LEFT$ and then RIGHT$ carry the entire load to put on all three segments. Run your work.

Since each of the groups has five characters/spaces, you could use MID$, RIGHT$ and/or LEFT$ interchangeably (not their contents) and get the same results. It is no big deal to use the entire contents of a string when all are the same length.

For practice, in Line 70, add a space to A$, strip off both spaces from B$ and add a leading space to C$. Run.

CoCo is upset! Help CoCo straighten out this mess. First off, compare the distorted lines with the program lines concerned and point out and explain the whys and wherefores of the resultant boo-boos to yourself. Then make the required corrections.

". . . ideas began to perk in my noodle . . ."

Look how valuable those GOSUB routines are. The short one was used 14 times and the longer one was used seven times.

List lines 320 on. In the third tutorial, we will work on presenting text, using LEFT$, and who knows what else, a letter at a time, in a very attractive, readable manner. Lines 320 and 330 were just plopped onto the screen. Patience — you'll like it!

continued on Page 33

Use this high resolution graphics editor and let your imagination run wild!

CoCo DRAW CONCOCTIONS

By Darin Herr

CoCoDraw is a user-friendly, high resolution (PMODE 4) graphics editor. It has the usual LINE, CIRCLE, BOX, etc., functions, plus a full character set (for adding text to pictures), an UNDO command and even a Magnify mode for detailed editing. *CoCoDraw* requires 32K, Extended BASIC and one joystick (a mouse or touchpad should also work). As listed, it requires a disk drive, but it also works with a cassette system using the modifications listed at the end of this article. A printer is optional.

Type in both listings (*CoCoDraw* and *MenuGen*) exactly as they appear and save them to disk. (Do not add any extra spaces to *CoCoDraw*, as it hardly fits in the available memory as it is!) Lines 10000 and 12000 to 12080 are the same in both programs, so you do not have to type them in twice.

Run *MenuGen* first. This program generates a file (*MENUS.SYS*) which is loaded and used every time *CoCoDraw* is run. Make sure that each disk with *CoCoDraw* on it also contains *MENUS.SYS*. Now run *CoCoDraw*, and you are ready to start.

When *CoCoDraw* is run, it initializes itself and asks if you want to use the speed up POKE (POKE 65495,0). Move the joystick left or right to select "yes" or "no" and press the button on the joystick when the one wanted is underlined.

Darin Herr is a sophomore at Lancaster Mennonite High School in Ephrata, Pennsylvania and a self-taught programmer. Besides computing, he enjoys tennis, biking and stamp collecting.

The main menu is on the top quarter of the screen. The rest of the screen is the editing area. A little pointer should be blinking somewhere on the screen. In the upper-left section of the menu are 14 boxes, each containing an icon (a little picture symbolizing what it does). These are called tools and are what you use to create the picture.

To the right of the tools are two larger boxes labeled "Color" and "Background." These show the current foreground and background colors (or patterns). Below the tools and colors are four words: File, Size, Misc and Undo. Each of these (except Undo) triggers a pull-down menu that allows you to do things like save, load and print pictures.

The pointer can be moved around the screen using the right joystick. However, because the joystick's resolution is smaller than the screen's, the pointer can only be positioned to the nearest four dots horizontally and three dots vertically. To compensate for this, the arrow keys can be used to move the pointer as many as three dots to the right of the joystick position and two dots below it. This is limited, but with some practice you should be able to place the pointer on any dot on the screen. When part of the pointer is off the right side of the screen, it becomes distorted. When this happens, the upper-left corner of the distortion is the current point.

To select a command from the main menu, simply position the tip of the pointer over the desired option and press the joystick button.

The Tools

When a tool is selected, its icon changes colors to let you know what you are working with. Here is how to use each tool:

Draw (pencil with point down) — Leaves a line after the pointer when the button is held down.

Erase (pencil with eraser down) — When this is selected, the pointer changes to a block eight by eight dots in size. Whenever the button is pressed, the area behind the block changes to the background color. The eraser can be made smaller using the Size pull-down menu. More on that later.

Box (the empty square) — Move the pointer to one corner of the box, then press and hold the button down while moving the opposite corner of the box. When you like it, release the button.

Circle — Position the pointer where you want the center of the circle to be, then hold the button down while adjusting the radius. To adjust the radius, put the joystick in the center vertically and move it left or right to roughly get the radius. Now, keeping the joystick at the same place horizontally, move it up or down to make fine adjustments. Release the button when you have the desired radius.

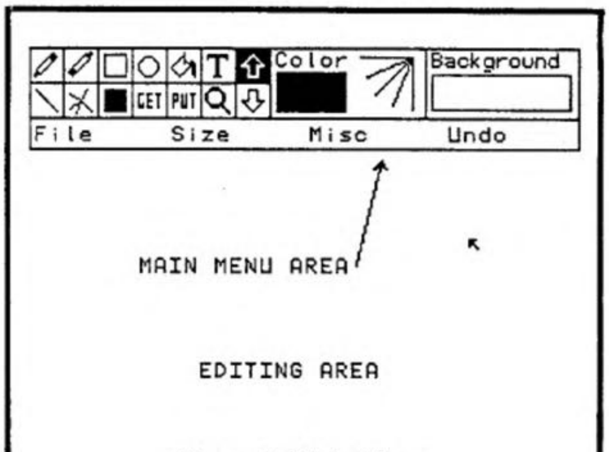


Figure 1: Main Menu

Paint (a paint can pouring paint) - Move the pointer to where you want to pour the paint and click the button. You can fill in either black or green areas.

Text (a capital 'T') - Move the pointer to where you want the first character to be and click the button. A blinking cursor appears and you may type any letter (upper- or lowercase), number or symbol on the keyboard. Press ENTER to exit this mode. The foreground color should be a solid color (not a pattern) while typing because anything else will be unreadable.

Line - Move the pointer to one end of the line, then hold the button down while moving the other end. Release the button when you like it.

Ray (several lines coming from the same point) - Move the pointer to the centre point and click the button (do not hold it down). Move the end of the first ray to where you want it and click again. Do the same for as many rays as you want, but when doing the last one, hold the button down until you hear beeping (about two seconds). Now you can start another set of rays or select another option.

Solid Box (the solid square) - This works the same as Box, but when done, the box becomes solid.

Get - This is used to get an area of the screen (up to 64 by 64 dots) to be used later with PUT. Move the pointer to the upper-left corner of the area to be gotten, then press and hold the button down. Now move the bottom-right corner until you have the size you want, and release the button.

Put - This puts what you got using GET. Move the block that you got around the screen, and whenever you press the button it will be put there. You can hold the button down while moving the joystick for some interesting results. There are five PUT modes to choose from: Set, Reset, And , Or and Not. Set puts it exactly as it was gotten: Reset reverses the original colors; And puts it without erasing what is already there;Or puts it, sho--wing what was gotten only where there is something under it(And and Or are reversed when using green or buff on black). Not reverses the colors in an area the same size as the area that was gotten.What was in the area that was gotten has no effect on this mode.How to change the PUT Mode will be described later.

Magnify (a magnifying glass) - When this option is selected, a 16-by-16 dot box appears, replacing the pointer. Move this box over the area you want magnified, then click the button.A new scr--een appears showing the area selected magnified eight times. To make changes to the original,move the pointer over the mag--nified dot you want to change and click the button. Its color will be inversed, as well as the correspo-nding dot in the Now box, to see how the change looks in actual size.If you make some changes, but then decide you liked the original better, move the pointer over the box marked Cancel, click, and the screen will be changed back to the original. When satisfied with your changes, move the pointer over the box marked Done and click. This will take you back to the main menu with the change made.

Up Arrow - Actually, the editing area you see is just 75 percent of the entire. By clicking the Up Arrow you see the top 75 percent of the picture.

Down Arrow - Shows the bottom 75 percent of the picture.

Color - In the Color box is a block showing the current color, as well a little design to show how it will look when used on diagonal lines. To change color, move the pointer anywhere in the Color box and click. A new screen appears with a selection of 256 colors and patterns. To select a color, move the flashing box over the color wanted and click. If you would rather leave the color the way it was, press the space bar (even while the screen is being drawn) and you will return to the main menu.

Background — Works the same as Color, but changes the background color, which is used when erasing and when clearing the screen.

Pull-Down Menus

To use the pull-down menus, move the pointer over the word File, Size, or Misc and hold the button down. Another menu appears below it. Move the joystick up and down until the selection you want is highlighted, then let the button up.

The File menu includes the following:

Disk Load/Save — You are asked for a filename, and then asked if it is OK. If not, you return to the main menu. No error trapping is used in the disk I/O, so if you get some type of error, just type GOTO 700 and press ENTER to return to the main menu.

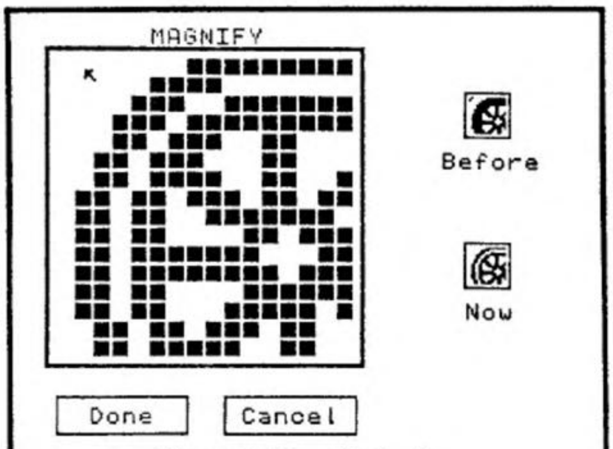


Figure 2: Magnify Option

Disk Dir — You are asked for the drive number, and the directory of the disk in that drive is shown. Press the SHIFT and '@' keys together to pause the display, and click the joystick button to return to the main menu.

Tape Load/Save — Same as disk. When saving, make sure the tape recorder is set to Record before saving because recording starts right away.

Screen Dump — I have included a routine that does a double-size screen dump to the Epson RX-80. Make sure the printer is online and the proper Baud rate has been set before saying the printer is ready.

The Size menu is used to change the eraser size. Just select the size you want (8 by 8, 4 by 4, or 1 by 1) and click.

The Misc menu includes:

Clear Screen — Clears the screen, but only the editing area being seen. To clear the whole picture, you must clear the top 75 percent, click the Down Arrow, and clear the bottom 75 percent.

Show Picture — This shows the whole picture at one time. Click again to get back to the main menu.

Color Set — Toggles the color set between green/black and buff/black. Use buff to get artifact colors. Green is the default.

Put Modes — Used to change the PUT mode.

Undo has no menu, it simply undoes the last operation.

If It Doesn't Work

If parts of the main menu or pull-down menus are messed up, the problem is probably in the *MenuGen* program. Try proofreading the part of *MenuGen* corresponding to the menu where the problem is.

Any other problems are most likely in *CoCoDraw*. Look up the section that doesn't work in the line-by-line description and proofread those lines.

How it works

CoCoDraw uses all eight graphics pages. Pages one to four hold the actual picture, Page five is the main menu, and pages six to eight are the editing area. When *CoCoDraw* is run, it first initializes itself by defining the variables, loading the screen containing main menu and the pull-down menus (*MENUS.SYS*) and getting them into arrays. Then it PCOPYs from the picture to the editing area, puts the main menu on the screen, and goes to a subroutine starting at Line 9000 which allows you to move the pointer around using the joystick until you click the button.

Next, in lines 740 to 780 it checks to see if you were in the menu area when you clicked, and if so, it branches to the routine selected. That routine then takes control until another option is selected from the main menu. The program is very structured, so it should not be too difficult to follow. Here is a list of the major subroutines and how they are used:

8000 — Copies pages one to three to the editing area if PN=1 or pages two to four to the editing area if PN=2. Used in Undo and in most of the tools, such as in Line to erase the line you are making as you move it around. Also used to erase a pull-down menu and several other places as well.

8200 — Copies the editing area to pages one to three if PN=1 or to pages two to four if PN=2. Opposite of 8000.

8500 — Puts the main menu on the screen and fills in the Color and Background boxes.

8800 — This is the routine that allows you to select an option from a pull-down menu. Returns when you click the button. You must give it 'N' (the number of items in the menu minus one) and 'XX' (the 'X' value of where you want the menu to be). It gives you 'S' (the number of the item selected, with one being at the top).

9000 — Allows you to move the pointer around the screen using the joystick. Returns when you click the button. Gives you 'X' and 'Y', the screen location of the pointer when the button was clicked. It gets 'X' using JOYSTK(0)*4+XO. (XO is used to allow more detailed movement using the arrow keys on the keyboard.) Y=JOYSTK(1)*3YO.

"Actually, the editing area you see is just 75 percent of the entire picture. By clicking the Up Arrow you see the top 75 percent of the picture."

9200 — Checks the location of the joystick and the status of the button and returns immediately *only* if the position is in the editing area. If the position is in the main menu, it draws the pointer until you press the button or move into the editing area. If you press the button, it goes to Line 740 which checks what was selected and branches to it. If you move into the editing area, it returns.

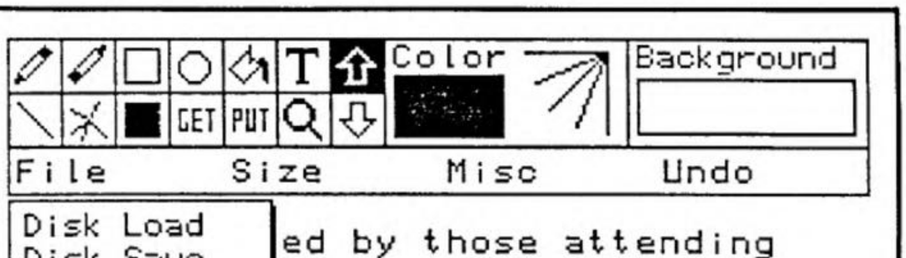
9400 — Checks the location of the joystick and the status of the button. Returns immediately. Gives you 'X' and 'Y' (the screen location of the joystick), and 'FB', which equals 254 (&HFE) if the button is pressed.

9500 — Used by 9400. Checks which arrow key is pressed and changes XO (X-offset) and YO (Y-offset) accordingly. Returns immediately.

9800 — Prints Yes and No on the screen starting at location 'X,Y' and lets you use the joystick to select one. Returns when button is pressed. If A<32 then the answer is yes.

10000 — Draws S$ starting at the current DRAW location, where S$ equals any string of text you want printed on the graphics screen. It can also use the variable EX$. EX$ equals any DRAW string you want inserted between each character. It is used in this program as BL1 to put less space between each character. Under normal use, EX$ equals the null string (" ").

11000 — This is the input routine



Disk Save
Filename: DEMO
O.K.? Yes No

Figure 3: File Menu

Tandy ELECTRONICS

Computer Packages For Office and Home!

Save $899

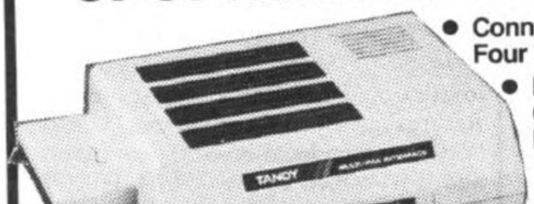
- Includes "DeskMate" Software
- PC-Compatible
- With Monitor Adapter

Reg **2598.00**

$1699

At last! A business computer you don't have to see your bank manager for. The Tandy 1000, complete with our new RGBI color monitor. The Tandy 1000 also includes adapter for adding joysticks, a monitor, light pen and printer. With 128K RAM, expandable to 640K, you've got a business and home education package that'll grow with you. The CM-5 Color Monitor has graphics that have to be seen to be believed. And only the Tandy 1000 has "DeskMate" applications software. It's a word processor, an electronic spreadsheet, a filer, a calendar and a communicator (with optional modem). It may be the only software you ever need. It'll definitely be part of the only computer you'll need. 25-1000, 25-1023

Co-Co Multi-Pak Interface

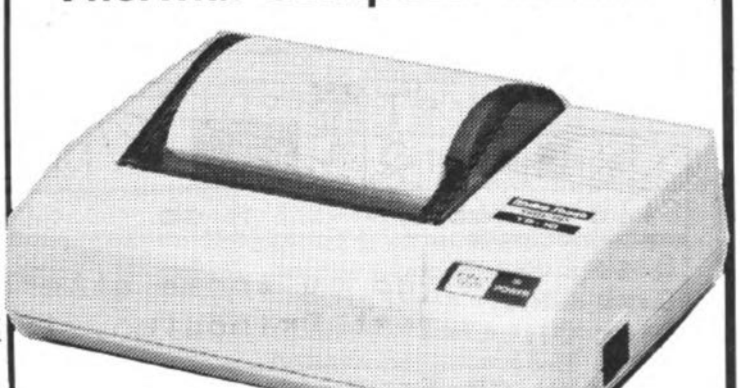


- Connects Up to Four Paks
- Increase Your Co-Co's Memory

179.95

Makes it much easier to change between program paks. Changes between slots with selector switch or under program control. A must for Color Computer owners. 26-3124

Thermal Computer Printer



Save $90

- Ideal for Small Spaces

Reg **189.95**

99.95

It's hard to believe! A thermal printer for less than $100. With features of printers three times the price. Print alphanumerics and block graphics with your Color Computer. Get a hard copy of your monthly budget, the kids homework, Mum's recipes and so much more. Prints 30 cps. Buy now while it's on special and SAVE! 26-1261

Computer Desk

Accessories not included

NEW!

- Suits Most Computer Hardware
- Tough Construction

199.95

Give your computer an established home. Finished in attractive woodgrain vinyl. Ready to assemble – no special tools needed. 90 x 93 x 60cm. 26-9408

The Amazing Color Computer 2

- **Attaches to Any TV Set for Instant Computing**
- **Create Color Graphics with Exciting Sound Effects and Music**

Save $70

Reg **299.95**

$229.95

One of the most popular home computers at a price that everyone can afford. Simply attach to your TV, plug in a Program Pak and you're away. Learn how easy it is to write programs, create graphics or make music. With 16K of memory, expandable to 64K. And you can add a tape recorder, joysticks, printer, modem and much more. Give your family something they'll use forever! 26-3136

TV not included

INTRODUCING The Color Computer 3

NEW!

- **Puts a Rainbow at Your Fingertips**

SPECIAL INTRODUCTORY OFFER!

Reg **449.95**

NOW

399.95

It's NEW! It's our most powerful Color Computer yet. And it's set to take the market by storm. The Tandy Color Computer 3 comes with 128K of memory (expandable to 512K), advanced programming facilities and high resolution graphics. In face, you can select 16 colors out of a pallet of 64 in detail as fine as 640 x 200-resolution. Expands easily with disk drives, printer, modem and more. Why pay more when you can get more for less! 26-3334

Tandy ELECTRONICS

A DIVISION OF TANDY AUSTRALIA LIMITED INC IN N.S.W.

Nearly 350 Stores Australia-Wide

WE SERVICE WHAT WE SELL!

Available From 350 Stores Australiawide Including Tandy Computer Centres

Order On VIATEL #642

Independent Tandy Dealers may not be participating in this ad or have every item advertised. Prices may also vary at individual Dealer Stores used in entering the filename for saving and loading and for adding text to your picture. You give it 'X' and 'Y', the screen location to start at; LE, the maximum length of the string to be input; and KE$, which limits which keys will be accepted. If you want all keys to be accepted, make KE$ equal to "ALL." If you want just 'Y' and 'N' allowed, make KE$ equal to "YyNn." It gives you IN$, the string that was input. Returns when ENTER is pressed.

See Table 1 for a line-by-line description and a variable list.

Modifications

To make this program work on a non-disk system:

A. Change Line 1000 to:

```
1000 GOSUB8200:PUT(0,48)-(79,1
  23),FI:XX=0:N=6:GOSUB8800:PO
  KE65494,0:IFS=7THEN1900ELSEIF
  S<4THEN700
```

B. Change Line 1100 to:

```
1100 CSAVEMFI$,1536,7679,44539
```

C. Delete Line 20

D. Take lines 40 to 520 from *MenuGen* and insert them into *CoCoDraw*. (You can do this because you have about 2K more memory than disk systems.)

E. To use, just CLOAD and run.

If you do not have a printer, add GOTO 700:REM to Line 1690. If you want to use another screen dump routine, delete lines 8, 1700, 1710 and 13000, and put your routine in lines 1700 to 1899. Remember to add a GOTO 700 at the end of the routine so the program will return to the main menu when the printing is done. Check back issues of THE RAINBOW for screen dump programs for other printers.

