Registered by Australian Post — Publication No. QBG 4009

For your TANDY Color Computer

NZ $3.95 PNG K5.95
$4.95

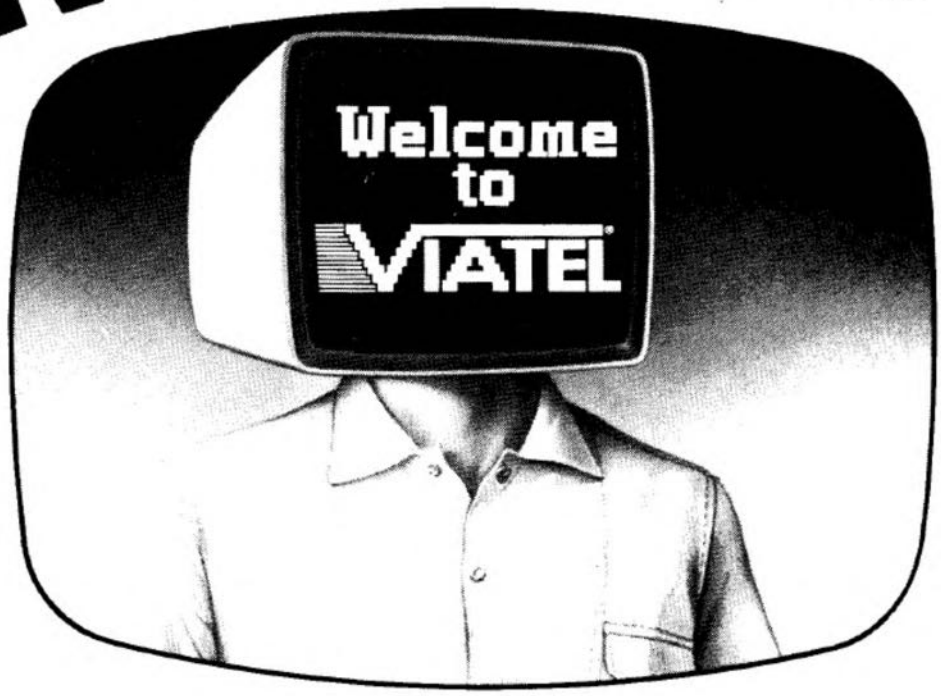
AUSTRALIAN RAINBOW

February, 1987

No.68

The end of the Rainbow

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

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

OTE | 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.

AUSTRALIAN RAINBOW

CONTENTS

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

MK 1 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 & Connectors
- Faster printing for Screen Dumps
- Six Baud rates; 300, 600, 1200 2400, 4800, 9600
- Power Pack required for printers not supplying power at pin 18 on printer connector eg Epson, BMC

ONLY: $97.00

(Includes postage)

add $9.00 for power pack if required

VD1+ Video Driver

- Connect to a Colour or Mono composite Video monitor
- Solderless Installation
- includes Sound Output

$30.00

G & G Fiala
PO Box 46
Thornleigh. NSW 2120
02–84–3172

BLAXLAND COMPUTER SERVICES PTY. LTD.

COCO & 1000 SPECIALISTS

(047) 39-3903

For ALL computer needs, including:

COMPUTER STATIONARY
TANDY COMPUTERS
CoCoTEX 3
AVTEK MODEMS

A complete, professional service awaits you at:

76A MURPHY ST. BLAXLAND 2774

CORRECTION

DISASSEMBLER

"DISASSEM", March Rainbow 1986
There are a few errors in this program.
Line 49 should read:

```
49 IF T>32767 THEN T=-(65536-T)
```

This results in a LBRA disassembly one greater than actual when branching backwards.

Another problem, I found, was in lines 105 and 106. They should be swapped, ie line 105 becomes line 106 and the previous line 106 should be line 105.

by W. Tritscher

Hint . . .

GIME That Lowercase

This one is for the CoCo 3. As you well know, the Color Computer features true lowercase in the 40- and 80-column modes, but not in the 32-column mode. However, the GIME allows emulation of the MC6847T1. To enable the lowercase in the 32-column mode, just enter POKE&H95C9,&H7F:POKE&HFF33,&H10.

Bob Rosen
Howard Beach, NY

WHAT IS THE COST OF EDUCATION ?

ANSWER... FOR AS LITTLE AS 28 CENTS PER MINUTE YOU CAN HAVE ACCESS TO THE WORLD'S LARGEST EDUCATIONAL DATABASE

MACQUARIE LIBRARY
- Dictionary
- Thesaurus
- Encyclopaedic Dictionary
- Book of Events
- Guide to Australian Law
- Dictionary of Trees and Shrubs
- Dictionary of Motoring
- Dictionary of Cookery
- History of Ideas
- Aussie Talk (Dictionary of Colloquialisms)
- Office Manual

Australian Encyclopaedia

Kirk-Othmer Encyclopaedia of Chemical Technology

E.R.I.C. (Educational Resource Investigation Centre)

U.P.I. United Press International News Service

The Think Tank

It doesn't matter what kind of Personal Computer you have, C.T.C. helps you utilize its full potential. C.T.C. improves the quality of education whilst reducing the cost, with a comprehensive range of educational products. All the information from these internationally recognised sources is available to you within seconds, allowing reference to any subject you wish to research. All you need is a personal computer with the appropriate communications software and a modem. C.T.C. can assist you in providing every requirement.

HOW DO I COMMUNICATE WITH C.T.C. ?

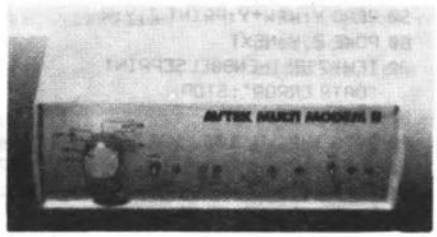
❑ MULTIMODEM II – Australia's No.1 modem

Australia's top selling modem now offers even more. You get:

Reliability. State of the art digital filters for reliable data transfer, even on noisy lines.

The Expansion Bus – an Avtek exclusive. Developments can be plugged straight in.

Total flexibility. Both 300/300 and 1200/75 (Viatel) at the flick of a switch.



Autoanswer as standard. A reliable and instant "ring detect" circuit is completely independent of the strength of the ring.

A 1200 baud full duplex option. At the lowest price in the country.

Multimodem II ——— $349.00

Multimodem II with 1200/1200 option . $585.00

❑ MINIMODEM II – leader in the value stakes!

Superb performance for those on a limited budget. The Minimodem II offers the same digital filtering and error correction as the Multimodem but at a much lower price. Full 300/300 baud **and** 1200/75 baud (Viatel standard) are provided at lower cost than some 300 baud only modems – check for value and you'll find Minimodem wins every time.

Avtek Minimodem ——— $199.00

❑ MEGAMODEM is here – other Smartmodems can't compete

The Avtek Megamodem provides complete compatibility with the industry standard Hayes SM300 and SM2400 modems, while also providing a very easy to use menu driven mode where suitable software is not available.
Full auto recognition of both ingoing and outgoing baud rates makes correct connection very easy. Full autoanswer with accurate ring detect circuitry means reliable remote operation.
For high speed operation, an internally fitted 1200/1200 V22 option is available.

Megamodem 300/300, 1200/75 V21/V23 ——— $499.00

Megamodem V21/V22/V23 (1200/1200 baud option fitted) ——— $699.00

Computer Telecommunications Corporation Limited
11th floor, 189 Kent St, Sydney, NSW 2000, Australia
Telephone: (02) 221 5593. Telex: 21366 CARCOY. Fax: (02) 251 2640
LOCAL CALL FEE ANYWHERE IN AUSTRALIA. (088) 251 308

RAINBOW Info

How To Read Rainbow

Please note that all the BASIC program listings 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 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 press 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 the CoCo and then have the editor/assembler assemble them into specific instructions that are understood by the 6809 chip, which controls your computer.

When using 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
Editor Alex Hartmann
Accounts Annette Morphett
Assistant Editor Julie Vidler
Advertising Graham Morphett
Art Jim Bentick
Sub Editors
Assembly Language: John Poxon
MC-10: Jim Rogers
Softgold: Barry Cawley
Forth: John Redmond
OS-9: Jack Fricker
Special Thanks to
Brian Dougan, Paul Humphreys, Darcy O'Toole, Martha Gritwhistle, Geoff Fiala, John Redmond, Sonya Young Michael Horn, and Mike Turk.