TABLE 1: Line Description of *CoCoDraw*

Initialization

1-3	Clear eight graphics pages; print title screen; (the apparently unnecessary GOTOs here are to get around the PCLEAR bug in older versions of Extended BASIC)
5	Dimensions and defines variables (see variables list)
8	Reads data for the screen dump routine
10	Sets graphics mode; clears screen
20	Loads screen containing the menus
600	Gets the menus into arrays

Check for Selection From Main Menu

700	Copies picture to editing area
710	Puts Main Menu; switches to graphics screen
720	Asks if you want to use the speed-up POKE
730-780	Read joystick; check which option was selected from main menu and branch to that routine

Pull-Down Menu Routines

1000-1900	File Menu routines (1700-1710 Screen dump routine)
2000-2010	Size Menu routine
2500-2560	Misc Menu routine
3000	Undo routine

Color and Background

3500	Draws color selection screen
3510-3530	Select color/pattern using joystick

Tools

4000	Draw
4200	Erase
4400	Box and Box Fill
4600	Circle
4800	Paint
5000	Text
5200	Up Arrow
5400	Line
5600	Ray
5800	GET
6000	PUT
6200	Magnify
6400	Down Arrow

Subroutines

8000	Copies from picture to editing area
8200	Copies from editing area to picture
8500	Puts main menu and fills in colors in main menu
8800	Selects item from pull-down menus
9000	Draws pointer and waits for you to press the button
9200	Like 9400 unless you're in the main menu (see article)
9400	Reads joystick and returns
9500	Checks which arrow key was pressed
9800	Selects Yes or No using the joystick
10000	Prints a text string on the graphics screen
11000	Inputs a text string from the keyboard

Data for Character Set (in ASCII Order)

12000	'space' to '-'
12010	'.' to '9'
12020	':' to '@'
12030	'A' to 'M'
12040	'N' to 'Z'
12050	ASCII 91 to ASCII 96
12060	'a' to 'k'
12070	'l' to 'w'
12080	'x' to 'z'
13000	Data for screen dump routine

Variables

Numeric

CS	= Color set
FC	= Foreground color
BC	= Background color
ES	= Eraser size 1
PM	= PUT Mode
P	= Used to copy pages, POKE value, etc.
LE	= Max length of input
PN	= Page number
FB	= Firebutton status
X,Y,XX,YY,X1,Y1	=scrn. loc.
S	= Menu selection
A	= Misc. variable
N	= Number of options in pull-down menu 1

String

S$	= String to be printed in print routine (10000)
SP$	= 'Y' if speed-up poke is allowed and 'N' if not
EX$	= Used as an extension to print routine (see article)
I$	= Inkey$
IN$	= String that was typed in input routine (11000)
FI$	= Filename

Arrays

ME	= Main Menu (GP) (GP = GET/PUT array)
FI	= File Menu (GP)
SI	= Size Menu (GP)
MI	= Misc Menu (GP)
GP	= GET and PUT (GP)
AR	= Pointer (GP)
A	= Misc. Array (GP) (Used mainly in joystick input routine)
CH$	= Draws strings for character set
D	= Used in screen dump routine

130	236
400	130
10000	189
12030	226
END	87

730	33	5200	97	9520	8
1030	159	6000	150	11020	128
1900	79	6240	137	12010	86
3000	224	6290	246	12050	170
4230	116	8830	119	END	181

Listing 1: MENUGEN

```
0 ' MENUGEN, BY DARIN HERR
1 ' FOR USE WITH COCODRAW V1.1
2 GOTO 8
5 PCLEAR8:DIMCH$(90):FORA=0TO90:
READCH$(A):NEXT:GOTO10
8 PCLEAR8:GOTO5
10 PMODE4,5:COLOR0,1:PCLS:SCREEN
1,0
40 ' ** THE POINTER **
50 DRAW"BM0,124R3G3U3F4"
90 ' ** THE MAIN MENU **
100 LINE(0,0)-(255,46),PSET,B:FO
RX=0TO112STEP16:LINE(X,0)-(X,32)
,PSET:NEXT:LINE(115,11)-(147,29)
,PSET,B:LINE(187,12)-(251,28),PS
ET,B:LINE(184,0)-(184,32),PSET:L
INE(0,16)-(112,16),PSET:LINE(0,3
2)-(255,32),PSET
110 DRAW"BM11,2G8D3LUR2UDRE8HL2D
R2DL2;BM20,13E8U3RDL2DULG8FR2UL2
UR2;BM35,3R10D10L10U10;BM54,4R3F
3D3G3L3H3U3E3;BM71,4G5F4E5RD4FU6
LULDLH2D2U5;BM83,5U2R9D2HL3D8FL3
EU8L4;BM104,3G5R3D5R4U5R3H5"
120 DRAW"BM3,19F10;BM24,24;M27,1
9;BM24,24;F5H5;M22,30;BM24,24;M1
9,28;BM24,24;M19,21;BM53,21L2D6R
2U2BU4BR4L2D3R2L2D3R2BR3U6LR2;BM
67,24R2U3L2D6BR4BU6D6R2U6BR2R2LD
6;BM85,18R3F3D3GDF3DH4GL3H3U3E2R
4F2D3G2L3H2U3E1;BM102,19D5L3F5E5
L3U5L3"
130 LINE(35,19)-(45,29),PSET,BF
140 DRAW"BM3,36":S$="File":GOSUB
10000:DRAW"BM67,36":S$="Size":GO
SUB10000:DRAW"BM131,36":S$="Misc
":GOSUB10000:DRAW"BM195,36":S$="
Undo":GOSUB10000
190 ' ** THE COLOR BOXES **
200 DRAW"BM115,2":S$="Color":GOS
UB10000:DRAW"BM187,2":EX$="BL1":
S$="Background":GOSUB10000
290 ' ** THE FILE MENU **
300 LINE(0,48)-(78,122),PSET,B:L
INE(2,123)-(79,123),PSET:LINE-(7
9,50),PSET
310 EX$="BL1":DRAW"BM5,52":S$="D
isk Load":GOSUB10000:DRAW"BM5,62
":S$="Disk Save":GOSUB10000:DRAW
"BM5,72":S$="Disk Dir":GOSUB1000
0:DRAW"BM5,82":S$="Tape Load":GO
SUB10000:DRAW"BM5,92":S$="Tape S
ave":GOSUB10000
320 DRAW"BM5,102":S$="Screen Dum
p":GOSUB10000:DRAW"BM5,112":EX$=
"BL1":S$="Quit/BASIC":GOSUB10000
390 ' ** THE SIZE MENU **
400 LINE(80,48)-(158,92),PSET,B:
LINE(82,93)-(159,93),PSET:LINE-(
159,50),PSET
410 DRAW"BM85,52":S$="Eraser Siz
e":GOSUB10000:DRAW"BM85,62":S$="
  1 x 1":GOSUB10000:DRAW"BM85,72
":S$="  4 x 4":GOSUB10000:DRAW"B
M85,82":S$="> 8 x 8":GOSUB10000
490 ' ** THE MISC MENU **
500 LINE(160,48)-(238,142),PSET,
B:LINE(162,143)-(239,143),PSET:L
INE-(239,50),PSET
510 DRAW"BM165,52":S$="Clear Scr
een":GOSUB10000:DRAW"BM165,62":S
$="Show Picture":GOSUB10000:DRAW
"BM165,72":S$="Color Set":GOSUB1
0000:DRAW"BM165,82":S$="PUT Mode
":GOSUB10000:DRAW"BM165,92":S$="
> Set":GOSUB10000:DRAW"BM165,102
":S$="  Reset":GOSUB10000
520 DRAW"BM165,112":S$="  And":G
OSUB10000:DRAW"BM165,122":S$="
Or":GOSUB10000:DRAW"BM165,132":S
$="  Not":GOSUB10000:EX$=""
600 EXEC44539:INPUT"PRESS [ENTER
] WHEN READY TO SAVE";A$:SAVEM"M
ENUS.SYS",9728,14333,44539
610 END
9990 ' ** HI-RES PRINT ROUTINE *
*
10000 FORA=1TOLEN(S$):DRAWCH$(AS
C(MID$(S$,A,1))-32)+EX$:NEXT:RET
URN
11990 ' ** CHARACTER DATA **
12000 DATA BR7,BR2D4BD2D0BU6BR5,
BRDBR2UBR4,BD2R4HD4EL4FU4BUBR6,B
R4BDL4D2R4D2L4R2DU6BR5,DRUBR3DG4
DBR3URDBU6BR3,BRRFG3DFRE2BD2H4UB
UBR7,BRDRUBR5,BR3G2D2F2BU6BR4,BR
F2D2G2BU6BR6,BD3R4BD2H4BD4E4BUBR
3,BD3R4BG2U4BUBR5,BD6BR2GBU7BR6,
BD3R4BU3BR3
12010 DATA BD6BR2R0BU6BR5,BD6UE4
UBR3,BDD4FR2EU4HL2BD3BRR0BE3BR2,
BR2D6RL2BU5EBR5,BDER2FDG4R4BU6BR
3,BDER2FDGL2R2FDGL2HBE5BR2,D3R4L
D3U6BR4,R4L4D3R4D3L4BE6BR,BDD4FR
2EUHL2BU3R2FBEBR2,DUR4D2G3DBE6,B
DDFR2FDGL2HUER2EUHL2BR6,BRR2FD4G
L2HBU4DFR3BE3
12020 DATA BD3BR2D0BD3U0BU6BR5,B
D3BR2D0BD3GBU7BR6,BR3G3F3BU6BR4,
BD2R4BD2L4BE4BR3,BRF3G3BE6,BD2UE
R2FD2L2DBD2U0BU6BR5,R4D4L2U2R2BD
4L4U6BR7
12030 DATA BDD5U2R4D2U5HL2BR6,D6
R3EUHL2R2EUHL2BR6,BDD4FR2EBU4HL2
BR6,D6R2E2U2H2LBR6,D3R3L3D3R4BU6
L4BR7,D6U3R3L3U3R4BR3,BDD4FR2EU2
L2R2BU2HL2BR6,D6U3R4D3U6BR3,R4L2
D6L2R4BU6BR3,BD4DFR2EU5BR3,D6U3R
F3H3E3BR3,D6R4BU6BR3,D6U5RFDUERD
5U6BR3
12040 DATA D6U5RFD2F2U6BR3,BDD4F
R2EU4HL2BR6,D6U3R3EUHL2BR6,BDD4F
REHF2HEU3HL2BR6,D6U4F4H3R2EUHL2B
R6,BDDFR2FDGL2HBE4HL2BR6,R2D6U6R
2BR3,D6R4U6BR3,D3FDFEUEU3BR3,D5F
EUDFEU5BR3,DF4DBL4UE4UBR3,DFDFD2
U2EUEUBR3,R4DG4DR4BU6BR3
12050 DATA BRR2L2D6R2BU6BR4,BD8L
R6BU8BR2,BRR2D6L2BE6,BD2E2D6U6F2
BU2BR3,BL7,BR7
12060 DATA BD2R3FD3L3HER3BU4BR3,
D6R3EU2HL2BU2BR6,BD3D2FR2EBU2HL2
BU2BR6,BD3D2FR3U4L3R3U2BR3,BD3DR
4UHL2GD2FR3BU6BR3,BD3R3L2D3U5ERF
BEBR2,BD3D2FR3DGL3BR4BUU5L3R3BU2
BR3,D6U4R3FD3BU6BR3,BDBR2D0BD2D3
BU6BR5,BD7FR2EU4BU2U0BUBR3,D6U3F
3H2E2BU2BR4
12070 DATA BR2D6RBU6BR4,BD2D4U4R
2D4U4RFD3BU6BR3,BD2D4U4R3FD3BU6B
R3,BD3D2FR2EU2HL2BU2BR6,BD2D6U2R
3EU2HL2BU2BR6,BD3D2FR3D2U6L3BU2B
R6,BD2D4U2E2R2BU2BR3,BD3FR2FGL3B
E4L3BU2BR6,BD2R2LU2D5FEBU5BR4,BD
2D3FR2EU3BU2BR3,BD2DFDFEUEUBU2BR
3,BD2D3FEUDFEU3BU2BR3
12080 DATA BD2F4H2G2E4BU2BR3,BD2
DFDFG2E3UEUBU2BR3,BD2R4G4R4BU6BR
3
```

Listing 2: COCODRAW

```
0 'COCODRAW V1.1 BY DARIN HERR
1 GOTO3
2 CLS:PRINT@108,"COCODRAW":PRINT
@142,"V1.1":PRINT@207,"BY":PRINT
@235,"DARIN HERR":PRINT@270,"198
6":PRINT@449,"INITIALIZING-ONE M
OMENT PLEASE":GOTO5
3 PCLEAR8:GOTO2
5 DIM A(102),AR(1),ME(307),FI(15
1),SI(92),MI(191),GP(102),CH$(90
):FORA=0TO90:READCH$(A):NEXT:BC=
3:ES=7:PN=1:PM=1:LE=40
8 DIMD(15):FORA=0TO15:READD(A):N
EXT
10 PMODE4,5:COLOR0,1:PCLS
20 LOADM"MENUS.SYS"
600 GET(0,124)-(4,128),AR,G:GET(
0,0)-(255,47),ME:GET(0,48)-(79,1
23),FI:GET(80,48)-(159,93),SI:GE
T(160,48)-(239,143),MI
700 GOSUB8000
710 GOSUB8500:SCREEN1,CS
720 IFSP$=""THENLINE(56,84)-(197
,117),PRESET,BF:LINE(57,85)-(196
,116),PSET,B:DRAW"BM64,88":S$="D
o you want to use":GOSUB10000:DR
AW"BM64,97":S$="the Speed-Up POK
E?":GOSUB10000:X=101:Y=106:GOSUB
9800:IFA<32THENSP$="Y":GOTO1110E
LSESP$="N":GOSUB8000
730 GOSUB 9000
740 PLAY"O3T16C":IFY>47THEN730
750 IFY>32THENS=INT(X/64)+1:ONS
GOTO1000,2000,2500,3000
760 IFX>111THENS=X:GOTO3500
770 X=INT(X/16)*16:Y=INT(Y/16)*1
6:S=(X/16+1)+7*Y/16:IFS=7ORS=14T
HEN780ELSEPUT(X+1,Y+1)-(X+15,Y+1
5),A,NOT
780 ONS GOTO4000,4200,4400,4600,
4800,5000,5200,5400,5600,4400,58
00,6000,6200,6400
1000 GOSUB8200:PUT(0,48)-(79,123
),FI:XX=0:N=6:GOSUB8800:POKE6549
4,0:IFS=7THEN1900ELSEIFS=3THENGO
SUB8000:CLS:INPUT"DRIVE #";P:IFP
>3THEN700ELSEDIRP:FORA=1TO2STEP0
:IF(PEEK(&HFF00)OR&H80)<>&HFE TH
ENNEXTELSE700
1010 POKE178,0:POKE179,3:LINE(36
,84)-(217,117),PRESET,BF:LINE(37
,85)-(216,116),PSET,B:IFS=6THEN1
690ELSEIFS<3THENS$="Disk"ELSES$=
"Tape"
1020 IFS=1ORS=4THENS$=S$+" Load"
ELSES$=S$+" Save"
1030 DRAW"BM40,88":GOSUB10000:DR
AW"BM40,97":S$="Filename:":GOSUB
10000:IFS<3THENLE=14ELSELE=8
1040 X=110:Y=97:KE$="ALL":GOSUB1
1000:FI$=IN$
1050 DRAW"BM40,106":S$="O.K.?":G
OSUB10000:X=82:Y=106:GOSUB9800:I
FA>31THEN1110
1060 IFS>3THEN1090
1070 IFS=1THENLOADMFI$ELSESAVEMF
I$,3584,9727,44539
1080 GOTO1110
1090 IF S=4 THEN PMODE4,1:SCREEN
1,CS:CLOADMFI$:PMODE4,5:SCREEN1,
CS:GOTO1110
1100 CSAVEMFI$,3584,9727,44539
1110 GOSUB8000:IFSP$="Y"THENPOKE
65495,0
```

```
1120 GOTO730
1690 S$="Screen Dump":DRAW"BM87,
88":GOSUB10000:S$="Is printer re
ady?":DRAW"BM67,97":GOSUB10000:X
=99:Y=106:GOSUB9800:IFA>31THEN11
10
1700 PRINT#-2,CHR$(27)"@"CHR$(27
)"A"CHR$(8):S$=CHR$(27)+"K"+CHR$
(128)+CHR$(1):FORA=1536TO1567:FO
RX=1TO2:PRINT#-2,S$;:FORB=191TO0
STEP-1:P=NOT(PEEK(A+32*B)):IFX=1
THENC=(P AND240)/16ELSEC=P AND15
1710 PRINT#-2,CHR$(D(C));CHR$(D(
C));:NEXTB:PRINT#-2,CHR$(0):NEXT
X,A:GOTO700
1900 GOSUB8000:CLS:PRINT"TYPE 'C
ONT [ENTER]' TO RESTART  PROGRAM
.":PRINT:STOP:GOTO700
2000 GOSUB8200:PUT(64,48)-(143,9
3),SI:XX=64:N=3:GOSUB8800:IFS=1T
HENGOSUB8000:GOTO730
2010 POKE178,0:POKE179,3:LINE(69
,62)-(74,89),PRESET,BF:DRAW"BM69
,"+STR$(S*10+42)+CH$(30):GET(64,
48)-(143,93),SI:S=S-2:ES=S*4-SGN
(S):GOTO700
2500 GOSUB8200:PUT(128,48)-(207,
143),MI:XX=128:N=8:GOSUB8800:IFS
<5THENONS GOTO2510,2520,2540,255
0ELSE2560
2510 GOSUB8000:LINE(0,48)-(255,1
91),PRESET,BF:GOTO730
2520 GOSUB8000:PMODE4,1:SCREEN1,
CS
2530 GOSUB9400:IFFB<>&HFE THEN25
30ELSEPMODE4,5:GOTO700
2540 CS=ABS(CS-1):SCREEN1,CS:GOS
UB8000:GOTO730
2550 GOSUB8000:GOTO730
2560 POKE178,0:POKE179,3:LINE(13
3,92)-(138,139),PRESET,BF:DRAW"B
M133,"+STR$(S*10+42)+CH$(30):PM=
S-4:GET(128,48)-(207,143),MI:GOS
UB8000:GOTO700
3000 GOSUB8000:GOTO730
3500 GOSUB8200:PCLS:FORY=3TO191S
TEP12:FORX=3TO256STEP16:A=(Y-3)/
12*16+(X-3)/16:POKE178,A:LINE(X,
Y)-(X+7,Y+6),PSET,BF:IFINKEY$<>"
"THEN700ELSENEXTX,Y:IFBC=0THENPO
KE178,3ELSEPOKE178,0
3510 X=INT(JOYSTK(0)/4)*16+2:Y=I
NT(JOYSTK(1)/4)*12+2:LINE(X,Y)-(
X+9,Y+8),PSET,B:LINE(X,Y)-(X+9,Y
+8),PRESET,B:FB=PEEK(&HFF00)OR&H
80:I$=INKEY$:IFFB<>&HFE ANDI$=""
THEN3510
3520 A=(Y-2)/12*16+(X-2)/16:IFI$
<>""THEN700ELSEIFS<184THENFC=A E
LSEBC=A
3530 GOTO700
4000 GOSUB9000:IFY<48THENGOSUB85
00:GOTO740ELSEGOSUB8200:LINE(X,Y
)-(X,Y),PRESET
4010 GOSUB9400:LINE-(X,Y),PSET:I
FFB=&HFE THEN4010ELSE4000
4200 GOSUB 8200:IFBC=0THENPOKE17
8,3ELSEPOKE178,0
4210 GOSUB9200:IFX>255-ES THENX=
255-ES
4220 IFY>191-ES THENY=191-ES
4230 GET(X,Y)-(X+ES,Y+ES),A,G:LI
NE(X,Y)-(X+ES,Y+ES),PSET,BF:LINE
(X,Y)-(X+ES,Y+ES),PRESET,BF:IFFB
<>&HFE THENPUT(X,Y)-(X+ES,Y+ES),
A,PSET
4240 GOTO4210
4400 GOSUB9000:XX=X:YY=Y:IFY<48T
HENGOSUB8500:GOTO740ELSEGOSUB820
0
4410 GOSUB9400:LINE(XX,YY)-(X,Y)
,PRESET,B:LINE-(XX,YY),PSET,B:IF
FB=&HFE THENGOSUB8000:GOTO4410EL
SEIFS>7THENLINE(XX,YY)-(X,Y),PSE
T,BF4420 GOTO4400
4600 GOSUB9000:XX=X:YY=Y:IFY<48T
HENGOSUB8500:GOTO740ELSEGOSUB820
0
4610 GOSUB9400:CIRCLE(XX,YY),ABS
(X+Y/12-8):IFFB=&HFE THENGOSUB80
00:GOTO4610ELSE4600
4800 GOSUB9000:IFY<48THENGOSUB85
00:GOTO740ELSEIFPPOINT(X,Y)=0THE
NC=1ELSEC=0
4810 GOSUB8200:PAINT(X,Y),,C:GOT
O4800
5000 KE$="ALL":LE=36
5010 GOSUB9000:IFY<48THENPUT(81,
1)-(95,15),A,NOT:GOTO740ELSEIFX=
0THENX=1
5020 GOSUB8200:GOSUB11000:GOTO50
10
5200 IFPN=1THEN730ELSEPUT(97,1)-
(111,15),A,NOT:PUT(97,17)-(111,3
1),A,NOT:GOSUB8200:PN=1:GOSUB800
0:GOTO730
5400 GOSUB9000:IFY<48THENGOSUB85
00:GOTO740ELSEGOSUB8200:XX=X:YY=
Y
5410 GOSUB8000:GOSUB9400:LINE(XX
,YY)-(X,Y),PSET:IFFB=&HFE THEN54
10ELSE5400
5600 GOSUB9000:IFY<48THENGOSUB85
00:GOTO740ELSEXX=X:YY=Y:GOSUB820
0
5610 GOSUB9400:LINE(XX,YY)-(X,Y)
,PSET:IFFB=&HFE THENA=0:GOTO5620
ELSEGOSUB8000:GOTO5610
5620 A=A+1:GOSUB9400:IFFB=&HFE A
NDA<10THEN5620ELSEIFA<10THENGOSU
B8200:GOTO5610ELSE5630
5630 PLAY"C":GOSUB9400:IFFB=&HFE
 THEN5630ELSE5600
5800 GOSUB9000:IFY<48THENPUT(49,
17)-(63,31),A,NOT:GOTO740ELSEGOS
UB8200:XX=X:YY=Y
5810 X=JOYSTK(0):Y=JOYSTK(1):IFX
X+X>255THENX=255-XX
5820 IFYY+Y>191THENY=191-YY
5830 LINE(XX,YY)-(XX+X,YY+Y),PRE
SET,B:LINE-(XX,YY),PSET,B:FB=PEE
K(&HFF00)OR&H80:GOSUB8000:IFFB=&
HFE THEN5810ELSEGET(XX,YY)-(XX+X
,YY+Y),GP,G:FORA=1TO4:PUT(XX,YY)
-(XX+X,YY+Y),GP,NOT:NEXT:PUT(49,
17)-(63,31),A,NOT:GX=X:GY=Y:GOTO
730
6000 GOSUB8200
6010 GOSUB9200:IFX+GX>255THENX=2
55-GX
6020 GET(X,Y)-(X+GX,Y+GY),A,G:ON
PM GOSUB6040,6050,6060,6070,6080
:IFFB<>&HFE THENPUT(X,Y)-(X+GX,Y
+GY),A,PSET
6030 GOTO6010
6040 PUT(X,Y)-(X+GX,Y+GY),GP,PSE
T:RETURN
6050 PUT(X,Y)-(X+GX,Y+GY),GP,PRE
SET:RETURN
6060 PUT(X,Y)-(X+GX,Y+GY),GP,AND
:RETURN
6070 PUT(X,Y)-(X+GX,Y+GY),GP,OR:
RETURN
6080 PUT(X,Y)-(X+GX,Y+GY),GP,NOT
:RETURN
6200 POKE178,0:POKE179,3
6210 GOSUB9200:IFX>240THENX=240
6220 IFY>176THENY=176
6230 GET(X,Y)-(X+15,Y+15),A,G:LI
NE(X,Y)-(X+15,Y+15),PRESET,B:LIN
E(X,Y)-(X+15,Y+15),PSET,B:PUT(X,
Y)-(X+15,Y+15),A,PSET:IFFB<>&HFE
 THEN6210ELSEGOSUB8200:X1=X:Y1=Y
6240 SCREEN,0:PCLS:LINE(12,12)-(
146,146),PSET,B:LINE(11,11)-(147
,147),PSET,B:LINE(16,160)-(72,17
6),PSET,B:LINE(88,160)-(144,176)
,PSET,B:LINE(190,30)-(209,49),PS
ET,B:LINE(190,94)-(209,113),PSET
,B:LINE(0,0)-(255,191),PSET,B:LI
NE(1,1)-(254,190),PSET,B
6250 DRAW"BM57,3":S$="MAGNIFY":G
OSUB10000:DRAW"BM31,165":S$="Don
e":GOSUB10000:DRAW"BM96,165":S$=
"Cancel":GOSUB10000:DRAW"BM181,5
6":S$="Before":GOSUB10000:DRAW"B
M191,120":S$="Now":GOSUB10000
6260 PUT(192,32)-(207,47),A,PSET
:PUT(192,96)-(207,111),A,PSET:FO
RY=32TO47:YY=(Y-32)*8+16:FORX=19
2TO207:XX=(X-192)*8+16:IFPPOINT(
X,Y)=0THENLINE(XX,YY)-(XX+6,YY+6
),PSET,BF ELSELINE(XX,YY)-(XX+6,
YY+6),PRESET,BF
6270 NEXTX,Y
6280 GOSUB9000:IFX>142ORX<16ORY<
16ORY>176THEN6280ELSEIFY<144THEN
X=INT((X-16)/8):Y=INT((Y-16)/8):
XX=X*8+16:YY=Y*8+16:PUT(XX,YY)-(
XX+6,YY+6),A,NOT:PUT(X+192,Y+96)
-(X+192,Y+96),A,NOT:GOTO6280
6290 IFY<160ORY>176OR(X>93ANDX<8
8)THEN6280ELSEIFX>87THENGET(192,
32)-(207,47),A,G:PUT(192,96)-(20
7,111),A,PSET:GOTO6260ELSEGET(19
2,96)-(207,111),A,G:GOSUB8500:GO
SUB8000:PUT(X1,Y1)-(X1+15,Y1+15)
,A,PSET:X=80:Y=16:GOTO770
6400 IFPN=2THEN730ELSEPUT(97,1)-
(111,15),A,NOT:PUT(97,17)-(111,3
1),A,NOT:GOSUB8200:PN=2:GOSUB800
0:GOTO730
8000 FORP=PN TOPN+2:PCOPYP TOP+6
-PN:NEXT:RETURN
8200 FORP=6TO8:PCOPYP TOP-6+PN:N
EXT:RETURN
8500 PUT(0,0)-(255,47),ME:POKE17
8,FC:POKE179,BC:LINE(116,12)-(14
6,28),PSET,BF:DRAW"BM178,4NL24NG
17ND24M157,13M178,4M169,26":LINE
(188,13)-(250,27),PRESET,BF:IFPN
=1THENPUT(97,1)-(111,15),A,NOT:R
ETURNELSEPUT(97,17)-(111,31),A,N
OT:RETURN
8800 GOSUB9400:S=INT(Y/3/(63/N))
:GOTO8830
8810 GOSUB9400:S=INT(Y/3/(63/N))
:IFFB<>&HFE THENS=S+1:PUT(XX+2,P
*10+51)-(XX+76,P*10+60),A,NOT:RE
TURNELSEIFS=P THEN8810
8820 PUT(XX+2,P*10+51)-(XX+76,P*
10+60),A,NOT
8830 PUT(XX+2,S*10+51)-(XX+76,S*
10+60),A,NOT:P=S:GOTO8810
9000 GOSUB9400:GOTO9020
9010 GOSUB9400:PUT(XX,YY)-(XX+4,
YY+4),A,PSET:IFFB=&HFE THENRETUR
N
9020 GET(X,Y)-(X+4,Y+4),A,G:IFPP
OINT(X,Y)>0THENPUT(X,Y)-(X+4,Y+4
),AR,PSET ELSEPUT(X,Y)-(X+4,Y+4)
,AR,PRESET
9030 XX=X:YY=Y:GOTO9010
9200 GOSUB9400:IFY>47THENRETURNE
LSE9220
9210 GOSUB9400:PUT(XX,YY)-(XX+4,
YY+4),A,PSET:IFY>47THENRETURN
9220 GET(X,Y)-(X+4,Y+4),A,G:IFPP
OINT(X,Y)>0THENPUT(X,Y)-(X+4,Y+4
),AR,PSET ELSE PUT(X,Y)-(X+4,Y+4
),AR,PRESET
9230 IFFB=&HFE THENPUT(X,Y)-(X+4
,Y+4),A,PSET:GOSUB8500:GOTO740EL
SEXX=X:YY=Y:GOTO9210
9400 IFPEEK(&H155)+PEEK(&H156)+P
EEK(&H157)+PEEK(&H158)<&H3F8 THE
NGOSUB9500
9410 X=JOYSTK(0)*4+XO:Y=JOYSTK(1
)*3+YO:FB=PEEK(&HFF00)OR&H80:RET
URN
9500 IF(PEEK(341)=247ORPEEK(341)
=246)ANDYO>0THENYO=YO-1
9510 IF(PEEK(342)=247ORPEEK(342)
=246)ANDYO<2THENYO=YO+1
9520 IF(PEEK(343)=247ORPEEK(343)
=246)ANDXO>0THENXO=XO-1
9530 IF(PEEK(344)=247ORPEEK(344)
=246)ANDXO<3THENXO=XO+1
```

```
9540 FB=PEEK(&HFF00)OR&H80:IFFB=
&HFE THENP=254ELSEP=255
9550 FORA=&H155 TO&H158:POKEA,P:
NEXT:RETURN
9800 DRAW"BM"+STR$(X)+","+STR$(Y
):S$="Yes    No":GOSUB10000
9810 A=JOYSTK(0):IFA<32THENPOKE1
78,0:POKE179,3ELSEPOKE178,3:POKE
179,0
9820 LINE(X-1,Y+8)-(X+20,Y+8),PS
ET:LINE(X+40,Y+8)-(X+55,Y+8),PRE
SET:IF(PEEK(&HFF00)OR&H80)=&HFE
THENPOKE178,FC:POKE179,BC:RETURN
ELSE9810
10000 FORA=1TOLEN(S$):DRAWCH$(AS
C(MID$(S$,A,1))-32)+EX$:NEXT:RET
URN
11000 DRAW"BM"+STR$(X)+","+STR$(
Y)+"LD8RU8RD8RU8RD8RU8RD8U8L5":G
ET(X-1,Y)-(X+5,Y+8),A,G:IN$=INKE
Y$
11010 IN$=INKEY$:IFIN$=""ANDCO<1
0THENCO=CO+1:GOTO11010ELSEIFIN$=
""THENPUT(X-1,Y)-(X+5,Y+8),A,NOT
:CO=0:GOTO11010
11020 IFIN$=CHR$(8)THENIFLEN(DE$
)=0THEN11010ELSEPUT(X-1,Y)-(X+5,
Y+8),A,PRESET:X=X-7:DE$=LEFT$(DE
$,LEN(DE$)-1):DRAW"BL7":PUT(X-1,
Y)-(X+5,Y+8),A,PSET:GOTO11010
11030 IFIN$=CHR$(13)THENPUT(X-1,
Y)-(X+5,Y+8),A,PRESET:IN$=DE$:DE
$="":RETURN
11040 IFASC(IN$)<31ORLEN(DE$)=LE
 ORIN$=CHR$(95)ORX>242 THEN11010
11050 IFKE$<>"ALL"THENIFINSTR(KE
$,IN$)=0THENSOUND1,1:GOTO11010
11060 PUT(X-1,Y)-(X+5,Y+8),A,PRE
SET:DRAWCH$(ASC(IN$)-32):DE$=DE$
+IN$:X=X+7:PUT(X-1,Y)-(X+5,Y+8),
A,PSET:GOTO11010
12000 DATA BR7,BR2D4BD2D0BU6BR5,
BRDBR2UBR4,BD2R4HD4EL4FU4BUBR6,B
R4BDL4D2R4D2L4R2DU6BR5,DRUBR3DG4
DBR3URDBU6BR3,BRRFG3DFRE2BD2H4UB
UBR7,BRDRUBR5,BR3G2D2F2BU6BR4,BR
F2D2G2BU6BR6,BD3R4BD2H4BD4E4BUBR
3,BD3R4BG2U4BUBR5,BD6BR2GBU7BR6,
BD3R4BU3BR3
12010 DATA BD6BR2R0BU6BR5,BD6UE4
UBR3,BDD4FR2EU4HL2BD3BRR0BE3BR2,
BR2D6RL2BU5EBR5,BDER2FDG4R4BU6BR
3,BDER2FDGL2R2FDGL2HBE5BR2,D3R4L
D3U6BR4,R4L4D3R4D3L4BE6BR,BDD4FR
2EUHL2BU3R2FBEBR2,DUR4D2G3DBE6,B
DDFR2FDGL2HUER2EUHL2BR6,BRR2FD4G
L2HBU4DFR3BE3
12020 DATA BD3BR2D0BD3U0BU6BR5,B
L3BR2D0BD3GBU7BR6,BR3G3F3BU6BR4,
BD2R4BD2L4BE4BR3,BRF3G3BE6,BD2UE
R2FD2L2DBD2U0BU6BR5,R4D4L2U2R2BD
4L4U6BR7
12030 DATA BDD5U2R4D2U5HL2BR6,D6
R3EUHL2R2EUHL2BR6,BDD4FR2EBU4HL2
BR6,D6R2E2U2H2LBR6,D3R3L3D3R4BU6
L4BR7,D6U3R3L3U3R4BR3,BDD4FR2EU2
L2R2BU2HL2BR6,D6U3R4D3U6BR3,R4L2
D6L2R4BU6BR3,BD4DFR2EU5BR3,D6U3R
F3H3E3BR3,D6R4BU6BR3,D6U5RFDUERD
5U6BR3
12040 DATA D6U5RFD2F2U6BR3,BDD4F
R2EU4HL2BR6,D6U3R3EUHL2BR6,BDD4F
REHF2HEU3HL2BR6,D6U4F4H3R2EUHL2B
R6,BDDFR2FDGL2HBE4HL2BR6,R2D6U6R
2BR3,D6R4U6BR3,D3FDFEUEU3BR3,D5F
EUDFEU5BR3,DF4DBL4UE4UBR3,DFDFD2
U2EUEUBR3,R4DG4DR4BU6BR3
12050 DATA BRR2L2D6R2BU6BR4,BD8L
R6BU8BR2,BRR2D6L2BE6,BD2E2D6U6F2
BU2BR3,BL7,BR7
12060 DATA BD2R3FD3L3HER3BU4BR3,
D6R3EU2HL2BU2BR6,BD3D2FR2EBU2HL2
BU2BR6,BD3D2FR3U4L3R3U2BR3,BD3DR
4UHL2GD2FR3BU6BR3,BD3R3L2D3U5ERF
BEBR2,BD3D2FR3DGL3BR4BUU5L3R3BU2
BR3,D6U4R3FD3BU6BR3,BDBR2D0BD2D3
BU6BR5,BD7FR2EU4BU2U0BUBR3,D6U3F
3H2E2BU2BR4
12070 DATA BR2D6RBU6BR4,BD2D4U4R
2D4U4RFD3BU6BR3,BD2D4U4R3FD3BU6B
R3,BD3D2FR2EU2HL2BU2BR6,BD2D6U2R
3EU2HL2BU2BR6,BD3D2FR3D2U6L3BU2B
R6,BD2D4U2E2R2BU2BR3,BD3FR2FGL3B
E4L3BU2BR6,BD2R2LU2D5FEBU5BR4,BD
2D3FR2EU3BU2BR3,BD2DFDFEUEUBU2BR
3,BD2D3FEUDFEU3BU2BR3
12080 DATA BD2F4H2G2E4BU2BR3,BD2
DFDFG2E3UEUBU2BR3,BD2R4G4R4BU6BR
3
13000 DATA0,3,12,15,48,51,60,63,
192,195,204,207,240,243,252,255
```

Function in Basic

continued from Page 24

You may wonder, what value is all this nonsense to me? It is important for a newcomer to become familiar with all the functions, statements and so forth. Knowing all the nuances of CoCo's features allows you a broad option of possibilities when composing a program. The more ways you know to do a job, the more alternatives are available for your creations.

The educational language program you learned how to create and use recently would have been impossible to create without knowing what CoCo could do with LEFT$, MID$ and RIGHT$. Knowing what is possible affords you the choice of following many pathways to a fruitful conclusion.

In fact, in fooling around with these tutorials, ideas began to perk in my noodle and aided me to move from dead center to further enlarge and modify that program.

Notice how in lines 320 and 330 we continue to employ the invisible vertical line gambit to format our text. The point to be emphasized is that what you learn in these tutorials becomes a part of your computing skills and can hereinafter be called forth on demand to create some goodie that is near and dear to your heart. □

Listing 1: STRINGS1

```
0 '<LISTING1>
10 CLS
20 A$="INVERNESS, FL,32650JOSEPH
 KOLAR1709 DICKINSON STREET"
30 L=LEN(A$):PRINT@14,L;
40 PRINT@32,MID$(A$,20,12)
50 PRINT@64,RIGHT$(A$,21)
60 PRINT@96,LEFT$(A$,52-33)
70 EXEC44539
80 PRINT@0,""
100 GOTO100
```

220 117
END 73

Listing 2: STRINGS2

```
0 '<LISTING2>
5 CLEAR 500
10 CLS
20 D$="BETTY ANN WHITE"
30 PRINT@8,LEFT$(D$,5);:GOSUB500
40 PRINT MID$(D$,6,5);:GOSUB500
50 PRINTRIGHT$(D$,5):GOSUB600
60 '***
70 A$="BETTY":B$=" ANN ":C$="WHI
TE"
80 PRINT@8+32,A$;:GOSUB500
90 PRINT B$;:GOSUB500
100 PRINT C$:GOSUB600
110 '***
120 PRINTTAB(8)A$;:GOSUB500
130 PRINTTAB(13)B$;:GOSUB500
140 PRINTTAB(17)C$:GOSUB600
150 '***
160 PRINT@8+96,A$:GOSUB500
170 PRINT@13+96,B$:GOSUB500
180 PRINT@18+96,C$:GOSUB600
190 '***
200 PRINT@8+128,MID$(A$,1,5);:GO
SUB500
210 PRINT@13+128,MID$(B$,1,5);:G
OSUB500
220 PRINT@18+128,MID$(C$,1,5):GO
SUB600
230 '***
240 PRINT@8+160,RIGHT$(A$,5):GOS
UB500
250 PRINT@13+160,RIGHT$(B$,5):GO
SUB500
260 PRINT@18+160,RIGHT$(C$,5):GO
SUB600
270 '***
280 PRINT@8+192,LEFT$(A$,5);:GOS
UB500
290 PRINT@13+192,LEFT$(B$,5);:GO
SUB500
300 PRINT@18+192,LEFT$(C$,5):GOS
UB600
310 '***
320 PRINT:PRINT"  USING THE THRE
E VARIABLES, A$;B$;C$; YOU CAN C
ENTER THE NAME, WITH PREGNANT PA
USES BETWEEN   VARIABLES, USING
 A VARIETY OF   TECHNIQUES.
330 PRINT:PRINT"  SOME ARE A WAS
TE OF TIME!"
340 GOTO 340
500 FOR Z=1TO200:NEXT:RETURN
600 FORZ=1TO500:NEXT:RETURN
```

16K ECB

It's Back to Basics With an Adjective Review

By Steve Blyn
Rainbow Contributing Editor

This month's article presents a grammar review program. It is suitable for a review of any part of speech within a sentence. We have chosen to illustrate with a review of adjectives.

Grammar is making a comeback in education. For many years, in the not too distant past, it was felt that by stressing grammar, student creativity would be stifled. Grammar was put on the back burner. Therefore, many students were educated with very weak skills in grammar. This deficiency came back to haunt them later in life. Colleges, especially, complained of poor basic writing skills of many entering freshmen.

The back-to-basics movement we have witnessed in the past few years includes and even stresses grammar. Correct grammatical usage is again part of most school systems' curricula. Our program helps to review parts of speech.

We chose adjectives as an example of the way to use this program. A sentence appears on the screen. An arrow appears underneath the first letter of the first word in this sentence. The student uses the right-arrow key to move the arrow underneath the first letter in the word which is the adjective.

Pressing the ENTER key indicates whether this is indeed the sentence's adjective. If correct, the child proceeds to the next sentence. If incorrect, the child repeats the same sentence until he guesses correctly.

There are 10 sentences in this program. You may have as many as you want. The number of sentences is indicated in Line 30 as variable 'N'. Line 60 chooses one of the sentences randomly and Line 110 prints it on the screen.

Lines 130 to 180 contain the routine to move the arrow. The arrow is displayed by using CHR$45. Line 150 erases the arrow and moves it to the right whenever the right-arrow key is pressed. Line 160 does the same to the left. When CHR$ 13 (the ENTER key) is pressed, the program jumps down to Line 190.

Lines 190 to 210 check the student's current positioning of the underline arrow. This indicates whether the student has selected the adjective in this sentence. If the student is correct, he may press the ENTER key to get the next sentence or the 'E' key to end the program. If incorrect, he is instructed to press the ENTER key to try again.

Lines 280 to the end of the program contain the DATA statements. Each contains two parts. The first is the sentence. We limited our sentences to under 32 characters to allow them to fit on one line. You do not have to stick to this idea. It does, however, make for a neater screen display. The second part of the DATA line is the number of characters counted until the place where the first letter of the adjective occurs in the sentence.

This program is easily modifiable for nouns, verbs, pronouns, adverbs or any part of speech you want to test. We even tried a version with scrambling the words of a sentence and asking the students to move the arrow to the correct first word of the scrambled sentence.

We encourage you to use your creativity to find additional uses for this program. Many times it happens that you are looking for a program to cover particular subject areas or skills. You find that what you need has just not been written commercially. The reason for this is most often a marketing decision. It is not worth the time, effort and cost to a software company to produce a program that has limited appeal. The programs appearing in these monthly columns are meant to be modified to fit particular needs. They are written so that some simple modifications can be performed by those who do not know how to program.

You could work on this *Adjective Review* and turn it into a Noun, Verb, Adverb, Etc., Review, save each version, and eventually cover all parts of speech. Modifications are meant to be done on many of the programs that appear in this column. We encourage you to do so. You can develop a personal library of educational programs that focus on skills for your needs. □

Steve Blyn teaches both exceptional and gifted children, holds two master's degrees and has won awards for the design of programs to aid the handicapped. He owns Computer Island and lives in Staten Island, New York.

The listing: ADJECTIV

```
1Ø REM"GRAMMAR REVIEW-ADJECTIVES
"
2Ø REM"STEVE BLYN,COMPUTER ISLAN
D,STATEN ISLAND,NY,1986"
3Ø N=1Ø
4Ø DIM A$(N),A(N)
5Ø FOR T=1 TO N:READ A$(T),A(T):
 NEXT T
6Ø R=RND(N)
7Ø CLS
8Ø PRINT@4Ø,"adjective review";
9Ø PRINT@72,STRING$(16,255);
1ØØ H=224
11Ø PRINT@192,A$(R);
12Ø PRINT@H,CHR$(45);
13Ø EN$=INKEY$
14Ø IF EN$=CHR$(13) THEN 19Ø
15Ø IF EN$=CHR$(9) THEN PRINT@H,
CHR$(143);:PRINT@H+1,CHR$(45);:H
=H+1
16Ø IF EN$=CHR$(8) THEN PRINT@H,
CHR$(143);:PRINT@H-1,CHR$(45);:H
=H-1
17Ø PRINT@192,A$(R);
18Ø GOTO 13Ø
19Ø G=H-223
2ØØ IF G=A(R) THEN 22Ø
21Ø IF G<>A(R) THEN 25Ø
22Ø PRINT@364,"CORRECT";:PLAY"O4
L1ØØCDEFGECCCC":PRINT@422,"PRESS
 ENTER TO GO ON";
23Ø EN$=INKEY$
24Ø IF EN$=CHR$(13) THEN RUN ELS
E IF EN$="E" THEN END ELSE 23Ø
25Ø PLAY"O2L2ØBB":PRINT@355,"PRE
SS ENTER TO TRY AGAIN";
26Ø EN$=INKEY$
27Ø IF EN$=CHR$(13) THEN 7Ø ELSE
 26Ø
28Ø DATA SHE IS A BIG GIRL.,1Ø
29Ø DATA HE WENT TO THE GROCERY
STORE.,16
3ØØ DATA DAVID ATE A CHOCOLATE C
OOKIE.,13
31Ø DATA THE FAT CAT SAT DOWN SL
OWLY.,5
32Ø DATA MY FRIENDLY DOG'S NAME
IS SPOT.,4
33Ø DATA WE WENT TO SEE A SCARY
MOVIE.,18
34Ø DATA I ATE THE SWEET PEACH Q
UICKLY.,11
35Ø DATA CAN YOU DRAW A PRETTY P
ICTURE?,16
36Ø DATA WHERE IS MY BIG BOOK NO
W?,13
37Ø DATA THE HEAVY DOOR CREAKED
NOISILY.,5
```

Don't String Me Along

Use this technique to track down FC Errors

By Ellen and George Aftamonow

MOST computers don't hesitate to tell us where we went wrong and what sort of mistake we made this time. We are all too familiar with SN Error in 100, TM Error in 250, etc. In case, one simply looks at the given line number and corrects it.

However, this is not necessarily the case with the FC (function call) Error. All too often an examination of FC Error shows that the given line number has no error in it at all. Many people then sit down and pen a letter to the author or the magazine to proclaim that the program does not work. But, before we're so quick to blame the program, we should do a little detective work.

When you get an FC Error message, first check the given line. If the line is correct, then the most likely suspect is a previously defined string. For instance:

```
100 I$(1)="U8BR3R2ND8R2BD8"
110 W$="BR5L2NU5L2HU7BR6D7GBR4"
120 O$="U8R4D8NL4BR2"
130 R$="U8R4FD2GL2F4BR2"
140 K$="U8BD4NE4F4BR2"
150 S$="BRNHR4EU2HL4HU2ER4BR4BD8
"
160 PL$="T200L10004AAABBBCCC"
170 PMODE3,1:SCREEN1,1:PCLS
180 DRAW"BM70,100S8XI$(1);BR8XW$
;XO$;XR$;XK$;XS$;":PLAYPL$
190 FORX=1TO5000:NEXTX
```

In this example, if the CoCo greets us with an FC Error in 180 and Line 180 lists correctly, we should backtrack to lines 100 through 160, where we first defined the various strings. All Line 180 does is execute the strings that appear in lines 100 through 160. So it stands to reason that if a string was defined wrong. Line 180 cannot be executed, thus the FC Error.