Phone: (075) 51 0577 Voice
Deadlines:
7th of the preceeding month.
Printed by:
Goldsoft
P.O. Box 1742
Southport, Qld. 4215
Registered Publication QBG 4009.

This material is COPYRIGHT. Magazine owners may maintain a copy of each program plus two backups, but may NOT provide others with copies of this magazine in ANY form or media.

PRINT #-2

A MONTH OF changes! I've always wanted something different happen to my routine in life, and this certainly does the job - very well!

The first change is that both Australian Rainbow Magazine and Australian CoCo Magazines are to have their names changed!

As of March, Australian Rainbow Magazine is to be known as Australian CoCo Magazine and the present Australian CoCo/Softgold Magazine is to be known as Softgold Magazine. (See following article from Graham for details.)

"So? How does that affect you?" I hear you ask. From where I stand, it'll mean extra work I'll have to do because of the extra Australian content in this magazine.

The second change comes from an entirely different corner of my work, and that is my work with Viatel. Over the past months, Goldlink's popularity has skyrocketted. More late nights. More change!

The third change comes from Tandy themselves. From January 1st, they are (or were) to be known as "InterTAN Australia Pty Ltd". I can see lots of last minute changes to copy as we swap the word "Tandy" for "InterTAN"!

Hardware News

CoCo 3 GIME chip problem

Some of you will remember your first CoCo 3 and how it had some problems. The reason for that is that the GIME chip was forced out of its sockets by the heat - InterTAN has followed up that problem and has fixed it.

Multipack

It is official that all the multipacks in InterTAN's warehouse have been modified. For those who still wish to have their multipack modified to run on the CoCo 3, the cost is $12.50 which includes everything from parts to labour.

Hardware Mods

And finally, hardware modifications. Last month I talked about putting in some hardware modifications for those hardware hackers. The first mod. to come in is a switch to flick between PMODEs 4 and 3 via a switch. We will publish details next month.

Blaxland Computer Services

Onto a sad note - it is with great sadness that we note Bruce Sullivans' wife death as of the 2nd January. Our condolences to you, Bruce.

Viatel

Goldlink has four new service providers as at end of January. The first of them is ABC Rent-A-Car. Based on the Gold Coast, you can rent a car at a VERY competitive rate. They even rent out Chauffeurred Limousines! They can be found on *6425268#.

The second service provider is East-West Airlines. All the information you'll ever need on flight times, cargo information, etc etc can be found on *64253#.

The third service provider is Trackmedia. They specialize in computer stationary and education software. Their node is *64267#.

And finally, the fourth Service Provider is Gold Key Apartments. They have a large range of holiday apartments you can book into on the Gold Coast or Cairns through Viatel.

Want to book a room? Type in node *6425231# and look around for the room of your choice.

(Credits go to Maurice phillips who done the graphic logos for Gold Key apartments and all of Trackmedia. The rest? Well they're all done by little old me!)

OS-9 Level 2

I remeber saying last month that OS-9 level 2 would be out by the end of this month. Well, it has been postponed - only slightly though. InterTAN now informs us that it'll be out by the end of March. Let's wait and see until then!

Corrections

I have an apology to make! Last month I stated that InterTAN had released the CoCo 3 Level 2.1. Really it was their Disk Extended Basic Level 2.1. Please accept my apology if anyone was misled.

A Change of Name .... An Exciting Future!

As of next month, we're changing the names of both our magazines.

We believe the time has come to break away from the US and to stand on our own feet.

This was the dream of our founder Greg Wilson, and it has always been a stated aim of the magazine since it came to Queensland.

Slowly, but very obviously over the past few years, the quality of articles & programs have built to the point where our contributing authors can consider themselves equal to, or better than, anything the rest of the world can offer.

Dealing with the US has not been easy, and although Lonnie Falk from Falsoft Inc has always been most understanding & helpful, the cost of importing information, added to the additional costs of an unstable currency have made international dealings - from the importer's standpoint - very very difficult.

As a point of interest, when we looked at the prospect of taking over Australian Rainbow magazine, the Aussie Dollar was valued at about 95c US.

During the ensuing years, it dropped to below 60c and currently struggles around the 65c mark.

Even with the additional price of the magazine over that of Greg's day, the price of Australian Rainbow has always been 'too cheap' balanced against the real costs of production. The business of producing further issues of Australian Rainbow is economically unsound without a major price change of around $2.00.

Given that pricing need, and given the quality of the articles & programs which are making it to our door from our own authors, the more intelligent decision seems to be to create an Australian magazine at a slightly reduced price.

Rather than being worried by this change, we're excited & stimulated by it. We're sure it can only mean great things for Tandy Colour Computer Users.

However it is going to be confusing for a month or so.

Because the NEW NAME of Australian Rainbow magazine is (ta da) AUSTRALIAN CoCo.

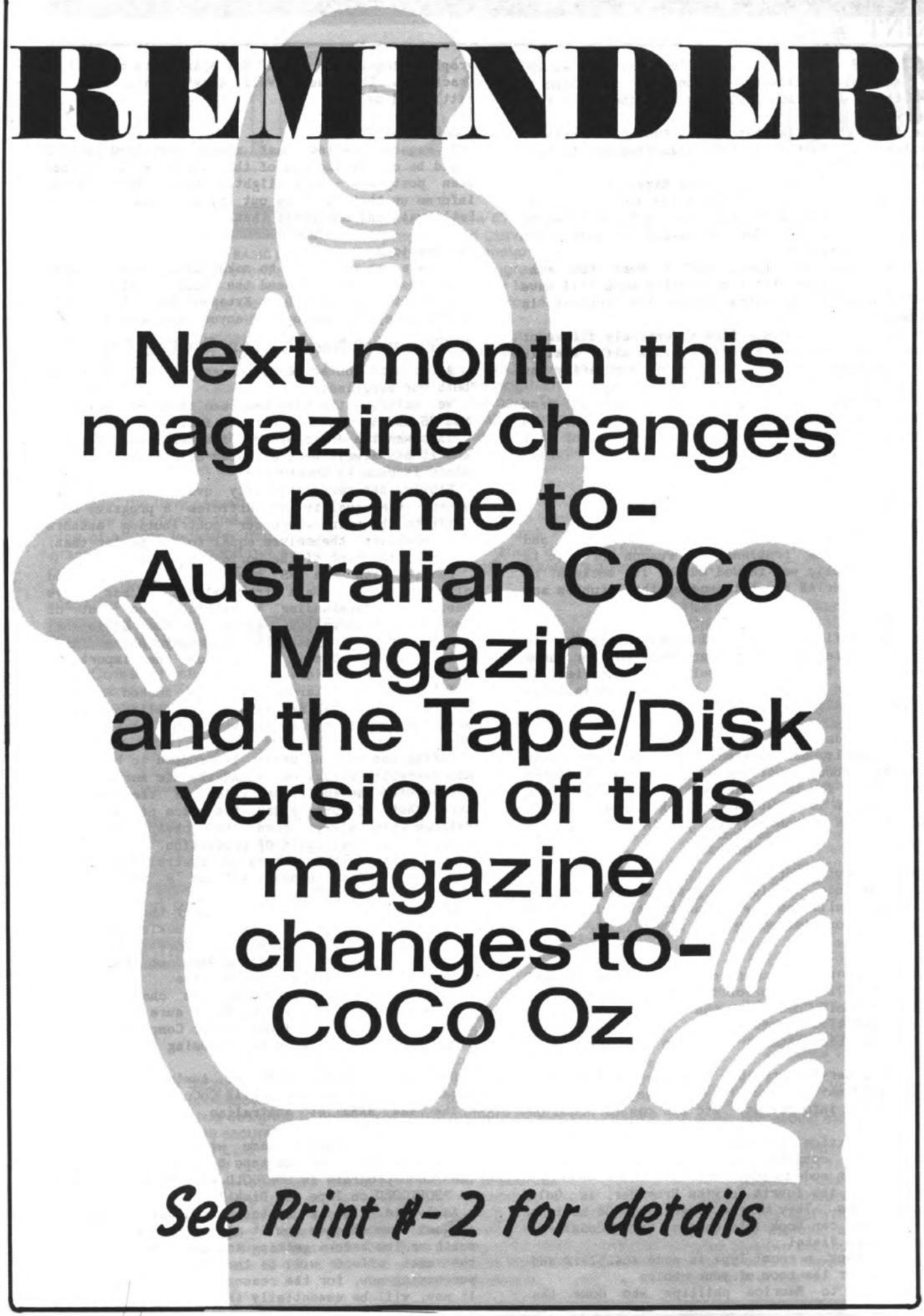
The NEW NAME of Australian CoCo/softgold is SOFTGOLD.