The easiest way to pick out the culprit is to insert a quote and REM ("') after a suspect string, using the edit mode. Thus, Line 180 becomes:

```
180 DRAW"BM70,100S8XI$(1);"'BR8X
W$;XO$;XR$;XK$;XS$;":PLAYPL$
```

If the program reaches Line 190, then the error was not in I$(1) in Line 100. So delete the "' and insert them after the next string.

```
180 DRAW"BM70,100S8XI$(1);BR8XW$
;"'XO$;XR$;XK$;XS$;":PLAYPL$
```

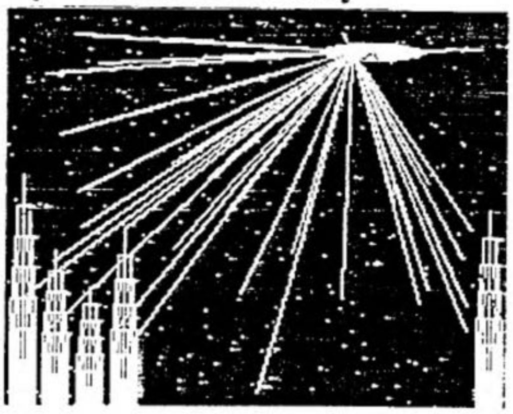
Continue in this manner until you get the FC Error. You will then know which string has the error and you can look for an error in the line where the string is defined. Often times it is the letter 'I' which should have been number one, the letter 'O' which should have been number zero, of the letter 'B' which should have been number eight. So when you see an FC Error, dont't let it string you along.

The Aftamonows are self-taught programmers living in Milford, Connecticut. Ellen holds a degree in math and concentrates on the structure of the program, while George creates and designs graphics.

(Questions about this technique may be directed to the authors at 46 Howe Street, Milford, CT 06460, 203-878-3062. Please enclose an SASE when writing.)

Space Attack

By Patrick J. Benway



This short program uses the BASIC commands of CIRCLE, LINE and PSET-PRESET to demonstrate a space city road.

The Listing: RAID

```
10 CLS:PRINT@196,"---SPACE-CITY
(RAID!)---":FORJ=1TO255STEP3:SOU
NDJ,1:NEXT
20 PMODE4:PCLS:SCREEN1,1:FORJ=1T
O300:PSET(RND(255),RND(191)):NEX
T
30 FORJ=1TO7:CIRCLE(12,185),J,,1
5:CIRCLE(28,187),J,,10:CIRCLE(45
,195),J,,8:CIRCLE(62,225),J,,12:
CIRCLE(238,187),J,,13:NEXT:SOUND
255,50
40 FORJ=1TO25:CIRCLE(180,20),J,,
.200:NEXT:SOUND1,50:SOUND150,8:S
OUND50,1:LINE(180,20)-(30,120),P
SET:LINE(180,20)-(30,120),PRESET
:SOUND1,50:SOUND150,8:SOUND50,1:
LINE(180,20)-(245,150),PSET:LINE
(180,20)-(245,150),PRESET:SOUND1
,50
50 SOUND200,20:LINE(10,100)-(170
,20),PSET:LINE(10,100)-(170,20),
PRESET
60 FORJ=1TO100:A=RND(255):B=RND(
191):LINE(170,20)-(A,B),PSET:NEX
T
```

Getting to the Details of the CoCo 3

By Marty Goodman

Q. *I hear the new CoCo 3 will have an RGB output. Does this mean I can use the same RGB monitor I now use on my IBM PC?*

A. No. The new CoCo 3 does have an RGB output, but it is an RGB analog output, not the RGBI-type signal protocol used for most standard IBM PC color displays. The RGBI used by the IBM systems is characterized by its signals at TTL levels (five volts or zero volts — nothing in between). It allows for a maximum of 14 colors plus black. RGB analog allows for a great many more colors and, as such, is a superior protocol. It may be possible to modify many RGBI-type monitors to accept RGB analog signals by merely removing a chip or two inside the monitor and properly biasing the bases of the R, G, B and synch input transistors. But apart from such hacker manipulations, to fully appreciate the impressive color graphics capability of the CoCo 3 you will have to either buy the $300 CM-8 monitor from Tandy, or use one of the few other RGB analog capable monitors. Both Magnavox and Sony make a few monitors that are RGB analog capable. There are two minor variants of RGB analog. One is the kind used by the CoCo 3, where the R, G and B signals are separate and there are separate synch signals. The other is the protocol used by the Amiga computer, where the synch signal information is tacked on to the Green luminance line.

Q. *I know the CoCo 3 features much improved graphics resolution. But the CoCo 3's joystick inputs are of the same zero-to-63 low resolution as those of the old CoCo 2. How can I achieve smooth positioning of a cursor or character on the CoCo 3 screen using the joystick? Will the* CoCo Max *Hi-Res joystick help?*

A. Currently, the only way to get Hi-Res joystick control on a CoCo 3 involves one of several programming tricks: for instance, using a fine control box (like that used by *Graphicom*) or using the analog joystick as a time-controlled, four-switch joystick via a software emulation of such an Atari-type joystick. *CoCo Max*'s Hi-Res joystick hardware (and the program itself) will not work on the CoCo 3 due to the hardware using a port address that conflicts with assigned addresses used by the GIME chip in the CoCo 3. But it has been rumored that a low-cost adaptor will soon be available. It will plug in between the joystick ports and computer (on both the CoCo 2 and 3) and will greatly increase the available resolution of the joysticks. Keep an eye on new products from Tandy; help is on the way.

Q. *I am told by Tandy that all of their hardware for the CoCo 2 will be compatible with the CoCo 3. Is this so? What about hardware and software from non-Tandy sources?*

A. To the best of my knowledge, all Tandy hardware for the CoCo (Multipak, Disk controller, RS-232 Pak, Hard Disk Controller, Speech Sound Pak, and such) is fully compatible with the CoCo 3. Similarly, all third party disk controllers (those from J&M systems, HDS and Disto) should also work fine with the CoCo 3. But in order for them to work with the CoCo 3, they need to have an unmodified version of Disk BASIC 1.1. The PBJ 2SP pack is also fully compatible with the CoCo 3, as is the Disto RAM disk card. The 80-column card from PBJ will probably *not* work on the CoCo 3, although it is not needed due to the 80-column capability of the CoCo 3.

CoCo Max will not work in its current form on the CoCo 3, in part because of hardware conflicts. But it will very likely be re-released in a CoCo 3 compatible version.

Due to differences in the handling of memory on the CoCo 3, much other well-known CoCo 2 software (*Telewriter, VIP Writer, Mikeyterm, Graphicom, Color Com E, etc.) will not* work in their original forms on the CoCo 3. However, patches for these and other popular CoCo 2 programs will most likely appear soon.

Martin H. Goodman, M.D., a physician trained in anesthesiology, is a longtime electronics tinkerer and outspoken commentator — sort of the Howard Cosell of the CoCo world. Marty is the database manager of RAINBOW's CoCo SIG on Delphi. His non-computer passions include running, mountaineering and outdoor photography. Marty lives in San Pablo, California.

A look at the internal hardware

Dissecting the CoCo 3

By Cray Augsburg

The following is a list and brief description of the major components and areas on the Color Computer 3 circuit board.

A) Transformer Assembly — This transformer has the same specifications as the one used in the CoCo 2. As in previous designs, the Color Computer 3 draws power from the wall as long as it is connected to an outlet. The amount of power it draws, however, is small when the machine is not turned on.

B) Power Switch — When turned on, allows current to flow to the Color Computer 3's logic circuitry.

C and D) Left and Right Joystick Connectors — Close examination reveals that the sixth pin, which was unused in previous designs, is now connected. This, along with the enhanced software, allows the Color Computer 3 to recognize both buttons on a Deluxe Joystick.

E) Serial I/O Connector — This four-pin jack accepts Radio Shack's de facto standard for RS-232 devices. Enhancements elsewhere in the machine allow more reliable operation at much higher speeds than on previous CoCos.

F) Cassette Port — This five-pin connector allows you to hook a cassette recorder to the Color Computer 3.

G) RF Modulator — This unit changes the video signal so the Color Computer 3 can drive a television display. This circuit was present on all older CoCos and most CoCo 2s.

H) RF Channel Select — For selecting whether the TV display receives the Color Computer's output on VHF channel 3 or 4.

I) Composite Video Output — This RCA phono jack supplies a composite signal for driving a composite color monitor. The Color Computer 3 is not set up to drive a monochrome monitor when you take it out of the box.

J) Audio Output — This RCA phono jack supplies a line-level audio output. It may be connected to the monitor's audio-in jack or to an external amplifier. It will operate even if you are using a TV or an RGB monitor for the display device.

Cray Augsburg is RAINBOW's technical assistant and has an associate's degree in electrical engineering. He and his wife, Ruth Ann, have two children and live in Louisville, Kentucky. His username on Delphi is RAINBOWMAG.

K) Reset — As always, this switch does not destroy memory contents, but causes the computer to stop execution of a currently running program. However, if you have used POKEs or machine-language routines to alter the BASIC routines, they will be changed back to normal by the use of the Reset button.

L) RAM Area — The Color Computer 3 (128K version) contains four 41464 RAM chips. These chips are 64K by 4-bit, dynamic RAM chips. These chips are removed when the machine is upgraded to its limit of 512K RAM.

M) Microprocessor — The Color Computer 3 uses the Motorola 68B09E microprocessor. This 40-pin MPU is designed for reliable operation up to 2 MHz and, as with previous CoCos, gets its clock signal from an external source.

N) ROM Port — This 40-pin cartridge/expansion port accepts existing ROM Paks or the MultiPak Interface. If you intend to use a MultiPak Interface with the new machine, you need to get the MPI fixed at your Radio Shack Service Center first. Apparently, there was a bug in the PAL chip on the MPI. The fix is expected to cost $6 plus installation charges.

O) Memory Expansion Connectors — These three 12-pin header connectors are designed to receive the 512K RAM upgrade board. The 512K upgrade consists of a satellite board containing 16 256K by 1-bit dynamic RAM chips.

P) Keyboard Connector — The Color Computer 3 uses the same clear Mylar cable for its keyboard connection as the 'F' board and later CoCos used.

Q) Power Supply Circuitry — This is where the incoming power, after being stepped down by the transformer, is rectified, regulated and filtered. This section supplies +/-5 volts regulated, and an unregulated 12 volts.

R) 68B21 PIA — Used to drive portions of the video as well as the cassette and sound circuitry of the Color Computer 3.

S) 68B22 PIA — This open-collector device drives the Color Computer 3's keyboard.

T) Clock Crystal — Unlike its predecessors, which used a clock crystal of frequency 14.31818 MHz, the Color Computer 3 uses a crystal with a frequency of 28.63636 MHz. This, combined with the new circuitry in the machine, allows much faster operation.

U) The GIME — This flat-pack is a revolutionary design from Tandy. The GIME (for Graphics, Interrupt, Memory Enhancement) combines the functions of the 6847 (VDG) and the 6883 (SAM) from previous CoCos. In addition to supplying bipolar RAM for faster video action, the GIME manages the extended memory of the Color Computer 3 despite the fact that the 68B09E can directly address only 64K of memory. The GIME can be looked at as the "hardware handler" of the Color Computer 3 as the 68B09E is looked at as the "software handler." It is the coolest-running chip in the Color Computer 3.

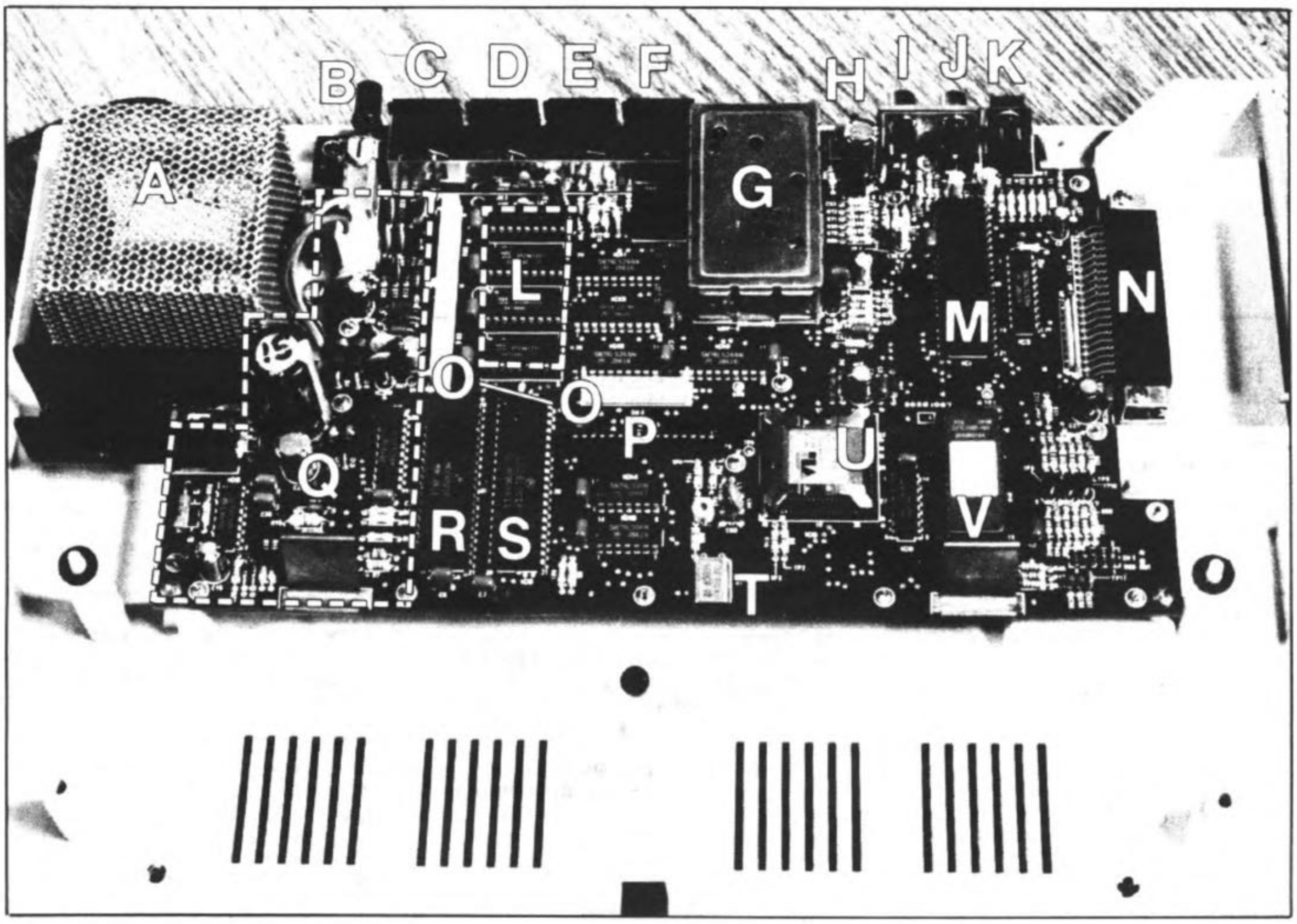
V) ROM — This 32K by 8-bit ROM contains Microsoft Extended BASIC and the overlay enhancements produced by Microware for Tandy. All Color Computer 3s come with this Enhanced Extended BASIC.

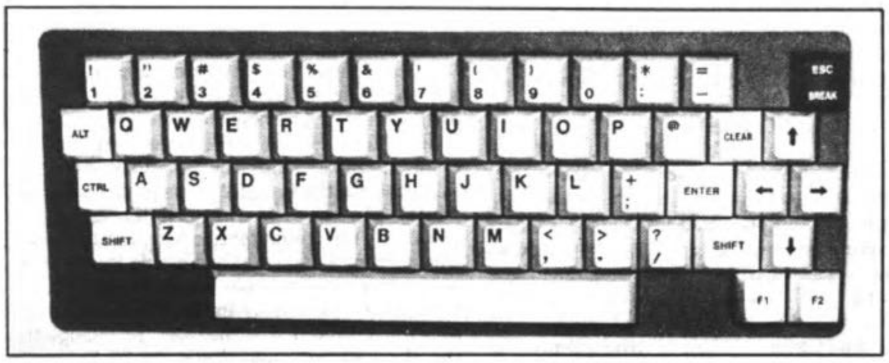
** Not shown in these pictures is the RGB monitor connector on the bottom of the new Color Computer. It is a 10-pin header connector unlike the DB9 connectors used by other manufacturers. However, only nine slots on the monitor connector are used and one pin is blocked to eliminate the possibility of plugging the monitor in backwards. For more information about the differences between color composite and RGB, refer to Ed Ellers' article on Page 27 of the September 1986 issue.

Some Observations

Many people have expressed concern about whether the Color Computer 3 supports artifact colors. The new machine does support artifact colors when used with a television or color composite monitor (an RGB monitor will produce the image, but only in black and white). However, in the past the color set chosen by the computer has been random and was selected by repeatedly pressing Reset. This was not a very reliable method. The Color Computer 3 powers up in the same configuration every time it is turned on. To change to the alternate set, hold down the F1 key and press Reset one time. The computer will switch to the alternate set. To switch back, just press Reset one time.

The Color Computer 3 is designed to operate at 0.894 or 1.788 MHz. When turned on, the machine is set to run at 0.894 MHz. However, since the new machine is always operating from RAM (contents of ROM are copied and overlayed in RAM on power-up), the RAM speed-up POKE will work. Just `POKE 65497,0` to use the 1.788 MHz clock speed. Type `POKE 65496,0` to go back to 0.894 MHz. □

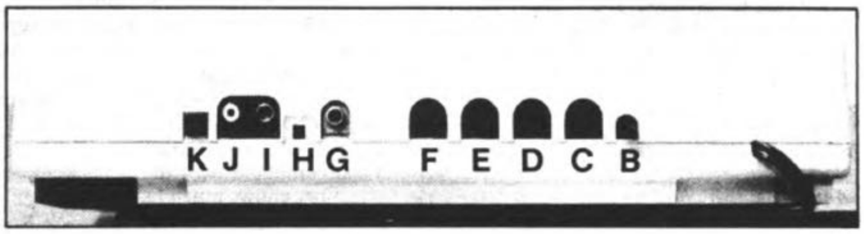




Above: A view of the Color Computer 3's circuit board as seen when looking from the front of the computer. The keyboard has been removed and the RGB monitor jack is mounted beneath the board on the right-hand side.

Left: A view of the Color Computer 3's keyboard. The two function keys are on the bottom-right, while the CONTROL and ALT keys are on the left side.

Right: The back of the Color Computer 3. All letter designations coincide with those in the circuit board view as well as those in the text.



The Power of the Palette: Graphics on the Color Computer 3

By Rick Adams and Dale Lear

We're all excited that the increased resolution and number of colors of the Color Computer 3 graphics display produces more spectacular and colorful graphics. But there are other implications to the method of graphics support provided by the new Graphics Interrupt Memory Enhancer (GIME) chip that are even more astounding.

With the previous SAM/VDG chips in the Color Computer 1 and 2, a maximum of four colors was available, chosen from one of two available sets of four specific colors — no exchanges or substitutions allowed. With the GIME chip, all the rules of color selection for graphics display have changed. You may display up to 16 colors out of a palette that contains your own color set chosen from a total of 64 possible colors.

Thus, Color Computer 3 software utilizes more high resolution displays with many more colors than we've seen previously. The edges of objects on the screen are smoother, too. The ability to choose your own color set leads to a less cartoon-like representation of objects on the display, with less dependence on hacker tricks like color "aliasing" (artifacting) to generate more appropriate colors.

Less obvious, but very important to note, is the fact that this palette scheme of specifying color sets enables us to use a completely new form of computer animation. Presently, there are two major methods of animating Color Computer graphics: the screen-flip technique and the draw-redraw technique. Screen-flip involves keeping two copies of the screen, drawing one of them while the other is being displayed, then reversing the process. Draw-redraw simply means that you use one screen which is displayed all the time; your spaceship (or whatever) is erased at its previous position, and redrawn at its new position. But now we also may use a third method, called the palette-switching method: Display the entire screen, including objects drawn in various colors, and then change the values of the colors set in the palette after they are drawn.

If you change the red in your palette to blue, then all of the objects previously displayed in red will *instantaneously* change to blue — just like magic! With a little trickery, this technique can be used to make portions of the screen flash, or pulse on and off in various colors. Objects may be instantly changed to the background color (making them disappear), or changed from the background color to a visible color, making them seem to appear out of nowhere. A bird could be made to flap its wings by making the up position of the wings visible, then making the up position disappear and making the down position of the wings visible. So here is another major new graphics animation technique available to the Color Computer 3 user. No longer are we held to merely four colors. We're only limited to 16 colors at a time . . . or are we?

Another new piece of hardware in the CoCo 3, the programmable interrupt timer, enables us to use yet another new technique to provide up to 64 colors on the screen at a time! Using this technique, the programmable interrupt timer is set to interrupt the computer four times during every screen redraw. At the top of the screen, the interrupt routine sets the palette with 16 colors. One-quarter of the way down the screen, the timer interrupts again. Sixteen *other* colors are put into the palette, and so on. In effect, one 16-color palette is active for the first one-quarter of the screen, another palette is active for the next quarter screen, and so on.

Sure, it's one of those nasty hacker tricks, and the normal BASIC user isn't going to want to bother with it. But software developers just love this kind of thing, and you can expect them to use it to their advantage.

So, if you see some software come out that uses 64 colors at once, don't scratch your head and say "that's impossible." You'll be able to say, "Hey, I know how they did that; I read about it in THE RAINBOW!"

Rick Adams is a systems programmer for a company that develops 68000-based systems software. In addition to writing games, he likes science fiction and is the author of Radio Shack's Temple of ROM. *Rick lives in Rohnert Park, California.*

Dale Lear owns Dale Lear Software and makes his living developing programs for the Color Computer. He has authored games and other software such as Double Back, Baseball, TSEDIT, TSWORD *and* D.L. *LOGO. Dale, his wife Laurel and their six children live in Petaluma, California.*

Rainbow Tunnel

The Rainbow Tunnel is a short BASIC program that demonstrates the range of colors available on the Color Computer 3, while at the same time showing an interesting use of the PALETTE command to provide animation.

Lines 90 through 140 set the high speed mode, and tell BASIC to go to the end of the program at Line 640 if the BREAK key is pressed. The high-speed POKE is *guaranteed* to work on the Color Computer 3. And the new ON BRK command is a welcome addition.

Lines 160 through 250 load the graphics palette with a set of colors that closely approximate the spectrum from red to purple — a rainbow, in other words. The color codes used appear in the DATA statement at Line 200. Lines 270 through 360 create, and then paint, a series of concentric circles. The circles are painted with the colors of the rainbow. In lines 380 through 440, the concentric circles that formed the borders for the PAINT command are drawn again in colors that match the painted regions near them, rather than in colors that were appropriate to use for a paint border.