The CoCoOz product name will also transfer to AUSTRALIAN CoCo, and the tape & disk versions of the CoCo programs in SOFTGOLD will be known simply as "SOFTGOLD on Tape (or Disk)".

As I said. It is a touch confusing, but we suggest you observe how it all works over the next month or two before getting too concerned, because the nett effect will be that the product you are purchasing now, for the reasons you are purchasing it now, will be essentially the same.

Alex

REMINDER



Next month this magazine changes name to- Australian CoCo Magazine and the Tape/Disk version of this magazine changes to- CoCo Oz

See Print #-2 for details

USA PRODUCT REVIEWS

Software Review

Disk Programming Package Provides Additional Security

From time to time everyone sits in front of his CoCo and thinks, "I sure wish someone would write a utility to do that. It would be so nice!" At least four of those wishes have now come true thanks to Bob van der Poel Software's *Disk Programming Package*.

The *Disk Programming Package* consists of four utilities that could be of use to all Disk BASIC programmers. The four utilities are ADDML, which appends machine language subroutines to your BASIC program; UNPACKER, which reverses the action of BASIC line packers; MLBASIC, which converts a BASIC program to a machine language file; and JOIN, which links several machine language routines into one file and optionally adds an autoexec feature.

The four utilities come on a single disk, which is not copy protected. The documentation consists of five sheets of typewritten instructions written in a chatty, friendly style that leads you through the operation of each of the utilties. The programs run exactly as the instructions state. The operation of the four utilities is so simple, I was able to run all of them without having to look at the instructions.

The ADDML utility is perhaps the most useful utility of the four. With it, you can take a BASIC program, code the slow parts in machine language, and then use the utility to combine the two into one file. When you run the utility, it asks you for the name of the BASIC program to add the machine language routine to, the name of the machine language file, and the name of the file to write the combined output to. The disk then churns for a while and you have a new BASIC program file, which is the original BASIC program and the machine language routine combined. The new BASIC program can be edited and re-saved many times because the machine language part now "rides" along with it in memory and on disk at the bottom of the new BASIC program. Also created is an extra line in the new BASIC program to calculate the execution offset of the machine language routine.

The UNPACKER utility is designed to reverse the effects of BASIC packer programs that delete spaces and create multi-statement lines when they can. Although a specific packer program is mentioned in the documentation, I found out that UNPACKER will take any BASIC program, packed or not, and create a new program with one statement per line and spaces between all the BASIC keywords.

MLBASIC is a fascinating utility. It converts a BASIC program from BASIC format to a format that can then only be loaded with the LOADM command. In addition, it encrypts the BASIC program and provides for protection against the BREAK key. With this utility, you can easily protect a BASIC program from being listed, edited or changed in any way. The machine language file that MLBASIC creates is fully ready to be placed on a PROM chip. The documentation states that this utility may cause problems on CoCos with Extended BASIC 1.0 because of the PCLEAR bug.

The final utility, JOIN, lets you take several machine language files and merge them all together into one file. You may also add an autoexec option to your new file.

Bob van der Poel has provided all CoCo users with a set of utilities that are both useful and practical. The *Disk Programming Package* is easy to use and well-documented. Dr. Megabyte recommends these utilities to anyone who may have a use for them.

(CMD Micro, 10447-124 Street, Edmonton, Alberta, Canada T5N 1R7; 403-488-7109, $14.95 plus $2 S/H)

— Mark E. Sunderlin

Software Review

A CoCo Nut's Best Friend: Spike, *The Electronic Robotic Dog*

This just has to be the ultimate science fair project! Imagine if you will, a CoCo-controlled robotic dog scampering around in front of your booth nipping at the heels of the judges!

The idea sounded hilarious to me when I first heard of it, but the folks at Electronic Motion Control have developed plans and software to actually accomplish this feat. In spite of the humorous setting I have depicted, the subject should not be taken lightly. The resultant robot built from the available plans is capable of some pretty neat tricks.

The plans consist of detailed drawings, diagrams and circuit board layouts, as well as a parts list of the hardware items needed. Besides a dedicated Color Computer 2 with ECB and 64K RAM, you also need various other circuit boards, motors and gear boxes. A list of where to buy these items is included in the package, many of which can be purchased from EMC.

A program is supplied on tape that allows you to program "Spike" to follow a specific pattern. As Spike runs the pattern he quickly learns his directions and runs it himself without any further training from his master. You program the initial pattern with the use of a joystick plugged into the CoCo. By the way, the computer is removed from its plastic case and installed directly onto Spike's chassis. Your TV or monitor is used during this initial programming process in order to see what you are doing. If you want, you can also hook up a printer and make a listing of the program with your unique pattern and other changes.

The CoCo 2 is modified to the extent of bypassing the AC power supply and using motorcycle batteries and charger for the power supply. Other minor CoCo logic board modifications are also required and detailed.

A clever option utilizing a Polaroid Sonic Board, like that

used in the auto-focus instant camera, allows Spike to "see" obstructions and react accordingly. You can also add an optional speech synthesizer so that Spike can "speak," or maybe just bark.

In looking over the detailed, 41-page instruction booklet, it appeared that everything you need to know to build this robot is there. One should not take this project lightly, however. It requires a lot of sheet metal work and some machining as well as electronic savvy; definitely not a project for the novice. The finished product is well worth the effort and adds still another dimension to your Color Computing interests.

Although this represents a "fun" type project, it will not be a snap to build and train Spike; but unlike the real thing, this dog won't leave a mess on your living room floor!

(Electronic Motion Control Inc., P.O. Box 17271, Air Port Station, Clearwater, FL 33520; 813-536-1694, $69.50 plus $3 S/H)

— Jerry Semones

Software Review

B.E.S.T. Expert System Toolkit

Thinking Software has produced a simple menu-driven approach to generating your own rule-based backward chaining expert system. An expert system asks the user a series of questions to be answered on a scale of zero to 10. The expert then decides on a correct solution. Questionnaires that rate the level of stress in your life or your life expectancy would be good examples of simple expert systems. Large scale systems can be created with hundreds of questions and dozens of possible solutions. An expert system could be built to help you decide what's wrong with your car or if you should take a sick pet to the vet.

Simple menus and clear prompts make *B.E.S.T.* easy to use. You must type in all the questions the expert might need to ask. Then, you type in all the possible solutions and rate each question as it relates to the solution. You can add or modify questions and solutions at any time, so you don't have to complete the system in one sitting.

The expert is smart enough to ask only the questions that it needs to decide on a solution. Often, this will be all the questions in the system. If, however, the expert can tell from your answers that certain solutions have zero possibility of being true, the expert skips any further questions that deal with those solutions.

You do need to keep in mind that the expert is only as good as the questions and solutions that you type into it. The expert is not smart enough to learn from experience.

Though I couldn't test it, the expert toolkit is supposed to work on the new CoCo 3. A talking version is also available for use with the Radio Shack speech cartridge.

Bad points: The expert toolkit would not run properly on a CoCo 1 (D board). Software developers need to remember that not all CoCos are 2s and 3s.

Any expert system that you create must be first booted from the toolkit disk, making it awkward for users who are unfamiliar with computers. Even more unfortunate is that the toolkit disk is copy protected. This one disk that is used to create and start-up any and all of your expert disks will be subject to a lot of wear, yet there is no way to make a backup for safety or to use with each expert system disk. You can, however, order a replacement disk for $5.

The screens that display the title and directions are shown for a set length of time. Average readers will have no problem, but slow readers may not be able to finish reading the directions in time. A prompt to press a key when finished reading would have been better.

The *B.E.S.T.* expert system you create is rather slow as it asks the questions, though the manual does state that the speed-up poke can be used if your computer can handle it.

(Thinking Software, 46-16 65th Place, Woodside, NY 11377; 718-429-4922, B.E.S.T. Expert System with Stock Market and Executive Health Expert Systems ready to consult, $59.95; talking version, $64.95)

— James Ventling

Software Review

Make Your Own *Banner*

Banner is designed to print banners, which it does without fuss or fanfare. The 16-page booklet of instructions is well-written. In clear, simple, easy-to-understand language it tells how to load the program and operate it.