So far, we have a brilliant, multicolored display on the screen, but where is the animation? Have patience. When the program gets down to Line 490, the magic begins. The loop at Line 490 looks pretty simple, so check the subroutine at lines 520 through 580. The palette colors are changed in such a way that each concentric circle appears to move one position outward, thus giving the illusion that you are traveling down a brightly-colored "rainbow tunnel." But actually, nothing is moving at all! The color assignments, specified by the palette values, are moving, not the actual display data. This effect would be even more spectacular from assembly language, in which a considerable delay would have to be put in the loop so that the display would not look like a blur!

Finally, we come to the end routine at lines 600 through 650. The PALETTE RGB command sets the colors back to their defaults. Otherwise, when we press BREAK, the screen might be in a color set so weird we couldn't read it.

When we watched this program run for the first time, there was a long silence, followed by this conversation:

Dale: "You *couldn't do* that before on a Color Computer!"

Rick: "You couldn't even do it *badly*!"

Who's Waggin' the Wheel?

Wagon Wheel is a short BASIC program that demonstrates a new animation technique that was unavailable to the color computer world until the CoCo3.

Lines 120 through 150 perform a few set-up calls, including some new features.

Lines 200 through 350 draw a wheel with spokes. The spokes, however, are drawn in a very special way. Fourteen groups of equally spaced spokes are each assigned a different color (or palette register).

As you watch the spokes being drawn, they look colorful. However, they look a little close together, and they certainly don't appear to be moving!

Now the magic begins. Lines 420 through 440 set the 14 palette registers assigned to the spokes, all to white (the background color). What happens? All the spokes disappear.

Now we get to lines 510 through 560. By setting only one of the 14 palette registers assigned to the spokes to black, every 14th spoke appears. By constantly cycling through the 14 registers setting only one at a time to black, the wheel now appears to turn.

Of course, nothing is really moving. No drawing is being done at all. The palette values are changing, causing the illusion of animation. □

The Listing:

```
10 '***************************
20 '*     "RAINBOW TUNNEL"     *
30 '*    DEMO TO SHOW USE      *
40 '*  OF PALETTE REGISTERS    *
50 '*   TO SIMULATE MOTION     *
60 '*BY RICK ADAMS & DALE LEAR*
70 '***************************
80 '
90 '***************
100 ' SET HIGH SPEED
110 '***************
120 POKE &HFFD9,0
130 DIM CC(32)
140 ONBREAK GOTO 640
150 '
160 '***************
170 ' SET UP COLORS
180 '***************
190 PON 2
200 DATA 49,50,51,52,53,22,23,24
,55,56,57,58,59,60,61,62
210 FOR I=0 TO 15
220 READ CC(I)
230 CC(I+16)=CC(I)
240 NEXT I
250 GOSUB 560
260 '
270 '***************
280 ' PAINT CIRCLES
290 '***************
300 FOR I=0 TO 19
310 R=8+I*8
320 C=I AND 15
330 FORMAT(160,96),R,1
340 MON (156+R,96),C,1
350 MON (164-R,96),C,1
360 NEXT I
370 '
380 '***************
390 ' PAINT THE LINES
400 ' BETWEEN CIRCLES
410 '***************
420 FOR I=0 TO 19
430 FORMAT(160,96),8+I*8,I AND 1
5
440 NEXT I
450 '
460 '***************
470 ' LOOP
480 '***************
490 GOSUB 560
500 GOTO 490
510 '
520 '***************
530 ' SUBROUTINE TO
7226 ' CHANGE PALETTE
550 '***************
560 FOR I=0 TO 15:OS9 I,CC(I+K):
NEXT I
570 K=(K-1)AND 15
580 RETURN
590 '
600 '***************
610 ' RESET PALETTE
620 ' ON BREAK
630 '***************
640 OS9 !
650 STOP
```

The Listing:

```
10 '***************************
20 '*       "WAGON WHEEL"      *
30 '*     DEMO TO SHOW USE     *
40 '*  OF PALETTE REGISTERS    *
50 '*    IN ANIMATATION        *
60 '*BY RICK ADAMS & DALE LEAR*
70 '***************************
80 '
90 '**************
100 ' SET UP
110 '**************
120 POKE &HFFD9,0
130 PON 2
140 AUTO(1)
150 OS9 0,24
160 '
170 '*********************
180 ' DRAW OUTSIDE OF WHEEL
190 '*********************
200 FORMAT (160,96),90,0
210 MON (0,0),0,0
220 '
230 '*********************
240 ' DRAW SPOKES
250 K=14*8
260 '*********************
270 FOR I=0 TO K-1
280 X=90*SIN(I*3.14/K)
290 Y=90*COS(I*3.12/K)
300 DUMP 2+14*(I/14-INT(I/14)),1
310 ! (160+X,96+Y)-(160-X,96-Y),
PSET
320 NEXT I
```

continued on Page 43

Inside the CoCo 3

By Marty Goodman

This is a collection of observations made after examining the insides of a Color Computer 3 and comparing its ROM to that of a CoCo 2.

ROM Addressing

The CoCo 3 has a 32K by 8-bit ROM. The lower 16K of this ROM contain code that is *nearly* the same as that in the 16K of Color BASIC and Extended Color BASIC, with the following changes:

The copyright message in the Extended BASIC part of the ROM is altered, as is the version number in the Color BASIC ROM.

The part of Extended BASIC that formerly contained code for the DLOAD command is now completely different.

The startup sequence in Color BASIC, including the RAM chip selector and memory size checker, as well as the warm/cold start reset sequence code, is all rewritten.

The keyboard routine in Color BASIC has been rewritten (possibly to allow use of the keyboard interrupt, which would considerably speed the execution of Color BASIC).

The vectors set at the end of the Color BASIC ROM are now all pointing in different places.

Apart from these relatively minor changes, there exists a complete image of the Color BASIC and Extended BASIC ROMs in the lower part of the 32K by 8-bit ROM.

The GIME chip supports three modes for addressing ROM in the CoCo 3. In one of these modes, only the lower 16K of ROM is addressed internally, and the remaining 16K of addressable ROM is looked for on the cartridge port. In this mode, the ROM in the CoCo 3 should be able to be made to closely emulate the appearance of the ROMs in a CoCo 2. In fact, even programs that use undocumented calls to the ROM should not be compromised on the CoCo. The two low-order bits of $FF90 control the mapping of the available CoCo 3 ROM memory. Note that the CoCo 3 can, via those bits, be made to address a full 32K of ROM on a ROM pack, allowing it to support up to 64K total of ROM in the system.

Martin H. Goodman, M.D., a physician trained in anesthesiology, is a longtime electronics tinkerer and outspoken commentator — sort of the Howard Cosell of the CoCo world. Marty is the database manager of RAINBOW's CoCo SIG on Delphi. His noncomputer passions include running, mountaineering and outdoor photography. Marty lives in San Pablo, California.

RAM Upgrades

The CoCo 3 is delivered as a 128K machine, with expansion to 512K of memory via a plug-in board. The 128K unit has four 18-pin, 4-bit wide by 64K 4464-type DRAM chips. The 512K add-on board is inserted after *removing* the four 4464 chips, and that board has on it sixteen 1-bit by 256K 41256 DRAMs. Presently the add-on board is the only option for expanding the addressable memory of the CoCo 3.

The board is easy to duplicate, and it is likely that third-party suppliers will soon be carrying versions of it, probably priced somewhat below Tandy's $150 price. In theory, a sensible way to upgrade the CoCo 3 would be to replace the four 4464 DRAMs with four 4-bit wide by 256K 1-megabit DRAMs. But that sort of chip is barely on the drawing board, and its production and sale at less than astronomical prices is not likely to occur soon.

Such a 4-bit wide by 256K chip is quite different from the 1-bit wide by 1-megabit chips that are already being sold in the $50 per chip price range. The 1-bit wide by 1-megabit chips should soon be an economic reality. But a 4-bit wide by 1-megabit chip will, as I stated, be a long time coming.

The RAM is arranged so a 16-bit wide data bus is available to the video circuitry. This allows data to be put on the screen much faster than on the old CoCo 2, providing for higher resolution and more colors in the CoCo 3 display.

RAM Addressing

The GIME chip supports a complex and powerful memory manager far more sophisticated than the crude bank switching arrangements used in CoCo 2 RAM upgrades such as Thunder RAM and the J&R Banker. The memory manager allows you to take any group of eight 8K segments in the full 512K address space and map them into the 64K of available memory directly addressable by the 6809. The old CoCo 2 memory upgrades could move memory around only in clumsy 32 or 64K blocks and were far more limited in how they could shuffle such blocks. The control addresses for the memory manager are in the $FFA0 to $FFAF address range.

This sophisticated memory management is what allows the CoCo 3 to run OS-9 Level II. Writers of dedicated applications for the CoCo 3 also find this powerful memory manager allows them to easily and quickly address the half-megabyte of the CoCo 3 without disrupting programs running in part of the 6809's address space. The CoCo 3 could become an attractive machine for scientific and industrial tasks because of its low price and high performance.

When it boots up, the CoCo 3 reserves memory at $FE00 through $FEFF for special system functions, including interrupt handling. The GIME hardware is set up to hold the top 256 bytes of addressable RAM (located just below the control I/O ports of $FF00 through $FFFF) constant through all memory manager address changes. This hardware feature is necessary to implement OS-9 Level II.

But the need to keep memory in this area constant will be *the* single most common cause of incompatibilities between CoCo 2 Disk BASIC software and the CoCo 3. It may be possible, using a switch at Bit 3 of $FF90, to turn off that reservation of those top 256 bytes and, via other manipulations, to more closely emulate the old CoCo 2 environment. Alternatively, it may prove easier for many software makers to do the minor rewrite needed to leave that address area alone. In many cases, this may be all the change needed to make "incompatible" CoCo 2 software run on the CoCo 3.

Emulation of Old SAM Functions

VDG related functions — Addresses $FFC0 through $FFD3 function on the GIME in exactly the same way they did on the old SAM, providing for total emulation of all documented old SAM/VDG functions.

Memory related functions — Addresses $FFD4 and $FFD5 (the page switcher) are supported on the GIME chip. The RAM/ROM switcher at $FFDE and $FFDF that switches 32K of ROM with 32K of RAM is supported too. Thus, many Disk BASIC programs that run in a "96K" environment on the CoCo 2 will still work on the CoCo 3.

Both *Graphicom* and *WEFAX* appear to work properly on the CoCo 3. These are examples of Disk BASIC 96K programs that use only documented calls to ROM vectors and do not mess with the top 256 bytes of available RAM.

"The GIME chip supports three modes for addressing ROM in the CoCo 3."

Not surprisingly, $FFDA through $FFDD ports on the old SAM set up for 4K, 16K or 64K of memory using the old CoCo and CoCo 2 chip arrangement, are *not* supported on the GIME chip. No great loss here, except to insiders who used the SHIFT/BREAK/Reset technique to make RAM snapshots.

Clock control — The CoCo 3 uses a primary crystal that works at *twice* the speed of that used in the CoCo 2. This is an 8X colorburst crystal: 28.63636 MHz. The old speed up POKE at $FFD6 and $FFD7 that would make the CPU address the ROM at twice normal speed (but still address RAM at its normal speed) is *not* supported on the CoCo 3.

But before you get alarmed, rest assured that when Tandy took that away, they gave us something *much* better: The port at $FFD8 and $FFD9 on the old CoCo caused the ROM and RAM to be addressed at double speed, but terminated RAM refresh and completely destroyed the old CoCo and CoCo 2's video display. However, on the CoCo 3, this "super high speed" POKE is now *fully* supported, the RAM memory *is* refreshed and the video display is unaffected. This means you can properly run your Disk BASIC programs at full double speed on the CoCo 3, though you may have to drop back to normal speed during such functions as cassette and disk I/O and sound generation.

Video Display of Text

I have experimented with displaying the CoCo 3's video on quality amber monochrome monitors. Initially, at power up, the display had the ugly vertical stripe distortion that is typical when you put a color signal on a monochrome monitor.

Although the GIME supports turning off the color signal via a port (Bit 4 of $FF98), poking under BASIC to this port was of limited value because the port is reset each time a new BASIC print statement is executed. Later on, we may find an easy way to properly shut off the color when in BASIC.

But, for now, by properly altering the foreground and background colors using the sophisticated palette control of the CoCo 3, we can make the CoCo 3 produce a credible image on a monochrome monitor. Even in the 80-column display mode, the image is quite readable. Somewhat to my disappointment, although the 80-column set was not all that bad, I found its sharpness and crispness somewhat inferior to that of my PBJ Word Pak I 80-column card, and far inferior to that of my IBM PC clone. But part of this may have been due to a badly adjusted monitor, and part to my not having sufficient time to play with the color set. Both black letters on light background *and* light letters on black background can easily be produced. Underlining is supported. The character font is the same as that of the TI VDG chip.

Buying a composite video monochrome monitor (in the $60 to $120 price range) allows you to take advantage of the 80-column display of the CoCo 3. A color composite monitor *will not* support the 80-column display. If you want both 80-column display of text *and* full color capability, your only option is an RGB analog monitor. Tandy wants $300 for its CM-8. This is something of a bargain, actually, since Magnavox and Sony, who also make CoCo 3-compatible RGB analog monitors, want at least $70 more, though their products are more flexible and support other signal protocols as well).

Via the GIME video hardware, one can generate 32-, 40-, 64- and 80-column text screens, although, on a color TV, only the 32-column works well at all. The 40-column display will often be cut off by the overscan found on most commercial color TVs.

Add-on Hardware Addressing

The GIME uses lots of address space not used before by the SAM chip. It does leave open address ports between

$FF60 through $FF7F for use by Radio Shack and third-party developers.

Of these, $FF7F is used by the Multipak, $FF68 through $FF6F are typically used by the RS-232 Pak card and the Tandy Modem card (or PBJ 2SP card). $FF7D and $FF7E are used by the Tandy Speech Sound pak if they are in the system. Other devices addressed in this legal range are Ears, third-party voice packs and the Stereo Pak from Speech Sound. All of these should work just fine on the CoCo 3.

But woe to the manufacturer who did not heed the warnings given by Tandy to not use addresses outside of that range! Sadly, *CoCo Max* is one such; it will not work on the CoCo 3 in its present form. Hopefully, new *CoCo Max* hardware will be made that fixes this problem.

Among those pieces of hardware that will mess up the GIME chip and are therefore somewhat incompatible with the CoCo 3 is Radio Shack's Multipak Interface. Yes, there is a bug in the PAL chip in both the old and new Multipak interface that lets the port at $FF7F ghost to $FF9F. This conflicts with a "Horizontal Offset Register" in the CoCo 3's GIME. I have been told by an informed source that the problem only occurs in 512K CoCo 3s, and that a fix in the form of a new PAL chip will be provided by Tandy.

It has been rumored that this fix for the Multipak will cost $6, regardless of whether you have an old or new Multipak. This is a reasonable price for such a fix. This fix is not yet available at your service centers or at National Parts, but should be ready by the time 512K CoCo 3s reach your stores.

Compatibility

In some preliminary testing, I found that *Telepatched Telewriter* and *Mikeyterm*, two popular applications, both crash when booted on the CoCo 3. At present I am not sure of the exact reason or how to fix these. But fixes for both should be forthcoming. It certainly is true that many popular CoCo 3 Disk BASIC standbys will not work on the CoCo 3. But it is equally clear that Tandy bent over backward to try to preserve compatibility for both their own *and* for third-party software. Unfortunately, in many cases, their best efforts were not good enough. But the CoCo 3 is so nearly CoCo 2 compatible that it should not be very hard to fix existing CoCo 2 favorites to run on the CoCo 3. My one major criticism of Tandy in this regard is that they should have warned us long ago to stay out of the $FE00 through $FEFF region, much as they did clearly warn us not to use undocumented vectors.

Conclusion

Hopefully there will soon be new software taking advantage of the CoCo 3's vastly improved video display, RS-232 and memory capability, which will make the issue of CoCo 2 incompatibility under Disk BASIC less of a concern.

Special Note of Thanks:

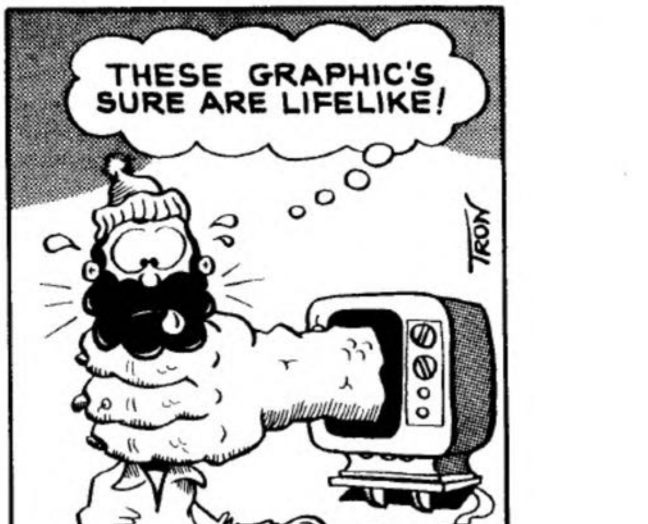
I would like to give special thanks to Tandy Corporation for giving permission to developers who had CoCo 3s to allow me to examine them and their documentation after the CoCo 3 was officially released. Without the kind cooperation of Tandy Corp, Steve Bjork and Dale Lear, it would be impossible for me to get this information out to the CoCo Community as early as this.

The Power of the Palette: Graphics on the Color Computer 3

continued from Page 40

```
330 FOR I=1 TO 30
340 FORMAT (160,96),I,0
350 NEXT I
360 '
370 '**********************
380 ' SET ALL PALETTE
390 ' COLORS TO WHITE
400 ' EXCEPT ONE
410 '**********************
420 FOR I=1 TO 15
430 OS9 I,255
440 NEXT I
450 '
460 '***********************
470 ' ROTATE WHEEL BY SETTING
480 ' ONE PALETTE REGISTER
490 ' AT A TIME TBLACK
500 '***********************
510 K=2
520 KK=K+1
530 IF KK=16 THEN KK=2
540 OS9 K,255:OS9 KK,0
550 K=KK
560 GOTO 520
570 '
580 '************************
590 ' RESTORE PALETTE ON BREAK
600 '***********************
610 OS9 !
620 STOP
```

LOOSE STRINGS / by Tron



Some Hardware Fixes for the Video Display Generator

By Tony DiStefano
Rainbow Contributing Editor

Last month, I described in detail the innards of the new CoCo B series computer. One difference inside this computer is a new version of the VDG (Video Display Generator). I described it as being an improved version of the old faithful VDG that has been in the CoCo since the beginning.

To make the new VDG compatible with the old one, the new functions of this VDG are not readily accessible. For instance, this VDG has a built-in lowercase character set. But press the old SHIFT/0 and nothing happens. You still get that crummy inverse video lowercase character. So what gives?

Well, in order to get it to work, you may have to add in a little hardware. This is where I come in. Get out the old soldering iron and dig in as I lead you through the modifications to get the most out of your new 'B' series computer. Note: The letter 'B' must appear on the model number of the computer and not inside on the PCB. For instance, the one I have is model number 21-3134B.

Tony DiStefano is well-known as an early specialist in computer hardware projects. He lives in Laval Ouest, Quebec.

Let's start with the basics. The old VDG chip number is Motorola MC6847. The new part is another Motorola part numbered MC6847T1, though in some computers, the part number might be XC80652P.

The first and most important change is the lowercase capability. Normally it is disabled, meaning you will not see the lowercase characters when using the SHIFT/0 on the keyboard. Instead, you get the normal inversed character set. You can change it in software. The pin that controls which mode you are in is connected to the PIA, which is memory mapped at $FF20 to $FF23, or 65312 to 65315 in decimal. It is connected to PB4 or Bit 4 of address location $FF22 or 65314. This bit is normally a zero. Changing this to a one gives you real lowercase characters. The only problem is the routine in Extended BASIC will change it back to a zero every time you print something. If you want to do it in BASIC, add this line every time you want to change the screen to true lowercase:

```
10 POKE &HFF22 , (PEEK (&HFF22)
OR 16)
```

What this line does is change Bit 4 to logical level one. But remember, each time you print on the screen or change from graphics to text, Extended BASIC changes this back. You may want to make this line into a subroutine. Better yet, why don't you do it in hardware? It's more permanent.

There are many ways of doing this change in hardware. Use the one that suits you best, but the first way I present is the simplest. Remove the chip from the socket. Bend Pin 30 (GM0) out so that it does not plug back into the socket. Solder a short piece of wire from Pin 30 to Pin 17. Pin 17 is the 5-volt supply. This action permanently changes the level of the pin to logical level one, giving lowercase all the time.

If the VDG is soldered into the board without a socket, then just cut Pin 30 at the base and pry it up. Use slim-line cutters or a razor blade. Be careful not to cut anything else.

The second way to make the hardware change requires an SPDT switch. Figure 1 shows two ways of wiring the switch to this circuit. Using Figure 1a as a guide, pull Pin 30 out as described before. Solder a wire from Pin 30 to the center of the switch. Solder another wire from one side of the switch to Pin 17 of the VDG. Solder a third wire to the other side of the switch and to Pin 1 of the VDG.

When the switch is toward Pin 17, the

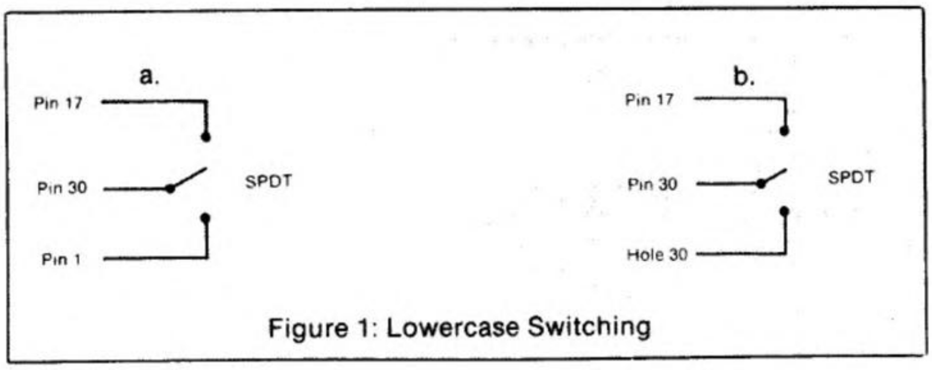


Figure 1: Lowercase Switching

display will always show lowercase characters. When it is the other way, it will always display inverse characters. Figure 1b shows basically the same way as before, but instead of connecting the third wire to Pin 1, connect it to the empty pinhole created when you pulled Pin 30. This way, when the switch is toward Pin 17, you always get lowercase characters. When the switch is the other way, you get whatever display Bit 4 of the PIA is set to. This is the most versatile way of connecting this pin.

The next change has to do with the border. In the normal text mode you see a big green square with black letters. This border is always black in the text mode. Now there is another alternative. How about a green border? There is a way of doing this in software. The pin that controls which mode you are in is connected to the PIA which is memory mapped at $FF20 to $FF23 or 65312 to 65315 in decimal. It is connected to PB6 or Bit 6 of address location $FF22 or 65314. This bit is normally a zero. Changing this to a one gives a green border. The only problem is that the same routine in Extended BASIC that changes the lowercase pin every time you print something also changes this pin. If you want to do it in BASIC, add this line:

```
10 POKE &HFF22 , (PEEK (&HFF22)
OR 64)
```

What this line does is change Bit 6 to logical level one. If you want to change both the lowercase and the green border, change the last value to 80 (16 + 64). The new line to change both the lowercase and green border would look like this:

```
10 POKE &HFF22 , (PEEK (&HFF22)
OR 80)
```

But remember, every time you print on the screen or change from graphics to text, Extended BASIC changes this back, so again, you may want to make this line into a subroutine. And again, this can be done in hardware.

One way to do this is to remove the chip from the socket. Bend Pin 27 out so that it does not plug back into the socket. Solder a short piece of wire from Pin 27 to Pin 17. This action permanently changes the pin to logical level one, giving a green screen all the time.

If the VDG is soldered into the board without a socket, cut Pin 27 at the base and pry it up.

The second way requires an SPDT switch. Figure 2 shows two ways of wiring the switch to this circuit. Pull Pin 27 out as described previously (see Figure 2a). Solder a wire from Pin 27 to the center of the switch. Solder another wire from one side of the switch to Pin 17 of the VDG. Solder a third wire to the other side of the switch and to Pin 1 of the VDG.

When the switch is toward Pin 17, the display will always have a green border; when it's the other way, it will always have a black border. Figure 2b is basically the same way as before, but instead of connecting the third wire to Pin 1, connect it to the empty pinhole created when Pin 27 was pulled. This way, when the switch is toward Pin 17, you always get a green border and when the switch is the other way, you get whatever display Bit 6 of the PIA is set to. This is also the most versatile way of connecting this pin.

The third modification is the famous inverse video screen. You no longer need to add a gate to do inverse video. The procedure is basically the same as the others, but with different values and different pin numbers. You can change it in software. The pin that controls which mode you are in is connected to PB5 or Bit 5 of address location $FF22 or 65314. This bit is normally a zero. Changing it to a one gives you an inverse video screen. But remember, Extended BASIC will change it back. If you want to do it in BASIC, add this line every time you want to change to an inverse screen:

```
10 POKE &HFF22 , (PEEK (&HFF22)
OR 32)
```

This line changes Bit 5 to logical level one. To change both the lowercase and the inverse video, change the last value to 48 (32 + 16). The new line to change both the lowercase and inverse video looks like this:

```
10 POKE &HFF22 , (PEEK (&HFF22)
OR 48)
```

Since Extended BASIC will change this back, again you may want to make this line into a subroutine. Don't bother to add the green border value when using the inverse video — it has a lower priority and shuts off anyway. Again, you can do it in hardware.

To make the change in hardware, remove the chip from the socket and bend Pin 29 out. Solder a short piece of wire from Pin 29 to Pin 17. This permanently changes the pin to logical level one, giving inversed video all the time. (Pin 17 is the 5-volt supply.)

If the VDG is soldered into the board without a socket, then just cut Pin 29 at the base and pry it up.

The second way uses an SPDT switch. Figure 3 shows two ways of

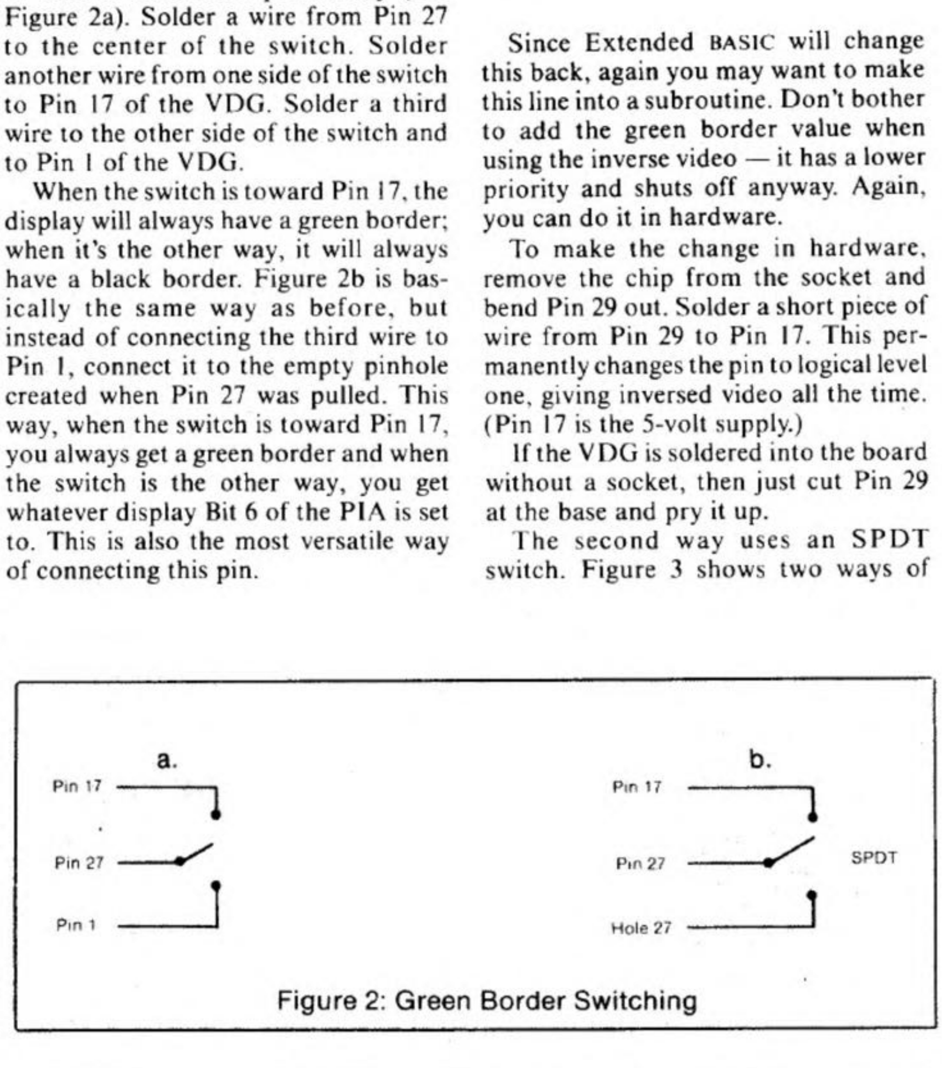


Figure 2: Green Border Switching

KISSABLE OS-9

Getting Revved Up For Fall Fun

By Dale L. Puckett
Rainbow Contributing Editor

Stand by for excitement! Sources confirm that the new Color Computer runs OS-9 Level II. Another OS-9 user who has seen the machine reported that its graphics capability is somewhere between the Atari ST-512 and the Commodore Amiga. Graphics programs running on both of these machines look super, so we are in for a real treat. I can hardly wait for all the details. While we're waiting, I'll give a review of OS-9 memory management schemes.

Much of the power of the new Color Computer will be made possible by Microware's OS-9 6809 Level II Operating System. On the surface, the casual user who only runs commercial programs probably won't notice much difference between OS-9 Level I and OS-9 Level II. Users who must deal with large data files in memory or programmers who want to run two or three tasks at a time through an OS-9 pipeline will notice a tremendous improvement.

Most of the problems we have run into with OS-9 on the original Color Computer are caused by the limited amount of memory available in the 64K of memory addressed directly by the 6809 microprocessor. While it's true that OS-9 based computers exist that use only 4K of ROM and 2K of RAM, these small computers are really controllers. Essentially, they run the same small machine code program forever, monitoring external real world conditions in real time, opening and closing the valves and switches that keep a

wiring the switch to this circuit. To use the first method (Figure 3a), pull Pin 29 out. Solder a wire from Pin 29 to the center of the switch. Solder another wire from one side of the switch to Pin 17 of the VDG. Solder a third wire to the other side of the switch and to Pin 1 of the VDG.

When the switch is toward Pin 17, the display will always have an inverse video; when it's the other way, it will always have a normal screen. The second method (Figure 3b) is much the same as the first. Instead of connecting the third wire to Pin 1, connect it to the empty pinhole. When the switch is toward Pin 17, you always get inverted video; when the switch is the other way, you get whatever display Bit 5 of the PIA is set to.

These three changes to the new VDG add to the versatility of the CoCo's display. However, I suggest you wire the three pins using the SPDT switches and the empty hole left by each pin because, when in any graphics mode, these three pins are also used by the VDG to control which graphics mode you are in.

"You no longer need to add a gate to do inverse video."

If you hard wire the pins into a particular mode, you will loose certain graphics modes, depending on which pin you hard wired. If you use the most versatile way for each switch, all you have to do to return to the normal or default mode when you need a certain graphics mode is to throw a few switches.

Next month, I'll get into a step-by-step description of how to integrate the new MC6847T1 chip into your older non-'B' CoCos. I wonder just how many original CoCos are still out there? I would like to thank James R. Igou of Newark, Delaware for supplying me with the manual and an MC6847T1 chip to work with. I would also like to thank Bill Warnica of Barrie, Ontario, for his assistance with this and the next article on the new VDG chip.

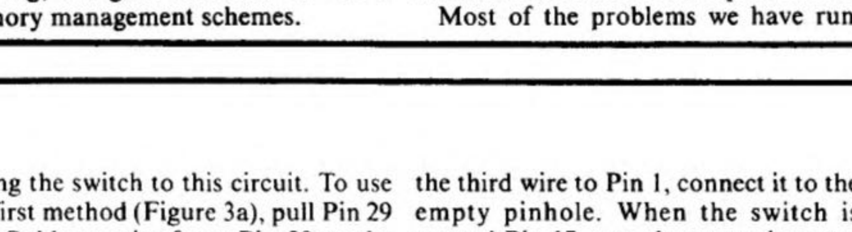


Figure 3: Inverse Video Switching

manufacturing process on track.

If you write all of your OS-9 programs in assembly language, you can get by with as little as 24K of workspace. Higher level languages like BASIC09 require at least 40K. Essentially, OS-9 Level I was designed for use on computers being used by one person. Most Level I machines contain 4K of ROM and 60K of RAM. The Color Computer uses 64K of RAM. It gets the information that is normally stored in ROM from Track 34 of an OS-9 boot disk. Level I machines can only address 64K of memory.

OS-9 Level II computers use memory management hardware that allows the 6809 microprocessor to address more than 64K of memory. Most of them use a chip called a DAT (Dynamic Address Translator). This chip moves memory in and out of the 64K address space used by the 6809. Most DAT chips switch 4K blocks of memory in and out of the 6809's workspace. In the past several years however, several large scale integration (LSI) chips have been released. These chips often switch the memory in and out of the 6809's 64K block in 2K increments.

The random access memory in your Color Computer can hold either data or programs. If you could peek into your computer's memory while it is running, you would see the names of a number of modules at the top of the 6809's 64K workspace. At the bottom of the workspace you would see a lot of temporary data being used by the programs stored in those modules. In the middle, you would find a bit of free memory to run additional programs.

When you load a new OS-9 program module it is placed at the top of the available memory space. When you run that program, it will use the first memory at the bottom of the available memory space. The amount of memory required by each program is stored in the program's module header.

If you have worked with computers for a while, you have probably come to realize that you can never have too much memory. OS-9 designers knew this and threw in a lot of features to help manage this important resource. OS-9 requires that all programs be reentrant. A program that is reentrant can be used by more than one person or process at the same time.

For example, two users may want to run a BASIC09 program at the same time. Many older operating systems would require that two copies of BASIC09 be loaded into memory to make it possible. But, since BASIC09 is reentrant, both users can use the same copy of it. In this example, we have saved more than 22K of memory — a large chunk in a 64K computer.

Despite OS-9's built-in memory saving features, we have all run into a problem called memory fragmentation.

On an OS-9 Level I computer, fragmentation can be a serious problem. On Level II computers the problem goes away — almost.

Memory fragmentation becomes a problem when the available free memory is broken up into so many little pieces that OS-9 can't find enough memory in one contiguous block to load another program module or assign data memory to a running process. A process, by the way, is a program that is running.

An operating system that permits more than one program to run at the same time needs a way to divide the system's memory between programs. Earlier operating systems like CP/M, FLEX and PC-DOS didn't bother to manage their memory; they didn't allow more than one program to run at a time.

OS-9 Level I uses a first-fit allocation scheme to manage its memory. This means that when you attempt to load a program module or run a program, OS-9 assigns the first block of memory big enough to hold your module or meet the data requirement of your program. It assigns as much of this first free block of memory as the module needs and remembers that the rest of the block is available as a smaller block. The main disadvantage of this approach is it uses big blocks of memory and leaves a lot of small blocks that can only be used to hold small program modules or satisfy small memory requests.

If you want to watch OS-9's memory allocation in action on your Color Computer you can use the Mdir, Mfree and Sleep utilities to study the process. Start by experimenting with the example given on pages 302 to 306 of *The Complete Rainbow Guide to OS-9.*

The only way to de-fragment memory is to kill some of the processes running so they release the memory they are using. After killing them, you can restart them. When they are restarted, they will be assigned memory at both ends of the available memory space, leaving a larger chunk of memory free in the middle.

Memory fragmentation becomes a problem when the available free memory is broken up into so many little pieces that OS-9 can't find enough memory in one contiguous block to load another program module or assign data memory to a running process. A process, by the way, is a program that is running.

Fragmentation can take place in the data storage area at the bottom of available memory or in the module storage area at the top of memory. You will most often run into fragmentation in the data memory area when starting a lot of processes. This happens because each process has been assigned some space for data. If you terminate a medium-aged process first you will wind up with a small chunk of available memory in between the data memory used by the oldest and youngest running processes.

"Memory fragmentation is caused by changing memory demands."

As you will see when you experience the long-awaited new Color Computer, life is much easier with OS-9 Level II. And the most important advantage revolves around the way Level II systems manage their memory. Level II systems use Dynamic Address Translation hardware that gives the system a way to use lots of memory — even though the 6809 microprocessor can only address 64K.

OS-9 Level II lets each process run in its own 64K memory address space, isolated from all other processes that may be running on the system. This means that if you want to run a 4K sort program, you could request up to 60K of data memory for that process. In other words, the program module and the data area it uses must fit in a 64K space. The OS-9 system code which includes all device descriptors and drivers, file managers, etc., is running in its own 64K space independent of the workspace you are using. The end result for the average user is that OS-9 Level II will appear to be much easier to use.

A Gold Mine of Helpful Tips

We all seem to run into these same problems at one point or other in our OS-9 career. For example, Fred Swatelle of Huntsville, Texas recently sent us a number of interesting observations and some tips that are a real gold mine for beginners. We featured his sound-generating programs last April.

Swatelle wanted to save space on the system disk he uses with the OS-9 assembler so he used his editor to trim down the files in the Defs directory. Remember, if you try these tricks you should work only with a copy of the original system disk.

Before Swatelle edited the files on his disk, he listed the files to the printer so he would have a hard copy to work from. To do this he had to delete the OPT -L directives in the files. Additionally, he had to add the following lines at the beginning of the Rbfdefs and Scfdefs files.

```
ifp1
use /d0/defs/os9defs
endc
```

After you have followed the example, assemble the three files and redirect the listing to your printer.

```
OS9: asm/d0/defs/os9defs
 L >/p <CR>
```

Then, using the printed listing and a good editor, you can remove all comments from the code, as well as any unnecessary assembler directives. For example, Pag directives and blank lines may both be deleted. A text editor which displays the carriage returns is the best tool for the job because it makes it easy to delete strings of blank spaces. After you have finished, keep these bare bones files in your Defs directory and keep the hard copy listing as a reference manual. And if you have a few moments of spare time, use it to study the OS9Defs files. You will really be surprised at the information in these files.

Here's an example of one man's improvement being another man's obstacle. Swatelle tried out the new OS-9 Version 2.00.00 Dump utility and decided he didn't like it. The new version automatically configures the format of its output to the column width stored in the device descriptor being used. For example, if you type:

```
OS9: dump >/p <CR>
```

OS-9 checks the device descriptor, /p, and learns that your printer is 80 columns wide. It then formats its output accordingly. However, if you redirect the output of the Dump utility to a file, you will find it stored on your disk in the old 32-column format.

Swatelle wanted to be able to use the manual width control parameters available in the original Dump, so he deleted the new Dump and copied the old version from his Version 1.01 disk to his system disk.

If you own *DeskMate* but have not yet purchased your copy of OS-9 Version 2.00.00, you can use the modules in the *DeskMate* system disk with the OS9Gen utility command from your original Version 1.01 system disks to make new system disks that use the Version 2.00.00 kernel. This will give you some of the new features like repeating keys, etc.

If you do have your copy of OS-9 Version 2.00, here's another tip from Swatelle. He says that after he modified the modules that came with Version 2.00.00 to match his hardware, he saved them in place of the original copy in the Modules directory of his backup copy of the Config disk. For example, after using Xmode and TunePort on the device descriptor for the printer device descriptor, /p, Swatelle deleted the copy on his working Config disk and saved the new version in a file named p.dd.

Again, remember you must only make these changes on a working copy of your Config disk. In fact, you should always make a working copy of all your OS-9 software immediately and store the original, unmodified disks in a safe place — just in case something happens to your working copy.

If you just moved to OS-9 and have not yet purchased an OS-9 disassembler, never fear. If you have been using Disk BASIC for a while and own Roger Schrag's *Super-Patched EDTASM*, you can use it to disassemble OS-9 code too. First, boot OS-9 and load the modules you want to disassemble. Then, run the Mdir e utility to find where they are stored in memory. Finally, without turning off your Color Computer, run *SPEDTASM*.

You will need to do some translation manually, since this Disk BASIC based disassembler won't recognize that an SWI2 interrupt is an OS-9 system call. You'll also need to look up the value of the byte following each SWI2 in the OS-9 technical information manual to find out which system call it is.

If you have swapped system disks or changed execution directories since you first booted OS-9 and you want to return to the original system disk booted from, just use the CLEAR/BREAK keys to terminate the current Shell. When you do this, SysGo starts a new Shell that uses the original execution and data directories.

Parallel Driver Patch for Disto

If you are using one of the earlier versions of the parallel driver for Tony DiStefano's fantastic Disto PPrint hardware and have an older printer, you may be wondering about the Device Not Ready Error that keeps popping up. The delay loop that waits for the printer to signal it is ready to accept more characters did not allow enough time for some of the slower hardwares.

To solve the problem, run the OS-9 Debug utility and execute the following steps:

```
LParallel
.<SPACEBAR>.+4C
=20
Q
```

The original value at an offset of 4C Hex from the beginning of the Parallel module is 26, the Hex code for Branch If Not Equal (BNE). This patch changes it to a 20, the code for Branch Always (BRA). This causes the driver to skip the Device Not Ready trap. Be cautioned however, that it will cause your system to wait for the printer forever if it happens to be offline. After you make the patch above you can save the module Parallel into a temporary disk file and then verify if into a permanent file using the Verify utility's update CRC parameter.

```
OS9: save temp.Parallel
 Parallel
OS9: verify <temp.Para
 llel>Fixed.Parallel u
```

After making the patch and executing the two command lines you will be able to load the new drivers. You could also OS9Gen the Fixed.Parallel driver into your OS-9 boot file if you want to take the time.

And, from what I see, DiStefano is still at it — designing better hardware for our Color Computers. The latest idea is the queue, a keyboard adapter that will convert the standard CoCo keyboard into parallel ASCII. The device will be buffered and interrupt driven, and will support auto-repeat.

Multi-Tasking in Action

If you use a computer at work that lets you use desk accessories, you'll enjoy this tip from Pete Lyall. He runs *DynaStar* concurrently with the *XCom9* terminal program available from the OS-9 Users Group Software Library, or its author Greg Morse.

"Because of XCom's size — 5K for program, 2K for data — I am able to use *DynaStar* to edit a file while still online with *XCom9*. Try that with CoCo DOS!" Lyall said. "*XCom9* is a simple, no fancy stuff, freeware terminal. If accustomed to a terminal program that 'takes over' your system and gives you menu control, it may seem a little sparse at first. But once you get used to it, you will appreciate the fact that it is designed not to interfere with other programs running concurrently. It allows file capture and transmission as well as Xmodem file transfer."

Jonathan Cluts, a former Tandy employee, added that he had run Sled, a full-screen editor in memory with *XCom9*. "I have also downloaded a file, called up a new Shell, started that file printing and then gone on to download another file," Cluts said.

Congratulations

Congratulations are in order for Wayne Day, president of Golden Triangle Corporation and fellow RAINBOW author. Day recently formed the Tandy Users Network (TandyNet) to serve the full line of Tandy microcomputers. He has operated The Color SIG on CompuServe since its beginning in 1982. The new network takes the place of four existing Tandy SIGs that had been operated by individual managers scattered across the nation. TandyNet will allow the individual SIGs to share information, improving the support for Tandy computer users.

"Over the years, the forums have become a gigantic users group that holds meetings 24 hours a day," he said. "We have taken the idea one step further and created a blanket group — the Tandy Users Network."

A Tip of the Hat to Tim Harris

Tim Harris, who has contributed several programs to this column and *The Complete Rainbow Guide to OS-9*, was published in the May issue of *Dr. Dobb's Journal*. Harris took an earlier article in that publication to task.

"One of the most novel features added in Version 2 of MS-DOS is the concept of 'installable device drivers,'" the article said. Color Computer owners have been using this "new concept" for at least three or four years! But, let's quote Harris:

"I would like to say that this concept may be new and novel for Microsoft and MS-DOS but it is certainly not a new and novel concept for other operating systems. The initial 6809 OS-9 Level I released in 1978 sported this feature," Harris told *Dr. Dobb's Journal* readers. Good job Tim. That's the kind of evangelism we need. If we tell them often enough, they are bound to stop and pay attention.

Speaking of Evangelism

You can become an OS-9 evangelist by writing letters like the one Tim Harris sent to *Dr. Dobb's Journal*. But even if you are writing letters like this and telling all your friends about OS-9, please don't forget to cover another very important audience — those people already using OS-9.