Need a "HAPPY BIRTHDAY" sign for the kiddie party or a "VOTE FOR JOHN HATHAWAY" sign for the political rally? *Banner* produces what you want with neatness and dispatch. Well, the dispatch depends on the printer. I used 600 baud and the speed was satisfactory, but not blinding. If your equipment permits, you can have only one cup of coffee while printing a sign of several words, but don't expect the printer to throw it out faster than you can fold the output.

The menus are clear, complete and easy to use. When the program is loaded with LOADM"BANNER, the first menu comes up on the monitor and asks what baud rate to use. After selecting a baud rate, you are asked if you want a line feed after each carriage return. After you answer this question the main menu comes up, providing eight possible selections. The first selection invites you to enter the message you want printed on the banner. Just type out the message as if you were using a typewriter and press ENTER. Next, select Display Message and check your entry for correctness. If all is well, select Print and watch the banner being prepared before your very eyes!

Banner uses Xs to fill out the characters on the sign or banner unless you select a different character at the main menu, such as graphics blocks for solid figures. The letters are Roman, well-proportioned and neatly formed. The result is easy to read and makes an attractive display.

Other selections from the main menu permit entries of various printer codes to change the height of the characters, etc.

Banner is a fun program and I recommend it even if you don't need banners!

(B. Erickson Software, P. O. Box 11099, Chicago, IL 60611; 312-276-9712, $25)

— Charles L. Redman, Jr.

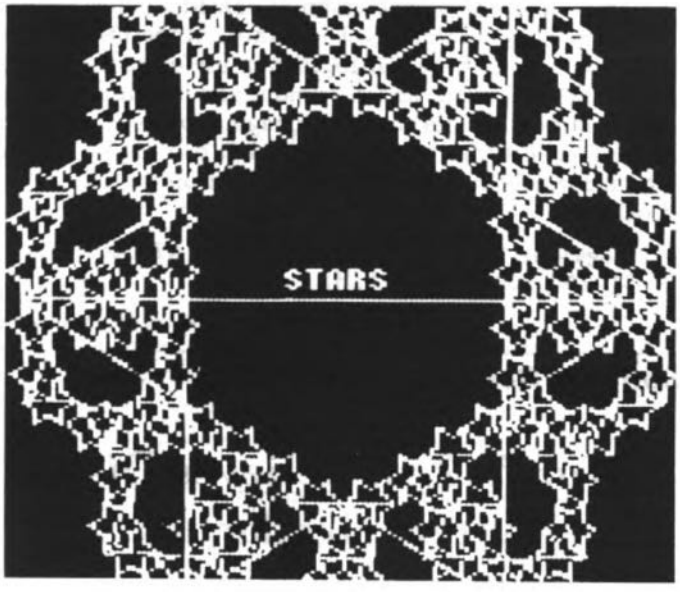
Easy-To-Use *D.L. LOGO* Is as Simple as ABC

What is *D.L LOGO*? In a nutshell, *D.L. LOGO* is a programming language that uses English instead of symbolic commands. No line numbers are used, as it is also a structured language. It has gained popularity in educational circles, and seems to be a good introductory language. Many school systems are standardizing on *D.L. LOGO* for young students.

To run *D.L. LOGO* on your Color Computer you must have a 64K Color Computer and one disk drive. The programming language uses the OS-9 operating system and is a good example of quality OS-9 programming. But you don't need OS-9 to use LOGO. A boot program is included in the back of the manual for users with Disk BASIC 1.0.

To use all of the functions you also need the following optional equipment: Multi-Pak, Speech/Sound Cartridge, X-Pad Graphics Tablet, two joysticks and a dot matrix printer. For speech capabilities you need both the Multi-Pak and Speech/Sound Cartridge. The Multi-Pak is also recommended for the X-Pad.

The documentation is excellent. The manual is slightly over 400 pages long and starts with the very basics and continues with a logical progression of all of the commands and functions. Many programming examples are given and a summary is included at the end of each chapter. I would classify the manual as hands-on and informative, but not technical.



D.L. LOGO uses a method of drawing called "turtle graphics." Two types of screens are used. One is a text screen for program development, and the other is a graphics screen for programs. When using the graphics screen, 16 background and foreground colors are possible. English commands are used to move the turtle around the graphics screen. The turtle leaves a tail. The graphics screen has 256 horizontal and 192 vertical steps.

You can do more than draw with *D.L. LOGO*. As in any other programming language, both string and numerical variables are supported. A full set of numerical and logical functions is included with variable precision from zero (an integer) to 100 places. When first loaded the precision is set to two places.

Sound/Speech is supported by SAY and SOUND primitives. (*D.L. LOGO* calls commands primitives.) Using these audio commands, an example is given for how to use your CoCo as a talking alarm clock. How about that?

A full set of file-handling commands is also included. An editor is included for memory-resident files. As in any other language, you can store and retrieve files using your disks. One example in the manual is a disk catalog program.

Dale Lear, the author of *D.L. LOGO*, has really done justice to LOGO for the Color Computer. I am pleased to see Radio Shack make the commitment to education by the introduction of this type of program.

There was one problem. I never could get the demo program to fully execute. I kept getting an OM Error. Several other programming examples on the production disk ran without a flaw.

I recommend *D.L. LOGO* as a full implementation of LOGO for the Color Computer. If you are having trouble understanding BASIC, or are already familiar with LOGO, try it; it's like learning your ABCs.

(Available in Radio Shack stores nationwide, Catalog No. 26-3033, $99.95)

— Dan Downard

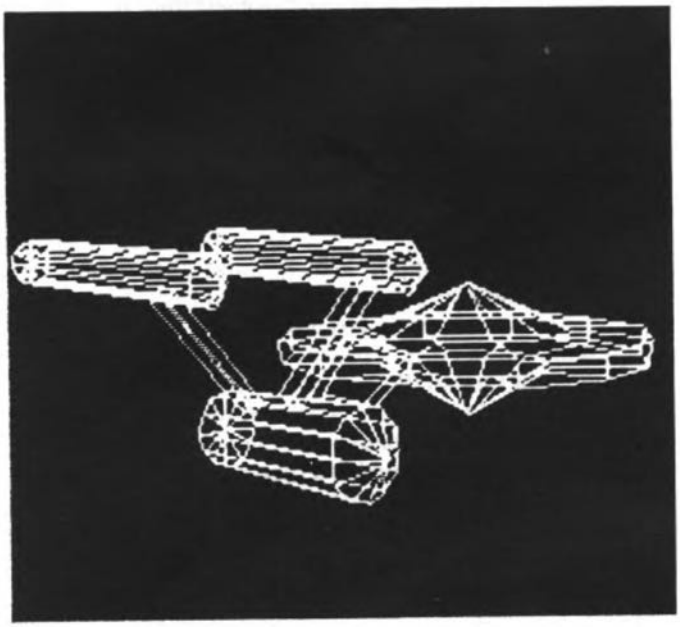
Software Review

You Can Have *3-D Graphics* Without the Glasses

As you may have noticed, one of the things the CoCo does best is graphics. Logicware has developed a program that makes CoCo shine in that regard, and with a flair.

This machine language program is available on tape or disk and needs 64K of RAM and Extended Color BASIC. It is not copy-protected so a backup copy is possible. A 29-page detailed set of instructions is well-done and full of examples on how to put the program through its paces.

3-D Graphics provides simultaneous rotation, movement, zoom and animation of 3-D graphics images. The images can be printed on most dot matrix printers; the program is specifically designed to work with Radio Shack printers.



Samples of what can be done with this software are provided. You can see a spaceship (*The Enterprise*), a cube, a sphere and a pyramid, all of which rotate and move about the screen to show every perspective.

Running the program is as simple as loading and executing THREED. You are prompted to enter a command. Pressing H provides a menu of options. A demonstration of 3-D with animation of the spaceship can be viewed.

A zero is used for the shortest possible delay time between successive graphics images. It provides very smooth rotation and alternate views of the spaceship. Increasing the value results in stop-action-type still shots. Exit the viewing mode by pressing the ENTER key.

A demonstration is also provided using a sphere, a cube and a pyramid. The unique thing about this demo is that not only are the objects rotating and moving on the screen, but the pyramid actually zooms from a small to a large size giving it the appearance of real 3-D as it moves toward the viewer. A pretty neat trick — and no 3-D glasses!

Some other unusual techniques are possible with this program. You can select a single view of any of the objects and make it as small or as large as you like.

It takes about 2½ minutes for the view to be completed, and it remains on the screen. While the picture is drawn on a black background, it can be inverted by pressing I and ENTER. At this point you can send the picture to the printer.

As you can see, the commands are very powerful and allow the programmer almost unlimited freedom in ways to view an object. Since the size of the object on the screen is proportional to the Scale Value, the objects size can be changed easily. The Standing Point of Observation Value (SPO) can also be changed for close-up or far away views. The X, Y and Z values of rotation and translation are extremely useful. The center of the screen represents a value of zero while positive values of X go to the right and negative values to the left. The Y values are positive going up and negative going down from the center. The Z values are positive coming toward the user (out of the screen) and negative going away from the user (into the screen). In a similar fashion, the X, Y and Z translations move the object off of center.

Also included is a program called Edit. This program is used to create your own 3-D graphics images. The user enters appropriate coordinates to create any image the imagination can dream up, although I found this part of the program to be a little user hostile. You have to resort to pencil and paper to create these images, since you are dealing with tri-axis information. While this is complex, it is not impossible and I suspect that many CoCo users will catch on quickly. I also feel that the Edit program is lacking in one important area. It needs a way to "see" what's going on. As it stands, you can't see the fruits of your labor without first saving the file and then looking at it with the THREED program. Another useful feature would be a disk command to allow the user to see the directory. A Dir option would be a real help since you wouldn't have to break out of the program to see the filenames you couldn't remember or forgot to write down.

In spite of these shortcomings, *3-D Graphics* is a powerful program that allows talented programmers to create some very interesting 3-D images. The authors, Robert Steidl and Johnathan Lein, have put together a useful package that challenges graphics lovers in a unique and imaginative way. You will soon find out that your imagination does not have to be limited to flat screen images, but can display depth and movement in a way not often seen on the likes of a CoCo.

(Logicware, 730 W. McDowell Rd., Phoenix AZ 85007; 602-821-2465, $32.95 plus $3 S/H)

— Jerry Semones

Software Review

Map 'n Zap Relieves Disk System Headaches

Do you want to learn about how and where programs are actually stored on your disks? Have you ever had the frustrating experience of I/O Errors with your disks? Have you ever killed a file only to immediately (yet, not quickly enough) realize it was your favorite game from THE RAINBOW? Then *Map 'n Zap* is for you!

Map 'n Zap is a disk editor and repair system intended for anyone who has discovered the headaches that go with owning a disk system. This person might be a layman or a big-time "hacker." Disk drives really are nice. They provide a great amount of fast and usually reliable data storage. Occasionally, though, things go awry and you are stuck with a disk you would like to use for a coffee cup coaster. The *Map 'n Zap* manual, along with the associated programs, will take you away from your misery and put you back in business.

With *Map 'n Zap*, you can alter how data appears on the disk. You can edit the directory and restore those inadvertently killed files. You can even edit your programs directly on the disk. The included manual takes you from step one of disk repair through the entire process. At the same time, it gives you a good deal of knowledge about your disk system and how files are stored. While the manual may sometimes get a little over the head of the average user, one or two rereadings of the confusing section will usually clear things up. It is obvious that much thought was given to providing a package for the novice as well as for the advanced CoCo user.

I am thoroughly impressed with what Duck Productions has done. They have offered an excellent package, which includes a bonus directory program and several disk utilities, at a quite affordable price. Everyone who uses a disk drive with his CoCo should have a disk zap program, and *Map 'n Zap* is definitely one which should not be overlooked.

(Duck Productions, 18 Rowe Court, Brampton, Ontario, Canada L6X 2S2; 416-456-0032, $19.95, $24.95 Cnd.)

— Cray Augsburg

Accessory Review

MouseTop — A Furry Companion for Your CoCo

Remember the ads for *Jaws II* — "Just when you thought it was safe to go back in the water"?

Well, just when you think you've seen everything, it always comes along. And here it is: The MouseTop mouse cover.

You don't need this little cover that fits right over your computer mouse. Oh, certainly, it keeps the grime, dust and grit off the mouse. But I have honestly seen darn few people use keyboard, disk drive and (for that matter) typewriter covers. Too much trouble.

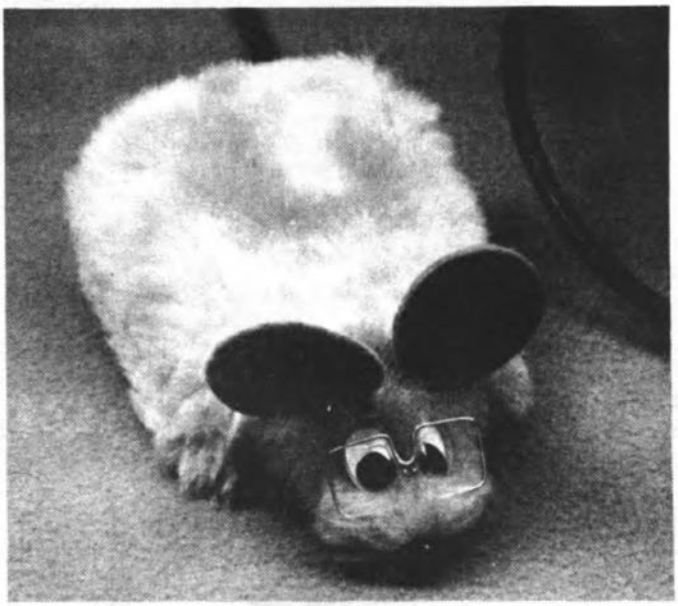
Those things don't have something that the MouseTop does — it's cute.

"What's that?" someone asked a few minutes after it arrived. "It's a mouse cover," I said. "It keeps my mouse clean and warm."

"It's *cute*," was the reply. "Really cute. At last, you have something cute to go with all that equipment."

OK. I've bought laser printers, 70-Meg hard drives, jazzy plotters and every computer Tandy's made in the past five years. Nothing has brought as much comment as this little MouseTop.

It *is* cute. Two ears, glasses and a shiny black nose. It fits



right over my computer's mouse. The "tail" is the mouse cord.

Each MouseTop is hand-made, which makes each unique. And they are machine washable.

I like this little varmint and I think you'll like it, too. It does add a touch of furry personality to your computer.

(H&H Enterprises, P.O. Box 2672, Corona, CA 91718; 714-737-1376, $5.95, 20/20 vision model [without glasses] $5.49)

— Lonnie Falk

Two-Liner

Use the joystick to make your player avoid the potholes. This one is great and is even more difficult when using the high-speed poke.

The listing:

```
1Ø IFN=ØTHENCLS3:PLAY"L25503ABGF
EDCCC":PRINT@Ø,"SCORE:";SC:X=31:
Y=31:K=K+2:FORT=1TOK+1Ø:SET(RND(
63),RND(26)+1,2):NEXTT ELSE C=JO
YSTK(Ø):IFC<1ØTHENX=X-1ELSEIFC>5
3THENX=X+1
2Ø Y=Y-1:IFPOINT(X,Y)=2THENPRINT
"FINAL SCORE:";SC:END ELSE SC=SC
+1:PRINT@Ø,"SCORE:";SC:IF Y<2THE
NN=Ø:GOTO1Ø ELSE N=1:SET(X,Y,5):
PLAY"L25501CC":FORT=1TO75:NEXTT:
SET(X,Y,3):GOTO 1Ø
```

Dean Amo
Wethersfield, CT

GAME

16K ECB

Only your skill and nerve can protect the planet from these elusive enemies.

Let the Laser Battle Begin

By Curt Coty

Def Mov is an action game which challenges you to defend your home planet against the enemy. After loading and running *Def Mov*, a low resolution title screen appears while the computer sets up the graphics screen. When the game board appears, the computer plays a short tune and the ground begins to scroll by as you fly your ship over the blue soil. Press the right joystick button to begin your battle against the enemy.