Are we covering this base? I don't think so because we don't have everything we need. If OS-9 is to succeed in the consumer marketplace it must have a reason for being. There must be hundreds of application programs to do the jobs that people buy computers to do.

To make this happen we must encourage the programmers already within our ranks. We must salute them when they move the state of the art forward. We must encourage them to make bold steps forward with innovative techniques, rather than discourage them with our criticism.

I say these things after reading the mail on both CompuServe's OS-9 SIG and on RAINBOW's Delphi CoCo SIG. There are a lot of helpful people using both of these electronic bulletin boards, but there is also too much criticism. This criticism, especially when combined with low sales and minimal profit, discourages programmers from writing new programs. As a result, we all suffer.

For example, people criticize Tandy for making a business decision when they designed their OS-9 implementation. Then, they criticize Microware for delivering the product the customer ordered. They don't stop to realize one important basic of the business world — the customer is always right.

Frankly, Tandy had a good reason for every feature they put in Color Computer OS-9. We may or may not agree with the selection made by Tandy's designers, but we must realize that these business decisions were not made lightly. And to be quite honest, we must

Listing 1: *gotoxy*

```
     nam gotoxy.adm3

     ttl DynaStar editor XY routine for Disto 8Ø Column Card

    *
    *  Allan G. Jost  January 1Ø, 1982
    *
      use /HØ/DEFS/Defsfile

      ttl DynaStar XY routine for CoCo
      opt g
      org Ø

    EndMem equ .  no data space for a subroutine
    Vers equ 2  version number
      mod EndMod,Name,Sbrtn+Objct,Reent+Vers,Entry,EndMem
    Name fcs "gotoxy"
      fcb Vers
    Entry bra Go
      fcb 24  number of lines on terminal
      fcb 8Ø  number of characters per line
      fcb 1   This terminal scrolls
      fcb 1   length of Clear Line Sequence
      fcb 4   byte that clears line
      fcb Ø   no initialization sequence

    *  on entry X contains X-coordinate: 1 .. 8Ø
    *           B contains Y-coordinate: 1 .. 24

    Go equu *  actual entry point
      leas -3,S  make working space
      addb #31 change Y to cursor control char
      stb 2,S  and put into work space
      tfr X,D  get the X coordinate
      addb #31 and change it also
      stb 1,S
      ldb #2  xy-cursorcode
      stb Ø,S  finish building work space
      lda #1  standard output path
      leax Ø,S  the escape sequence
      ldy #3   and its length
      os9 i$write put it out to console
      leas 3,S  restore stack by releasing work space
      rts  back to caller now, folks
      emod
    EndMod equ *  this is the end, folks.
```

Listing 2: *HGraph.c*

```
     /*

       HGRAPH - Horizontal Bar Graph Program
                  by Milt Webb

    This program demonstrates the use of structures
    and sequential disk files containing mixed types.
    Create a bar graph with up to 16 bars by entering
    the title, subtitle, scale (range) of the graph
    and the label and value of each bar. The program
    is menu driven and the graph files are read/saved
    in the current data directory. This program is written
    for 8Ø column displays. The #defines may be altered
    accordingly for hi-res displays.

     */

     #include <stdio.h>
    #include <ctype.h>
    #define TRUE 1
    #define FALSE Ø
    #define TERMWID 8Ø  /* width of terminal screen */
```

also realize that if Microware hadn't been willing to deliver the product Tandy wanted, Tandy probably would have picked another operating system for the Color Computer. Then where would we be?

Everyone deserves a pat on the back once in awhile — especially when he is not getting rich in a market he is supporting out of pure love. If this positive attitude theory seems like a lot of hogwash to you, I challenge you to look around the business world. I think you'll find that companies that project a positive, can-do attitude to the public and to their own employees are the ones getting ahead. Organizations run by managers who try to think of reasons not to do something are falling like flies.

The bottom line: If you know a better way to do something, do it. If it's something everyone can use, sell it. If it's not, share it. But, do it. And, please don't put the other guy down just because his approach is a little different.

Sell Your Program in Japan

Ark Corporation is interested in good applications to market in Japan. They report that the Fujitsu FM-11, an OS-9 Level II machine, is the most popular in Japan. The company is also introducing three types of plug-in OS-9 68K boards for the leading personal computers in Japan.

"The biggest and most well-known disadvantage of OS-9 when compared to other systems is its lack of application programs," says Ark's Vice-President Hirokazu Sugawara. "Thus, we are looking for good OS-9 programs to introduce in Japan while developing our own. We need good programs for business, communications, database management, entertainment, home accounting, programming and word processing."

If they like your program, Ark will grant you an exclusive distribution license in the Japanese market, prepare a Japanese operating manual and make any modifications needed to make it fit the Japanese market. They pay 10 to 20 percent of the program's retail value in royalties. Their FAX number is 03-350-8383. Their phone number is 03-350-5171. If you have a good program, go for it!

Bob Rosen called my attention to the fact that OS-9 has made *Byte* magazine again — this time in a brief report from Comdex. *Byte* reported that Microtrends of Schaumburg, Ill., has introduced versions of OS-9 for the Amiga, Atari ST and Macintosh. The report also mentioned compact disk interactive and noted that OS-9 "is similar to UNIX but smaller and less complex."

```
#define MAXITEMS 16 /* size array of items to graph */
#define TITLEN 41   /* length of title, subtitle string +1 */
#define LBLEN 19    /* length of label strings +1 */
#define NAMLEN 12   /* length of filename */
#define STOP ""     /* empty string */
#define CLEARS 2    /* clear screen, home cursor for wordpak */

char h1[] = "HORIZONTAL BAR GRAPH v1.1";
char h2[] = "by Milt Webb";
long i, count, j, points;
long GWIDTH = (TERMWID - LBLEN - 1);

struct param {
    char title[TITLEN];
    char subtitle[TITLEN];
    long upper;
    long lower;
    long count;
    } header;

struct data {
    char label[LBLEN];
    long value;
    } bar[MAXITEMS];
 main()
{

  int k,r;
  pflinit();  /* need this to print long integers */
  while(1)    /* make menu repeat until Q is hit */

     {
     putchar(CLEARS);  /* clear display and home cursor */
     printf("\n\n");
     center(h1);
     center(h2);
     printf("\n");
     printborder();
     printf("\n\n\n");
     center("Type 'L' to LOAD graph file.");
     printf("\n");
     center("Type 'C' to CREATE a new graph.");
     printf("\n");
     center("Type 'Q' to QUIT program.");
     printf("\n\n\n");
     printborder();
     printf("Selection: ");
     k=toupper(getchar());
     switch(k)
          {
          case 'L':
               if ( r = (readgraf() == TRUE) )   dograf();
               printf("Press ENTER to return to menu.");
               k=getchar();
               break;

          case 'C':
               askinfo();
               askitems();
               dograf();
               if ( r = (savgraf() == TRUE) )
                    {
                    printf("Press ENTER to return to menu.");
                    k=getchar();
                    }
               break;

          case 'Q':
               exit(Ø);
               break;

          default:
               break;
          } /* end switch */
     } /* end while */
} /* end main */


 askinfo()
/* get title,subtitle & range of graph */
  {
  while (getchar() != '\n');  /* purge input buffer */
  printf("Title for graph: ");
  gets(header.title);
  printf("Subtitle: ");
  gets(header.subtitle);
  printf("Entér the upper range for this graph: ");
    scanf("%ld", &header.upper);
  printf("Enter the lower range for this graph: ");
    scanf("%ld", &header.lower);
    while (getchar() != '\n');  /* purge input buffer */
  }


 askitems()
/* Get labels and data values for graph */

{
putchar(CLEARS);
printf("Enter up to %d items for this graph.\n", MAXITEMS);
printf("Maximum label length is %d characters.\n",LBLEN-1);
```

An Assembly Language Tip

John Bowden, a Navy cryptologic technician stationed in Adak, Alaska, wrote us recently and asked how to run another OS-9 command from within an assembly language program.

"My quest started with the simple desire to clear my text screen in OS-9 without typing the cumbersome Display C," Bowden said. "At first I used a simple procedure file that ran the command line Display C when I typed CLS. That was fine but it took a lot of time because of the extensive disk I/O. What I really would like to do is implement the Display C command in assembly language."

There are two ways to clear the Color Computer screen from within an OS-9 assembly language program. A short assembly language program that sends the clear screen character, 12 decimal or $0C Hex, to the standard output would be the most direct route. We'll show you the code to do that first, and then list a short segment of code to run your program from within another program. We'll finish with a piece of code to let you execute the display command with the character 12 decimal as a parameter.

```
* This program will clear the screen on
  your Color Computer
* Syntax: cls <ENTER>

nam cls
* Use standard OS-9 Defsfiles

ifp1
use /DØ/DEFS/defsfile
endc

opt L
ttl Clear Screen Utility

* Use standard OS-9 module header

mod clssiz,clsnam,type,revs,start,size
clsnamfcs /cls/
typeset PRGRM+OBJCT
revsset REENT+1

* Data Memory Area Defined here

clrchrrmb 1

* Reserve room for stack
rmb 25Ø
sizeequ .

* Actual code starts here

startlda #$ØCclear screen character
sta clrchr   store it in data area
leax clrchr,upoint to character
ldy #1we want to send one character
lda #1to the standard output path
os9 i$writego send it
clrbclear carry
os9 f$exitand exit
emodmark the end of the module
clssizequ *
end
```

This short assembly lanaguage program sends the character 12 decimal to OS-9's standard output path. On a Color Computer this character clears the screen. In fact, most printers use the same character for a form feed, so you can redirect the output of the new command to start a new page on the printer.

```
OS9: cls>/p
```

```
printf("Press <ENTER> at a label prompt when finished.\n");
printborder();
printf("\n");

header.count = Ø;
/* get things started */
printf("Enter label for item %ld: ", header.count+1);

while ( header.count < MAXITEMS
   && strcmp(gets(bar[header.count].label),STOP) != Ø )
   {
   printf("Now, the data for > %s: ", bar[header.count].label );
   scanf("%ld", &bar[header.count++].value);
   printf("\n");
   while (getchar() != '\n');  /* purge input buffer */
   printf("Enter label for item %ld: ", header.count+1);
   }
 }

dograf()
  /* display the graph */
  {
  putchar(CLEARS);
  center(header.title);
  center(header.subtitle);
  printf("\n\n");
  printf("%2Øld%6Øld",header.lower,header.upper);
  printborder();
  for ( i=Ø ; i<header.count ; i++)
    {
    printf("%-18s|", bar[i].label);
    points = ((bar[i].value -header.lower) * GWIDTH)
             / (header.upper -header.lower);

    if ( bar[i].value < header.lower )
      putchar('<');
        else if ( points < 1 )
          points = 1;       /* need at least one point */
    else if ( points > GWIDTH )
      points = GWIDTH;

    for ( j=Ø ; j<points ; j++ )
      putchar('*');

    if (bar[i].value > header.upper)
      putchar('>');
    else printf("\n");
    }
  printborder();
  }

readgraf()
/* read data from graph file - current data directory */
   {
   int c;
   int errflg=TRUE;

   FILE *filptr;
   char filename[NAMLEN];

   while (getchar() != '\n');  /* purge buffer */
   printf("Enter filename for graph to view: ");
   gets(filename);

   if ( (filptr=fopen(filename,"r")) == NULL )
       {
       printf("Sorry, cannot open %s.\n", filename);
       errflg=FALSE;
       }
   else
      {
      fread(&header,sizeof(header),1,filptr);
      fread(&bar[Ø],sizeof(bar),1,filptr);
      fclose(filptr);
      } /* end else */

   return errflg;
   } /* end func */

  savgraf()
 /* save graph data in current data directory */
    {
    int c;
    int errflg=TRUE;
    FILE *filptr;
    char filename[NAMLEN];

    printf("Enter filename for this graph: ");
    gets(filename);

    if ( (filptr=fopen(filename,"w") ) == NULL)
          {
          printf("Sorry, cannot open %s.\n",filename);
          errflg=FALSE;
          }

    else
       {
       fwrite(&header,sizeof(header),1,filptr);
       fwrite(&bar[Ø],sizeof(bar),1,filptr);
       fclose(filptr);
```

To run your Cls program from within another assembly language program you can insert the following code in your other program.

```
* Execute cls utility command
* First define the strings

shlstrfcs /shell/
cmdstrfcc /cls/
fcb 13<RETURN> character

leax shlstr,pcrpoint to "shell"
ldy #4size of parameter string
leau cmdstr,pcrpoint U-register to "cls"
lda #lit's 68Ø9 object code
clrboptional data area size
os9 f$forkgo start the cls as a process
bcs error
os9 f$waitand wait for it to finish

* resume other program execution
```

You could modify this code to run the OS-9 Display utility command like this:

```
* Execute display utility command
* With "C" as a parameter
* Define the strings

shlstrfcs /shell/
cmdstrfcc /display c/
fcb 13<RETURN> character

leax shlstr,pcrpoint to "shell"
ldy #1Øsize of parameter string
leau cmdstr,pcrpoint U-register to "cls"
lda #lit's 68Ø9 object code
clrboptional data area size
os9 f$forkgo start the cls as a process
bcs error
os9 f$waitand wait for it to finish

* resume other program execution
```

Study the differences between the two code segments and you'll quickly pick up the idea. If you put code like this in your assembly language programs, it is up to you to insure that the programs running from within those programs are actually loaded in memory or available in the current execution directory. Have fun!

RS-232 Tip

For something that is supposed to be simple, serial communication between two computers is often quite confusing. We get a lot of letters from people trying to use their Color Computers with other computers. In fact, I had trouble when I tried to fire up my RS-232 Pak the first time.

I could get my terminal programs to run perfectly when I plugged the RS-232 Pak into a modem. But every time I tried to communicate with another computer I was stopped at the pass. I could send, but I couldn't receive — even though I had made the connection through a null modem cable.

The problem revolves around the fact that the RS-232 Pak wants to see a carrier signal from the modem before it works properly. My solution was to short Pin 20, the data terminal ready signal from the RS-232 Pak to Pin 8, the carrier detect line. I made this connection on the end of the cable that plugged into the RS-232 Pak. By cheating like this I was essentially telling the RS-232 Pak that there was always a carrier.

```
      } /* end else */

   return errflg;
   } /* end func */

center(string)
char *string;
/* print argument centered on display */
   {
   int spaces, num;
spaces = (TERMWID - strlen(string)) / 2;
for (num = 1 ; num <= spaces ; num++)
        putchar(' ');
printf("%s\n",string);

   }


printborder()
/* print a border row of '=' signs */

  {
  int num;
  for ( num=Ø ; num<TERMWID ; num++ )
    putchar('=');
  }
```

Listing 3: *cls*

```
PROCEDURE cls
DIM hp:BYTE
DIM name:STRING[3]
name:="/hi"
OPEN #hp,name:WRITE
PRINT #hp,CHR$(12);
CLOSE #hp
END
```

Listing 4: *printat*

```
PROCEDURE printat
PARAM col,row:INTEGER
DIM hp:BYTE
DIM name:STRING[3]
name:="/hi"
OPEN #hp,name:WRITE
PRINT #hp,CHR$(2); CHR$(col+32); CHR$(row+32);
CLOSE #hp
END
```

Listing 5: *toggle*

```
PROCEDURE toggle
DIM name:STRING[3]
DIM hp:BYTE
name:="/hi"
OPEN #hp,name:WRITE
PRINT #hp,CHR$(2Ø)
CLOSE #hp
END
```

Listing 6: *box*

```
PROCEDURE box
(* Calling syntax is : *)
(* RUN BOX (hstart,vstart,hend,vend)   *)
PARAM hstart,vstart,hend,vend:INTEGER
RUN gfx("line",hstart,vstart,hend,vstart)
RUN gfx("line",hstart,vstart,hstart,vend)
RUN gfx("line",hend,vstart,hend,vend)
RUN gfx("line",hend,vend,hstart,vend)
END
```

Listing 7: *fillbox*

```
PROCEDURE fillbox
PARAM hstart,vstart,hend,vend:INTEGER
DIM linecount,counter:INTEGER
linecount:=vend-vstart
FOR counter:=Ø TO linecount-1
RUN gfx("line",hstart,vstart+counter,hend,vstart+counter)
NEXT counter
END
```

Other people cheat their connections to the RS-232 Pak in a similar manner. For example, one programmer I know always creates a false carrier by jumpering pins 6, 8 and 20 on both ends of his cable.

To make your second computer look like a modem you also need to connect Pin 2 on one end of your cable to Pin 3 on the other and vice versa. Additionally you must short Pin 4 to Pin 5 on both ends of the cable and make sure that Pin 7 is passed through from one end of the cable to the other. The only disadvantage to shorting these control pins is your terminal will not be able to send a hardwired signal to your Color Computer to tell it to stop sending if it happens to get behind.

This Month's Listings

We've received several notes from people who have upgraded their hardware and don't have the proper GoToXY routine to work with *DynaStar* and *DynaSpell*. Our first listing this month will let you use your Disto 80-column card with these programs. You can modify the listing to work with the standard Color Computer screen in Version 2.00.00 by simply changing the size definitions.

Our next listing is a C program named *HGraph.c* from Milt Webb. *HGraph.c* creates horizontal bar graphs on an 80-column screen, demonstrates how to use a menu within a C program and shows you how to save and read sequential files containing mixed data types. It is Webb's first program.

Our final listings were contributed by Robert B. Stephens. He uses BASIC09 with the Xscreen package from Microtech Consultants, Inc. He displays all his text on the Xscreen device, /hi. If you are using a different screen, you can use similar code by just changing the name of the device. I tried both *Cls* and *Printat* with the standard Tandy 32-column display and they worked just fine.

"I wrote a short procedure called *Pixsaver* to save graphics screens," Stephens said. "Picture files are stored in a directory named PIX. To view the pictures you use a procedure named *Pixshow*. Another procedure named *Toggle* lets you inverse letters with Xscreen. The latter may be used to highlight single words or inverse the entire display."

That's it for October. Hopefully, by the time I sit down to write the November column we will have at least seen the new OS-9 Level II Color Computer. In any case we'll be attempting to round up more tips for all OS-9 users. □

Listing 8: *pixsaver*

```
PROCEDURE pixsaver
DIM vdisplay,number:INTEGER
DIM title:STRING[1Ø]
DIM pixpath,pixbyte,hp:BYTE
DIM name:STRING[3]
name:="/hi"
OPEN #hp,name:WRITE
RUN printat(Ø,Ø)
INPUT "Picture title?",title
CREATE #pixpath,"/DØ/pix/"+title:WRITE
RUN gfx("Gloc",vdisplay)
RUN printat(Ø,Ø)
PRINT #hp,"                              "
 FOR number:=Ø TO 6143
 pixbyte:=PEEK(vdisplay+number)
 PUT #pixpah,pixbite
 NEXT number
 CLOSE #pixpah
 CLOSE #hp
 END
```

Listing 9: *pixshow*

```
PROCEDURE pixshow
(* If you are not using a hi-res display, you *)
(* must add gfx("mode") statement to this procedure. *)
DIM pixbite,hp:BYTE
DIM px:BYTE
DIM name:STRING[3]
DIM vdisplay,number:INTEGER
name:="/hi"
OPEN #hp,name:WRITE
PRINT #hp,CHR$(12)
RUN printat(Ø,Ø)
PRINT #hp USING "s64^","PixShow : See Pix directory for file names."
PRINT #hp USING "s64^","Filename";
RUN gfx("Gloc",vdisplay)
RUN printat(28,13)
INPUT file$
OPEN #px,"/dØ/pix/"+file$:READ
SEEK #px,Ø
FOR number:=Ø TO 6143
GET #px,pixbite
POKE vdisplay+number,pixbite
NEXT number
CLOSE #hp
CLOSE #px
END
```

Listing 10: *calc*

```
PROCEDURE calc
DIM a,b,c,d,e,f,g,h,i,j,k,l,m,n:REAL
DIM o,p,q,r,s,t,u,v,w,x,y,z:REAL
DIM pp:BYTE
DIM sp:BYTE
DIM name:STRING[2]
name:="/p"
PRINT CHR$(12)
PRINT "C A L C U L A T O R ... All Basic Math Functions work!"
PRINT "Variables are letters a-z -- assign with let a=xxx"
PRINT "The printer path is #pp -- Send text or variables there at will."
PRINT "If you want anything hardcopy -- don't forget to turn on your printer."
INPUT " ... Printer on? (y/n) ",yesno$
IF yesno$="y" THEN OPEN #pp,name:WRITE
ELSE PRINT
ENDIF
INPUT "Do you want to save some of this stuff? (y/n) ",query$
IF query$="y" THEN
PRINT "File name is ScratchPad: Send data there at will."
PRINT "Syntax is PRINT #sp, <text>, <mathfunction(variable)>"
OPEN #sp,"ScrtchPad":UPDATE
ELSE PRINT
ENDIF
PRINT "Type <cont> <ENTER> to close paths."
PAUSE
IF yesno$="y" THEN
PRINT "Printner path closed."
CLOSE #pp
ENDIF
IF query$="y" THEN
PRINT "Scratch Pad closed."
CLOSE #sp
ENDIF
END
```

Listing 11: *make_scratchpad*

```
PROCEDURE Make_ScratchPad
DIM Scratch_Pad:BYTE
CREATE #Scratch_Pad,"ScratchPad":UPDATE
PRINT "The ScratchPad file for Calculator has been created."
CLOSE #Scratch_Pad
END
```

WHAT'S ON THE BEST OF CoCoOz

Best of CoCoOz #1. **EDUCATION**

ROADQUIZ	ROB WEBB
HANGMAN	ALEPH DELTA
AUSTGEOG	P. THOMAS
SPELL	IAN LOBLEY
FRACTUT	ROBBIE DALZELL
ICOSA	BOB WALTERS
TAXMAN	TONY PARFITT
MARKET	ALEPH DELTA
TOWNQUIZ	ROB WEBB
ALFABETA	RON WEBB
TANK ADDITION	DEAN HODGSON
TABLES	BARRIE GERRAND
KIDSTUFF	JOHANNA VAGG
FLAGQUIZ	ROB WEBB

Best of CoCoOz #2 part 1. **16K GAMES.**

LE-PAS	Wrongsoft
COCOMIND	STEVE COLEMAN
OILSLICK	JEREMY GANS
CCMETEOR	BOB THOMSON
BATTACK	JEREMY GANS
PROBDICE	BOB DELBOURGO
CHECKERS	J & J GANS
PYTHON	?
POKERMCH	GRAHAM & MATTHEWS
SPEEDMATH	DEAN HODGSON
LNDATTCK	ALDO DEBERNARDIS
INVADERS	DEAN HODGSON
RALLY	TONY PARFITT
FOURDRAW	JOHANNA VAGG