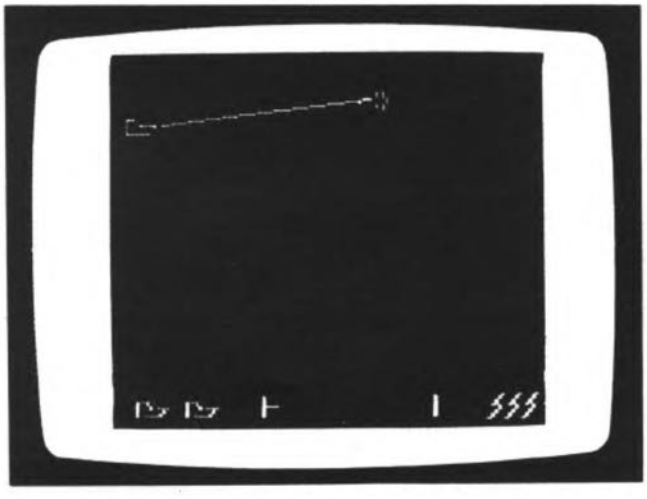
The object of the game is to shoot down the enemy before his ship reaches the left side of the screen. To do this, you use the right joystick to move your ship vertically and to fire the laser. Your spare ships are located on the lower-left part of the screen. You also have three superzappers in your arsenal that you can activate by moving the joystick completely to the left. Be careful not to activate a superzapper by mistake, for they will prove to be very valuable as the game progresses. Your superzappers are the z-shaped lines on the lower-right side of the screen. When activated, superzappers destroy the enemy regardless of where he is.

The enemy attacks in waves of 10, and beginning with the fourth wave they begin to shoot back at you. With each new wave, the enemy becomes faster and harder to hit.

Located in the lower center of the screen is your enemy casualty gauge. When you shoot down about 35 enemies and the gauge reaches the full mark, you have a chance to earn an extra superzapper. You now enter what is called the bonus tunnel. To earn an extra superzapper you must successfully fly your ship through the jagged tunnel without crashing into the walls. Once the ship has started moving you can't stop it, so be ready.

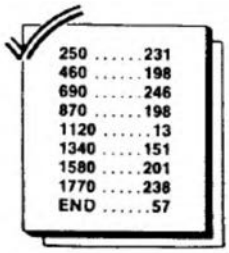
The scrolling of the ground and the movement of the ships are all created using the GET and PUT statements. I used combinations of the PMODE4 and PMODE3 screens to add color to the high resolution PMODE4 screen. Score is kept by summing the horizontal positions of the enemy ships each time you hit one. Therefore, the quicker you shoot the enemy, the greater the X value will be, and in turn, the more points you will earn.

I welcome comments on the quality of my program, or perhaps suggestions on improving the play. I am only familiar with BASIC language, however. Good luck and have fun! □



(Questions or comments about this game may be addressed to Curt at 4072 Eleven Mile Rd., Auburn, MI 48611. Please enclose an SASE when writing.)

Curt Coty attends college in Michigan, and is pursuing a degree in electrical engineering. A self-taught programmer, Curt also enjoys sports.



The listing: DEF MOV

```
1Ø 'DEF MOV BY CURT COTY  1Ø/24/
86
2Ø POKE 65495,Ø
3Ø CLS4
4Ø PRINT@1,STRING$(3Ø,"*");
5Ø PRINT@481,STRING$(3Ø,"*");
6Ø PRINT@237,"def mov";
7Ø PRINT@362,"by curt coty";
8Ø SCREENØ,1
9Ø DIM BG(245,1Ø),SG(1Ø,1Ø)
1ØØ PMODE4,1:PCLSØ
11Ø DRAW"BM9,5Ø;U7;R5;F3;R1Ø;G4;
L13
12Ø ' DRAW GROUND
13Ø PMODE3,1:COLOR2,1:G=165
14Ø FOR T=Ø TO 255
15Ø R=RND(3):G=G+(R-2)
16Ø IF G<16Ø THEN G=16Ø ELSE IF
G>17Ø THEN G=17Ø
17Ø PSET(T,G):NEXT T:PAINT(Ø,19Ø
)
18Ø 'DRAW SUPERZAPPERS
19Ø DRAW"BM23Ø,175;C3;G5;R5;G7
2ØØ DRAW"BM24Ø,175;C3;G5;R5;G7
21Ø DRAW"BM25Ø,175;C3;G5;R5;G7
22Ø 'DRAW RESERVE SHIPS
23Ø DRAW"BM15,185;C3;U7;R5;F3;R1
Ø;G4;L13
24Ø DRAW"BM45,185;C3;U7;R5;F3;R1
Ø;G4;L13
25Ø 'DRAW ENEMY GUAGE
26Ø DRAW"BM19Ø,175;CØ;D1Ø
27Ø DRAW"BM9Ø,175;CØ;D1Ø
28Ø GOSUB 36Ø
29Ø SOUND1,5
3ØØ RUN 31Ø
31Ø DIM SH(25,25):GET(5,35)-(3Ø,
6Ø),SH
32Ø DIM MT(35,2Ø):GET(1ØØ,1ØØ)-(
135,12Ø),MT
33Ø DIM EN(3Ø,15)
34Ø GOSUB 58Ø
35Ø GOTO 45Ø
36Ø ' GROUND SCROLLING ROUTINE
37Ø SCREEN1:POKE 65314,248
38Ø PLAY"O3L16AO4CO3DO4CO3AO4CO3
AO4CO3DO4CO3AO4CO3L16BO4CDCO3BAG
ABAGF#EF#GF#EDC#DEDC#O2BABAGF#GA
BO3C#DO2BAGF#EL8D"
39Ø GET(Ø,16Ø)-(1Ø,17Ø),SG
4ØØ GET(1Ø,16Ø)-(255,17Ø),BG
41Ø PUT(Ø,16Ø)-(245,17Ø),BG
42Ø PUT(245,16Ø)-(255,17Ø),SG
43Ø IF PEEK(6528Ø)=126 OR PEEK(6
528Ø)=254 THEN RETURN
44Ø GOTO 39Ø
45Ø 'MAIN LOOP
46Ø X=22Ø:LL=9Ø:ES=2:Y=35:SZ=23Ø
:SL=45:NO=Ø:SC=Ø:W=1:SN=-2Ø
47Ø A=JOYSTK(Ø):B=JOYSTK(1)
48Ø IF PEEK(6528Ø)=126 OR PEEK(6
528Ø)=254 THEN GOTO 83Ø
49Ø IF B>53 THEN Y=Y+3 ELSE IF B
<1Ø THEN Y=Y-3
5ØØ IF Y>135 THEN Y=135 ELSE IF
Y<Ø THEN Y=Ø
51Ø PUT(5,Y)-(3Ø,Y+25),SH
52Ø PUT(X,R)-(X+3Ø,R+15),EN,PSET
53Ø X=X-ES
54Ø IF X<1Ø THEN GOSUB 1Ø8Ø
55Ø IF A=Ø THEN GOSUB 92Ø
56Ø IF ABS(SN-X)<15 THEN GOSUB 1
83Ø
57Ø GOTO 47Ø
58Ø ' FIRST ENEMY
59Ø NE=Ø
6ØØ COLOR 3,1
61Ø DRAW"BM228,1ØØ;D2;R1Ø;D3;L5;
R1Ø;U3;R2;U3;L2;U3;L1Ø;R5;D3;L1Ø
;D1
62Ø GET(223,93)-(253,1Ø8),EN,G
63Ø R=93:W=W+1:ES=ES+1:IF ES>8 T
HEN ES=8
64Ø CLS4:PRINT@298,"score";SC;:P
RINT@266,"wave no.";W;
65Ø FOR T=1 TO 2ØØØ:NEXT T:PMODE
4,1:SCREEN1:POKE 65314,248
66Ø RETURN
67Ø ' SECOND ENEMY
68Ø COLOR 3,1
69Ø DRAW"BM228,1ØØ;E5;R1Ø;D2;L4;
G3;F3;R4;D2;L1Ø;H5
7ØØ GET(225,93)-(255,1Ø8),EN,G
71Ø R=93:ES=ES+1:W=W+1:IF ES>8 T
HEN ES=8
72Ø CLS4:PRINT@298,"score";SC;:P
RINT@266,"wave no.";W;
73Ø FOR T=1 TO 2ØØØ:NEXT T:PMODE
4,1:SCREEN1:POKE 65314,248
74Ø RETURN
75Ø ' THIRD ENEMY
76Ø COLOR 3,1
77Ø DRAW"BM226,1ØØ;D2;R2Ø;U3;L2Ø
;D1;U1;R8;E3;R7;D9;L7;H3
78Ø GET(225,93)-(255,1Ø8),EN,G
79Ø W=W+1:R=93:ES=ES+1:IF ES>8 T
HEN ES=8
8ØØ CLS4:PRINT@298,"score";SC;:P
RINT@266,"wave no.";W;
81Ø FOR T=1 TO 2ØØØ:NEXT T:PMODE
4,1:SCREEN1:POKE 65314,248
82Ø RETURN
83Ø 'SHOOTING ROUTINE
84Ø PMODE3,1:COLOR 2,1
85Ø LINE(3Ø,Y+11)-(255,Y+11),PSE
T
86Ø PLAY"T255;L255;BBBBGGGEEC
87Ø COLOR1,Ø:LINE(3Ø,Y+11)-(255,
Y+11),PSET
88Ø ES=ES+.5:IF ES>8 THEN ES=8
89Ø NO=NO+1:IF NO>12 THEN NO=12
9ØØ IF ABS((Y+3)-R)<6 THEN GOSUB
 12ØØ
91Ø GOTO 51Ø
92Ø 'SUPERZAPPER ROUTINE
93Ø IF SZ=26Ø THEN RETURN
94Ø PMODE4,1:COLOR 1,Ø
95Ø LINE(3Ø,Y+11)-(X+15,R+7),PSE
T
96Ø FOR T=1 TO 5Ø
97Ø PLAY"T255;A;B;C
98Ø SCREEN1:PMODE3,1
99Ø PLAY"T255;D;E;F
1ØØØ SCREEN1:PMODE4,1
1Ø1Ø NEXT T
1Ø2Ø PMODE4,1:SCREEN1:COLOR Ø,1:
POKE 65314,248:LINE(3Ø,Y+11)-(X+
15,R+7),PSET
1Ø3Ø PMODE3,1:DRAW"BM"+STR$(SZ)+
",175;C2;G5;R5;G7
1Ø4Ø SZ=SZ+1Ø
1Ø5Ø PUT(X,R)-(X+3Ø,R+15),MT,PSE
T
1Ø6Ø GOSUB 12ØØ
1Ø7Ø RETURN
1Ø8Ø 'ENEMY REACHES LEFT SIDE
1Ø9Ø PLAY"T5;L2ØØ;O5;BAGFEC;O3;B
AGFEC;O1;BAGFEC;L1;C
11ØØ FOR T=1 TO 4ØØ:NEXT T
111Ø IF SL=-15 THEN 191Ø
112Ø PLAY"T1ØØ;L1ØØ;O2;BBAAGGFFE
ECCEFGAB
113Ø DRAW"BM"+STR$(SL)+",185;C2;
U7;R5;F3;R1Ø;G4;L13
114Ø PUT(5,Y)-(35,Y+2Ø),MT
115Ø PUT(X,R)-(X+35,R+2Ø),MT
116Ø R=RND(14Ø):X=22Ø
117Ø PLAY"O3
118Ø SL=SL-3Ø
119Ø RETURN
12ØØ 'SHIP HITS ENEMY
121Ø PUT(X,R)-(X+35,R+15),MT
122Ø PUT(X,R)-(X+2Ø,R+15),EN
123Ø PLAY"T255;L255;BBCCBBCCBBCC
124Ø PUT(X,R)-(X+35,R+15),MT
125Ø SC=SC+INT(X)
126Ø ES=ES-(NO*.5):NO=Ø
127Ø R=RND(14Ø):X=22Ø
128Ø NE=NE+1:IF NE=1Ø THEN GOSUB
 67Ø ELSE IF NE=2Ø THEN GOSUB 75
Ø ELSE IF NE=3Ø THEN GOSUB 58Ø
129Ø LL=LL+3
13ØØ PMODE3,1:PSET(LL,18Ø,Ø):PSE
T(LL-1,18Ø,Ø):PSET(LL-2,18Ø,Ø)
131Ø IF LL>186 THEN SOUND 1Ø,1Ø:
GOSUB 135Ø
132Ø SC=SC+INT(X)
133Ø IF W>3 THEN SN=RND(21Ø)
134Ø RETURN
135Ø 'BONUS TUNNEL ROUTINE
136Ø PUT(5,Y)-(35,Y+2Ø),MT
137Ø CLS4:PRINT@225,"prepare to
enter bonus tunnel";
138Ø PRINT@298,"score";SC;
139Ø PMODE4,1:COLOR 1,Ø
14ØØ LINE(Ø,7Ø)-(2Ø,9Ø),PSET
141Ø LINE(Ø,128)-(2Ø,1Ø8),PSET
142Ø I=9Ø
143Ø FOR P=2Ø TO 255
144Ø R=RND(3):I=I+(R-2)
145Ø IF I<8Ø THEN I=8Ø ELSE IF I
>1ØØ THEN I=1ØØ
146Ø PSET(P,I,1):PSET(P,I+18,1)
147Ø NEXT P
148Ø SH$="U7;R5;F3;R1Ø;G4;L13
149Ø DRAW"BMØ,1ØØ;"+SH$
15ØØ SCREEN1:POKE 65314,248:I=1Ø
Ø:SOUND1ØØ,1Ø
151Ø FOR X=Ø TO 15ØØ:NEXT X:SOUN
D 2ØØ,1
152Ø DRAW"BMØ,1ØØ;CØ;"+SH$
153Ø FOR P=Ø TO 235 STEP 2
154Ø A=JOYSTK(Ø):B=JOYSTK(1)
155Ø IF B>53 THEN I=I+1 ELSE IF
B<1Ø THEN I=I-1
156Ø DRAW"BM"+STR$(P)+","+STR$(I
)+"C1"+SH$
157Ø IF PPOINT(P+6,I-8)=1 OR PPO
INT(P+15,I)=1 OR PPOINT(P+19,I-4
)=1 THEN 165Ø
158Ø DRAW"BM"+STR$(P)+","+STR$(I
)+"CØ"+SH$
159Ø NEXT P
16ØØ FOR T=1 TO 2Ø:PLAY"T255;L55
;CEFGAB":NEXT T
161Ø CLS4:PRINT@229,"**bonus sup
erzapper**";
162Ø SZ=SZ-1Ø
163Ø PMODE3,1:DRAW"BM"+STR$(SZ)+
",175;C3;G5;R5;G7"
164Ø GOTO 17ØØ
165Ø 'SHIP CRASHED
166Ø DRAW"BM"+STR$(P+9)+","+STR$
(I-3)+";G5;E1Ø;G5;H5;F1Ø;H5;L5;R
1Ø;L5;U5;D1Ø
167Ø SOUND 1,2Ø
168Ø FOR T=1 TO5ØØ:NEXT T
169Ø CLS4:PRINT@268,"no bonus";
17ØØ COLOR Ø,1
171Ø PMODE4,1
172Ø DRAW"BM"+STR$(P+9)+","+STR$
(I-3)+";CØ;G5;E1Ø;G5;H5;F1Ø;H5;L
5;R1Ø;L5;U5;D1Ø
173Ø COLOR 1,Ø
174Ø LINE(Ø,7Ø)-(255,7Ø),PSET
175Ø LINE(Ø,128)-(255,128),PSET
176Ø PAINT(Ø,9Ø),1,1:PAINT(Ø,9Ø)
,Ø,Ø
177Ø PMODE3,1:COLOR 6,1
178Ø LINE(92,18Ø)-(189,18Ø),PSET
```

continued on page 31

Understanding Relationships Between Fractions, Decimals and Whole Numbers

By Fred B. Scerbo
Rainbow Contributing Editor

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.

Another new year is upon us. It is a time for reflection and a time for new resolutions. As I look back on the past year, I feel that a good deal of what I set out to do in the "Wishing Well" successfully came to pass. We have had games, educational programs and a heavy dose of tutorial approaches in the column as a whole. What does the future hold?

Well, with a little luck I hope to be able to introduce some CoCo 3 programs in the "Wishing Well" before the next year is out. However, I will make this one *firm* resolution: I will not forget the overwhelming numbers of you who have a CoCo 1 and 2. Knowing the vast numbers of RAINBOW readers who rely on these pages for software, I feel it will be a long time until the CoCo 3 users are in the majority. Therefore, fear not. I am not about to discard our trusty Color BASIC and Extended BASIC just because something new has come down the line.

Now, let's get down to business for this month.