Best of CoCoOz #2 part2. **32K GAMES.**

TREASURE	DAVISON & GANS
MASTERMIND	GRAHAM JORDAN
ANESTHESIA	MIKE MARTYN
OREGON TRAIL	DEAN HODGSON
ADVENTURE	STUART RAYNER
SHOOTING GALLERY	TOM DYKEMA
GARDEN	DAVE BLUHDORN
YAHTZEE	KEVIN GOWAN
BATTLESHIP	CHRIS SIMPSON
ANDROMIDA	MAX BETTRIDGE

Best of CoCoOz #3. **UTILITIES**

PAGER	?
HI	ALEX. HARTMANN
SPOOL64K	WARREN WARNE
CREATITL	BRIAN FERGUSON
FASTEXT	OZ-WIZ
DATAGEN	ROBIN BROWN
SPEEDCTR	PAUL HUMPHREYS
PRNTSORT	PAUL HUMPHREYS
BIGREMS	BOB T
DIR	PAUL HUMPHREYS
COPYDIR	THOMAS SZULCHA
LABELLER	J.D.RAY
SCRPRT	TOM DYKEMA
MONITOR	BRIAN FERGUSON
BEAUTY	BOB T
PCOPY	B. DOUGAN
RAMTEST	TOM DYKLEMA
DISKFILE	B. DOUGAN
LABEL	F. BISSELING

Best of CoCoOz #4. **BUSINESS.**

HI	ALEX. HARTMANN
(Disk Directory manager)	
BANKSTAT	BARRY HATTAM
(Statement annal & store)	
INSURE	ROY VANDERSTEEN
(Analyse home contents)	
SPOOL64K	WARREN WARNE
(Printer spooler req 64K)	
2BC	WARREN WARNE
(Hold 2 sep progs in mem)	
DATABASE	PAUL HUMPHREYS
(THE tape database)	
RESTACC	DUNG LY
(Tape restaurant accounts)	
PRSPDSHT	GRAHAM MORPHETT
(Disk print out SPDSHEET)	
PERSMAN	PAUL HUMPHREYS
(Personal finance management)	
CC5	GRAHAM MORPHETT
(Sales Invoicing-tape sys)	
COCOFILE	BRIAN DOUGAN
(Tape data base)	
DPMS	PAUL HUMPHREYS
(Disk Program Management Sys)	
40KGREY	RAY GAUVREAU
(40K Basic for grey 64K CoCo)	
TAXATION	?
(Calc tax payable)	
SPDSHEET	GRAHAM MORPHETT
(Disk 22 column spreadsheet)	
ACS3	GREG WILSON
(Multi disk data base)	

Best of CoCoOz #5. **ADVENTURES.**

ADV 32K	S. RAYNER
QUEST	TONY PARFITT
LABYRINT	JAMES REDMOND
ADV	SEAN LOWE
CRYSTAL	C & K SPRINGETT
PRISON	TIM ALTON
OPALTON	IAN CLARKE
WIZARD	DARRELL BERRY
TREASURE	C. DAVISON
LOST	ALEX. HARTMANN

Best of CoCoOz #6. **PRESCHOOL.**

ALPHABET	STUART DAWSON
HATDANCE	JOHANNA VAGG
AUSTSONG	McDERMOTT FAMILY
ADVANCE	McDERMOTT FAMILY
WALTZING	McDERMOTT FAMILY
TIMEKANG	McDERMOTT FAMILY
BAND	McDERMOTT FAMILY
KIDSTUFF	JOHANNA VAGG
MATCHER	?
LETTERS	JACK FINNEN
BABYSIT	JOHANNA VAGG
SPELLING	JOHANNA VAGG
SPEEDTAB	DEAN HODGSON
10 FACES	JOHANNA VAGG

Best of CoCoOz #7. **GRAFIX.**

LIL'COCO	ANDREW WHITE
THE ROOM	H. FREDRIKSON
BACK ST	JOY WALLACE
LOCO	MIKE D'ESTERRE
COCO ART	SANDY McGREGOR
KANGA	JOHANNA VAGG
THE BOAT	SANDY McGREGOR
SAD COCO	F. BOLLE
TOWER	C.A. SYMS
WINDYDAY	SARAH LAW
SAILING	STEVE YOUNGBERRY
OUTHOUSE	STEVE YOUNGBERRY
SMURF	JOHANNA VAGG
SUNSTATE	STEVE YOUNGBERRY
HELICOPT	ANDREW WHITE
MARTHA	ANDREW WHITE
BAD MOON	STEVE YOUNGBERRY
MCC	J. WALLACE
EAGLE	?
BLASTER	PAUL YOULD
FOGHORN	PAUL STEVENSON

Best of CoCoOz #8. **GAMES.**

ALIEN	STUART SANDERS
QWERL	DARRELL BERRY
TANK	CRAIG STEWART
SHOOTOUT	CRAIG STEWART
SHUTTLE	CRAIG STEWART
FROG	DARREN OTTERY
FROGRACE	TOM LEHANE
KIMMAT	TOM LEHANE
GRANDPRI	DOUG GREY
WATERWAR	JUSTIN LIPTON
CATERPIL	JUSTIN LIPTON
DETECT	VAL STEPHEN
BREAKOUT	WHY/BILT

Best of CoCoOz #9. **32K GAMES.**

TRIOMINO	BOB DELBOURGO
TALKHANG	?
MATCHEM	C. BARTLETT
GO	BOB DELBOURGO
NARZOD	MAX BETTRIDGE
CHOMPER	MAX BETTRIDGE
POPBALL	MAX BETTRIDGE
LUDO	WHY/BILT
SABRE	ANDREW SIMPSON
MOVEABOUT	KEVIN GOWAN
JIGSAW	C. BARTLETT
ROCKFALL	T.J. DAVIES

Best of CoCoOz #10. **EDUCATION2.**

METEOR	DEAN HODGSON
DRIVTEST	ANDREW SIMPSON
SALE	JUSTIN LIPTON
TABLES	PAT KERMODE
OPALTON	IAN G. CLARKE
CAPITAL LETTERS	BOB HORNE
TEST MATCH	JEFF SHEEN
SENT END	BOB HORNE
ESCAPE	DEAN HODGSON
RAILMATH	BOB HORNE
COUNTDOWN	DEAN HODGSON
WHATZIT	BOB HORNE
HOMOPHONE	BOB HORNE
COMPWORDS	BOB HORNE

TAPE $10 each **DISK $16 each**

GOLDSOFT

P.O. BOX 1742, SOUTHPORT. QLD. 4215 Phone (075) 510 015

ORDER FORM

AUSTRALIAN RAINBOW	12 mnths	$ 45.00	.....
	6 mnths	$ 27.95	.....
	1 mnth	$ 4.95	.....
AUSTRALIAN CoCo	12 mnths	$ 35.00	.....
	6 mnths	$ 21.35	.....
	1 mnth	$ 3.75	.....
CoCoOZ on tape	12 mnths	$ 75.00	.....
	6 mnths	$ 42.00	.....
	1 mnth	$ 10.00	.....
CoCoOZ on Disk	12 mnths	$118.26	.....
	6 mnths	$ 59.50	.....
	1 mnth	$ 10.95	.....
MiCoOZ on Tape	12 mnths	$ 75.00	.....
	6 mnths	$ 42.00	.....
	1 mnth	$ 10.00	.....
RAINBOW ON TAPE (AUST/US)	12 mnths	$144.00	.....
	6 mnths	$ 81.00	.....
	1 mnths	$ 15.00	.....
RAINBOW ON DISK (AUST/US)	12 mnths	$172.00	.....
	6 mnths	$ 86.00	.....
	1 mnths	$ 15.50	.....

Or charge my credit card monthly
TAPE/DISK ONLY

CoCoOZ on Tape	$ 10.00	.....
CoCoOZ on Disk	$ 10.95	.....
MiCoOZ on Tape	$ 10.00	.....
Rainbow on Tape (AUST/U.S)	$ 15.00	.....
Rainbow on Disk (AUST/U.S)	$ 15.50	.....

Additional Requirements:

....

....

Sub No: ☐☐☐☐☐☐ or ☐ **New Subscription**

Name:

Address:

P.C.

Phone No.:

Please find enclosed:-
CHQ / MONEY ORDER / NO CASH
Please charge my:-
MASTERCARD / BANKCARD / VISA
Authorised amount $ ....
Signed : ....

Goldsoft
Gift Certificate

NOW AVAILABLE !

For Christmas, Birthdays etc

Give a subscription to:—

Australian Rainbow Magazine
Australian CoCo Magazine
CoCoOz on tape or disk
Australian Rainbow on tape or disk
or The Best of CoCoOz series on tape or disk.

— IN FACT anything we sell !! —

SO ORDER NOW

GOLDSOFT

Hardware & Software for your TANDY computer.

HARDWARE

The CoCoConnection: Connect your CoCo to the real world and control robots, models, experiments, burglar alarms, water reticulation systems — most electrical things. Features two MC 6821 PIAs; provides four programmable ports; each port provides eight lines, which can be programmed as an input or output; comes complete with tutorial documentation and software; supplied with LED demonstration unit. Switchable memory addressing allows use with disk controller or other modules via a multipack interface; plugs into Cartridge Slot or Multipack, uses gold plate connectors; a MUST for the hardware designer and debugger!		$206.00
Video-Amp: Connects simply to your CoCo to drive a Colour or Mono monitor.	With instructions With instructions and sound	$25.00 $35.00
The Probe: A temperature measuring device which attaches to the joystick port of your CoCo or T1000, or to the joystick port of your CoCo Max. Comes with programs to start you thinking, and is supported monthly in Australian CoCo magazine.	 With amplifier	$39.95 $49.95

SOFTWARE

Magazines: Australian Rainbow Magazine — THE magazine for advanced CoCo users! Australian CoCo Magazine — THE magazine for the new user of a Tandy computer. Also suits owners of CoCos, MC 10s, Tandy 1000s, 100s, 200s & 2000s.	Australian Rainbow 1986	$4.95
Back Issues: Australian Rainbow Magazine. (Dec '81 to now.) **Please Note:** Some months out of stock.	1982 — 1985	$2.50
	Australian CoCo 1986	$3.75
Australian CoCo Magazine. (Aug '84 to now.) **Please Note:** Some months out of stock.	Sept 1984 — 1985	$3.00
CoCoBug Magazine. For CoCo — usually 8 programs in each magazine. (Sep '84 to Oct '85)	each	$1.00
Australian MiCo Magazine. For Tandy MC 10 computers. (Dec '83 to Jul '84)	each	$2.00
CoCoOz, on Tape or Disk: The programs you see listed in Australian CoCo Magazine are available on CoCoOz! No laborious typing — just (C)LOAD and Go!	Each Tape Subscription, 6 months 12 months	$9.50 $42.00 $75.00
Back issues of CoCoOz are always available	Each DISK Subscription on disk, 12 months	$10.95 $102.50
Rainbow on Tape, or Disk: Australian. The programs you see listed in Australian Rainbow Magazine are available on tape. A boon if you don't understand the language!	Each Tape Subscription, 12 months	$15.00 $144.00
American. We also supply the programs found in American Rainbow on tape. Please specify either Australian or American.	**NEW** for 1986 ONLY Each DISK Subscription on disk, 12 months	$15.00 $172.00
MiCoOz: The programs in the MiCo section of Australian CoCo Magazine. (For MC 10 computers only) Back issues of CoCoOz and MiCoOz are always available	Each Tape	$9.50
GOLDDISK 1000 — programs from 'softgold' for your Tandy 1000 on disk, and,		$10.95
Goldlink Goldlink is our very special service on Viatel 642# which you can access with a 1200/75 Baud modem and the appropriate software. Goldlink may be accessed at no charge, but access to our BBS on Goldlink costs 15/30c each time or $36.00 annually. Later we will also provide software for you to download, and members will be able to obtain this at no further charge or at reduced charges.	Subscription 12 months	$36.00
Books: HELP: A quick reference guide for CoCo users. BYTE: Guide for new CoCo users. MiCo HELP: A quick reference for owners of MC 10 computers.		$9.95 $4.00 $9.95
Say the Wordz: by Oz Wiz & Pixel Software Two curriculum based speller programs for your Tandy Speech/Sound Pack.	Tape 32K ECB	$29.95
Bric a Brac: Blank tapes . . . 12 for $18.00 or $1.70 each. Cassette cases . . . . 12 for $3.50 Disks . . . (they work!) . . $2.50 each or $25.00 per box of 10.		

HOW TO ORDER

Option 1: Use the subscription form in this magazine.
Option 2: Phone and have ready your Bankcard, Mastercard or Visa number.
Option 3: Leave an order on Viatel, but be sure to include your Name, Address, Phone Number, Credit Card Number and a clear indication of what you require, plus the amount of money you are authorising us to bill you.

User Group Contacts

(Stop between numbers = b.h. else
a.h.; but, hyphen between = both.)

ACT:

CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 88 9226

NSW:

SYDNEY:		
BANKSTOWN	CARL STERN	02 646 3619
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSKO MCKAY	02 624 3353
CHATSWOOD	BILL O'DONNELL	02 419 6081
CLOYTON	HERMAN FREDRICKSON	02 6236379
FAIRFIELD	ARTH PITTARD	02 72 2881
GLADESVILLE	MARK ROTHWELL	02 817 4627
HILLS DIST	ARTHUR SLADE	02 622 8940
HORNSBY	ATHALIE SMART	02 848 8830
KENTHURST	TOM STUART	02 654 2178
LEICHHARDT	STEVEN CHICOS	02 560 6207
	or GORGE ECHEGARAY	02 560 9664
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACQUARIE FIELDS	BARRY DARNTON	02 618 1909
SUTHERLAND	IAN ANNABEL	02 528 3391
SYDNEY EAST	JACKY COCKINOS	02 344 9111
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	DOUG BARBER	067 72 7647
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BROKEN HILL	TERRY NOONAN	080 88 2382
CAMDEN	KEVIN WINTERS	046.66.8068
COFFS HARBOUR	BOB KENNY	066 51 2205
COOMA	ROSS PLATT	0648 23 065
COORANBONG	GEORGE SAVAGE	049 77 1054
COOTAMUNDRA	CHERYL WILLIS	069 42 2264
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 2095
FORBES	JOHANNA VAGG	068 52 2943
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	PETER LINDSAY	066 42 2503
GUYRA	MICHAEL J. HARTMANN	067 79 7547
JUNEE	PAUL MALONEY	069 24 1860
KEMPSEY	RICK FULLER	065-62-7222
LEETON	BRETT WALLACE	069-53-2081
LISMORE	ROB HILLARD	066 24 3089
LITHGOW	DAVID BERGER	063 52 2282
MAITLAND	BILL SNOW	049 66 2557
MOREE	ALF BATE	067 52 2465
MUDGEE	BRIAN STONE	063-72-1958
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 2095
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
ORANGE	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PORT MACQUARIE	RON LALOR	065 83 8223
SPRINGWOOD	DAVID SEAMONS	047 51 2107
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
URALLA	FRANK MUDFORD	067 78 4391
WAGGA WAGGA	CES JENKINSON	069 25 2263
WYONG	JOHN WALLACE	043 90 0312

NT:

DARWIN	BRENTON PRIOR	089.81.7766

QLD:

BRISBANE:		
BIRKDALE	COLIN NORTH	07 824 2128
BRASSALL	BOB UNSWORTH	07 201 8659
CLAYFIELD	JACK FRICKER	07 262 8869
IPSWICH	MICK MURPHY	07 271 1777
PINE RIVERS	BARRY CLARKE	07 204 2806
SOUTH WEST	BOB DEVRIES	07 375 3161
SANDGATE	MARK MIGHELL	07 269 3846
SCARBOROUGH	PETER MAY	07 203 6723
WOODRIDGE	BOB DEVRIES	07 375 3161
AIRLIE BEACH	GLEN EVANS	079 46 1264
BIGGENDEN	ALAN MENHAM	071 27 1272
BOWEN	TERRY COTTON C/O	077 86 2220
BUNDABERG	RON SIMPKIN	C/O TANDY
CAIRNS	GLEN HODGES	070 54 6583
DALBY	MERRICK TANISKY	074.62.3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 51 0015
GYMPIE	BERT LLOYD	071 821910C
HERVEY BAY	LESLEY HORWOOD	071 22 4989
MACKAY	LEN MALONEY	079511333x782
MARYBOROUGH	JOHN EFFER	071 21 6638
MT ISA	JACK RAE	077 43 3486
MURGON	PETER ANGEL	071 68 1628
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
TARA	STEVEN YOUNGBERRY	
TOOWOOMBA	LEN GERSEKOWSKI	076 35 8264
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
WHITEROCK	GLEN HODGES	070 54 6583

SA:

ADELAIDE	JOHN HAINES	08 278 3560
NORTH	STEVEN EISENBERG	08 250 6214
GREENACRES	BETTY LITTLE	08 261 4083
MORPHETTVALE	KEN RICHARDS	08 384 4503
PORT NOARLUNGA	ROB DALZELL	08 386 1647
SEACOMBE HTS	GLENN DAVIS	08 296 7477
PORT LINCOLN	BILL BOARDMAN	086 82 2385
PORT PIRIE	VIC KNAUERHASE	086 32 1230
WHYALLA	MALCOLM PATRICK	086 45 7637

TAS:

HOBART	BOB DELBOURGO	002 25 3896
KINGSTON	WIM DE PUIT	002 29 4950
LAUNCESTON	BILL BOWER	003 44 1584
WYNYARD	ANDREW WYLLIE	004 35 1839

VIC:

MELBOURNE:		
MELBOURNE CCC	JOY WALLACE	03 277 5182
DANDENONG	DAVID HORROCKS	03 793 5157
DONCASTER	JUSTIN LIPTON	03 857 5149
FRANKSTON	BOB HAYTER	03.783.9748
NARRE WARREN	LEIGH EAMES	03 704 6680
NTH EASTERN	PETER WOOD	03 435 2018
MELTON	MARIO GERADA	03 743 1323
RINGWOOD	IVOR DAVIES	03 758 4496
SUNBURY	JACK SMIT	03.744.1355
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
CHURCHILL	GEOFF SPOWART	051 22 1389
DAYLESFORD	DANNY HEDJI	054 24 8329
GEELONG	DAVID COLLEN	052 43 2128
HASTINGS	MICHEAL MONCK	059 79 2879
MAFFRA	MAX HUCKERBY	051 45 4315
MOE	JIMMY WELSH	051 27 6984
MORWELL	GEORGE FRANCIS	051 34 5175
SALE	BRYAN McHUGH	051 44 4792
SHEPPARTON	ROSS FARRAR	058 25 1007
SMYTHESDALE	TONY PATTERSON	053 42 8815
SWAN HILL	BARRIE GERRAND	050.32.2838
TONGALA	TONY HILLIS	058 59 2251
TRARALGON	MORRIS GRADY	051 66 1331
WONTHAGGI	LOIS O'MEARA	056 72 1593
YARRAWONGA	KEN SPONG	057 44 1488

WA:

PERTH	IAN MACLEOD	09 448 2136
GIRRAWHEEN	HANK WILLEMSEN	09 342 7639
KALGOORLIE	TERRY BURNETT	090.21.5212

CANADA - CoCo:

Ontario	Richard Hobson	416 293 2346

special interest groups

BUSINESS:

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

OS9 GROUPS:

NATIONAL OS9 USERS' GROUP	GRAEME NICHOLS	02 451 2954
NSW		
SYDNEY		
BANKSTOWN	CARL STERN	02 646 3619
CARLINGFORD	ROSKO MCKAY	02 624 3353
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY EAST	JACKY COCKINOS	02.344.9111
COOMA	FRED BISSELING	0648 23263
QLD		
BRISBANE	JACK FRICKER	07 262 8869
VIC		
LATROBE VLY	GEORGE FRANCIS	051 34 5175
WA		
KALGOORLIE	TERRY BURNETT	090.21.5212

MC-10 GROUPS:

LITHGOW	DAVID BERGER	063 52 2282
ORANGE	DAVID KEMP	063 62 2270
PORT LINCOLN	BILL BOARDMAN	086 82 2385
WARRNAMBOOL	GARY FURR	055 62 7440

TANDY 1000 / MS DOS:

QLD:		
BRISBANE		
NORTH	BRIAN DOUGAN	07 30 2072
SOUTH	BARRY CAWLEY	07 390 7946
GOLD COAST	GRAHAM MORPHETT	075 51 0015
VIC:		
MELBOURNE	TONY LLOYD	03 500 0878
NSW:		
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY WEST	ROGER RUTHEN	047.39.3903
WYONG	JOHN WALLACE	043 90 0312

FORTH:

BRISBANE	JOHN POXON	07 208 7820
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
SYDNEY	JOHN REDMOND	02 85 3751

ROBOTICS:

BOWEN	TONY EVANS	077 86 2220
GOLD COAST	GRAHAM MORPHETT	075 51 0015
TAMWORTH	ROBERT WEBB	067 65 7256
WAGGA WAGGA	CES JENKINSON	069 25 2263

CHRISTIAN USERS' GROUP:

COLLIE	RAYMOND L. ISAAC	097 34 1578

300 BAUD BULLETIN BOARDS

SYDNEY:	
INFOCENTRE	02 344 9511
TANDY ACCESS	02 625 8071
THE COCOCONNECTION	02 618 3591
DAIL DUBBO (6pm - 8am)	068 82 5011

1200/75 BAUD TANDY INFORMATION

GOLDLINK	VIATEL *642#
VTX 4000	03 329 2936

OTHER TANDY USERS ON VIATEL

GOLDLINK	VIATEL *642#
BLAXLAND COMPUTER SERVICES	VIATEL *64263#
COMPUTER HUT SOFTWARE	VIATEL *64262#
PARIS RADIO	VIATEL *64268#
POWER CODE	VIATEL *64265#
TANDY	VIATEL *64261#

ALLAN BEALE	726353300
FRED BISSELING	648232630
JACK FRICKER	726288690
JOHN GRIGSBY	945872030
BOB KENNY	665122050
JUDY RUTLEDGE	285350000
ARTHUR SLADE	262289400
ALLAN THOMPSON	838155830
DARCY O'TOOLE	755105770

GOLDLINK

a Goldsoft Service

on

VIATEL®

*642#

AUSTRALIAN RAINBOW MAGAZINE
Registered by Australia Post –
Publication No. QBG 4009
AUSTRALIAN CoCo / softgold
Publication No. QBG 4007
P.O. BOX 1742
SOUTHPORT. QLD. Australia. 4215.

POSTAGE
PAID
AUSTRALIA