Mail and Wishes

During the last year, the "Wishing Well" has become much more difficult to prepare. There are several reasons for this. First, many readers have suggested programs that are virtually impossible to create in BASIC. I cannot write programs that jump in and out of OS-9 or alter the configuration of the BREAK key so it re-boots your disk system. These things are just not possible without getting into assembly language. Remember, the "Wishing Well" was designed to take ideas you submit and synthesize them into real programs in BASIC. Sometimes, ideas from four or five different sources will serve as the stimulus for a different program. Remember, creating a brand new program each month does take time and is not the easiest task in the world.

Secondly, some of the most requested ideas readers have submitted can no longer be created in the "Wishing Well." The most requested features are sequels to *Rockfest, Baseball Fever, Football Fever* and most of my other graphic designs.

Unfortunately, our legal department has informed me that we can no longer reproduce logos and symbols that are protected under copyright. We have not had a major problem with this in the past, but the legal department is paid to warn magazines about areas where trouble could develop. Therefore, don't expect *Rockfest III* or *Basketball Fever.* The lawyers say no.

A few readers have written expressing outrage that I did not drop everything to write their program to project pork belly futures or to calculate schedules for school crossing guards. Some ideas are just too limited and I cannot devote a whole column to a program that only two or three RAINBOW readers can use. My goal always has been to create programs that the greatest number can use. That's why I won't abandon CoCo 1 and 2 either.

A lot of recent mail has commended the Life Skills series and programs like *Color Change Quiz II.* Sheila Jackson from Moline, Ill., wrote that her two children use these programs extensively and asked for more of the same. Kenneth Burdon of Plaistow, N.H., echoed the same ideas in expressing thanks for programs like *Math Driller II,* which his grandchild uses.

Elementary educator Jack Lamoureaux also expressed a need for a program like these dealing with the issue of relationships between numbers. All of us have struggled with those less-than and greater-than signs, < and >. Therefore, the next Life Skills program, offered this month, will deal with number relationships and recognition. Mrs. Jackson, Mr. Burdon and Mr. Lamoureaux, this one's for you, as well as all the other "Wishing Well" readers who have a use for an educational program dealing with a valuable math skill.

The Program

Life Skills 5 is designed to fit into a 16K Color BASIC computer or a 20K MC-10 without modifications. Like most other Life Skills programs, this one has a variable skill level, and three different types of material are covered. On running the program, the user will be presented with three choices: fractions, decimals or whole numbers.

The program is written in such a way as to adjust the location of the numbers displayed on the screen, especially in the case of fractions.

If D is selected for decimals, the next line reads:

```
Select the number of
decimal places (2-6).
```

If F for fractions or W for whole numbers is pressed, the choice is:

```
Select the number of
number places (1-5).
```

Pressing the corresponding number gives the choice you desire. The next prompt asks:

```
Do you want to have
hints given? (Y-N)
```

By hints, I mean explaining what the greater-than (>) or less-than (<) signs stand for. If N is chosen, the multiple choice selections A through C will only have the signs shown.

Next, the screen gives us our first problem and says:

```
Look at the two values below.
The first number is . . .
the second number.
```

Below this are printed the two values, whether they are whole numbers, decimals or fractions. Next will be the three choices:

```
A) Greater than . . . >
B) Less than . . . <
C) Equal to . . . =
```

If no hints are given, only a row of dots and the signs will appear next to the letters. The user need only press the letter desired. If the answer is wrong, the screen displays:

```
Sorry, try again !!
```

If the answer is correct, the screen will say:

```
VERY GOOD! THAT IS CORRECT!
PRESS ENTER TO CONTINUE!
```

An arrow will also flash next to the correct response. Pressing ENTER advances to the next problem while pressing @ gives us our score card, which has now become a standard feature of all these educational programs. When the score card is on the screen, pressing Y reruns the program, pressing N ends it, and pressing C continues the program already in progress.

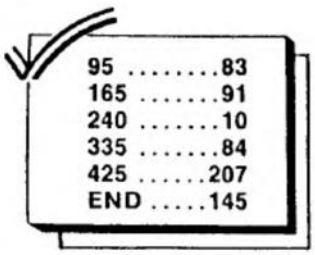
I have not included the function of Computer Paced Learning introduced a few issues ago because it is not appropriate for this material. Still, you may adjust the level of the difficulty at the beginning of each run.

Like all my other educational programs, examining the listing will help you get an idea of how variables can be used to create multi-subject programs. Without the use of variables, it might be necessary to write three different programs to cover fractions, decimals and whole numbers. Study the listing and it might give you some ideas for your own programming skills.

Conclusion

I hope all of you will find this program to be a valuable addition to your educational program library. If my mail has been any indication, many young CoCo users have been cutting their teeth on the software available only in these pages. Therefore, give me a hand in coming months by suggesting some other ideas for material and subjects to be covered in this type of program. Since our options are now so limited, as I mentioned earlier in the article, your input would be helpful.

Also, don't be hesitant to suggest an idea for a game. I have some ideas percolating and someone out there might just provide enough stimulus to get the idea going! See you next month! □



The listing: MATHQUIZ

```
10 REM**************************
15 REM* LIFE SKILLS MATH QUIZ  *
20 REM*  NUMBER RELATIONSHIPS  *
25 REM*   RECOGNITION SKILLS   *
30 REM*     BY FRED B.SCERBO    *
35 REM*      60 HARDING AVE     *
40 REM*  NORTH ADAMS,MA 01247  *
45 REM*   COPYRIGHT (C) 1986   *
50 REM**************************
55 CLS0
60 FORI=1TO32:PRINTCHR$(172);:NE
XT
65 FORI=1TO192:READA:PRINTCHR$(A
+128);:NEXT
70 FORI=1TO32:PRINTCHR$(163);:NE
XT
75 DATA109,104,96,109,104,100,11
0,108,106,109,108,109,,,30,28,26
,29,,,30,20,30,20,30,16,20,30,,2
1,28,29
80 DATA101,,,101,,,106,96,104,10
1,,100,,,26,,24,21,16,22,16,,26,
,26,,16,26,,21,,20
85 DATA101,,,101,,,107,106,96,10
1,99,98,,,27,19,18,21,22,16,,,26
,,26,,16,26,,21,19,19
90 DATA101,,,101,,,106,104,,101,
,,,,,,26,21,20,18,,,26,,26,,16,2
6,,,,21
95 DATA101,,106,101,,,106,,,101,
,97,,16,26,,26,21,,20,18,,26,,26
,21,16,26,21,21,,21
100 DATA103,99,106,103,98,97,107
,,,103,99,103,,,27,19,26,23,18,,
27,17,27,17,27,23,17,27,23,21,19
,23
105 PRINT@293," NUMBER RELATIONS
HIPS ";
110 PRINT@325,"  RECOGNITION SKI
LLS  ";
115 PRINT@357,"   BY FRED B.SCER
BO   ";
120 PRINT@389,"  COPYRIGHT (C) 1
986  ";
125 PRINT@421," (F)RACTION,(D)EC
IMAL ";:PRINT@453,"  OR (W)HOLE
NUMBERS  ";
130 X$=INKEY$:IFX$=""THEN130
135 IFX$="F"THEN FR=1:GOTO190
140 IFX$="D"THEN FR=0:VL=32:A=0:
GOTO155
145 IFX$="W"THEN FR=0:VL=32:GOTO
190
150 GOTO130
155 PRINT@421," SELECT THE NUMBE
R OF ";:PRINT@453," DECIMAL PLAC
ES (2-6)";
160 X$=INKEY$:IFX$=""THEN160
165 X=VAL(X$):IFX<=1THEN160
170 IFX>6THEN160
175 N$=".":FORI=1TOX:N$=N$+"#":N
EXT
180 GOTO230
185 GOTO185
190 PRINT@421," SELECT THE NUMBE
R OF ";:PRINT@453." NUMBER PLACE
S  (1-5)";
195 X$=INKEY$:IFX$=""THEN195
200 X=VAL(X$):IFX<=0THEN195
205 IFX>5THEN195
210 A=10^X:N$="#":FORI=1TOX:N$=N
$+"#":NEXT
215 D$="  ":FORI=1TO6-X:D$=D$+"
":NEXT:FORI=1TOX:D$=D$+"-":NEXT:
GOTO230
220 VL=32
225 N$="###":GOTO230
230 L$=CHR$(91):R$=CHR$(93)
235 AR$="=>"
240 PRINT@421," DO YOU WANT TO H
AVE  ";:PRINT@453," HINTS GIVEN
? (Y-N) ";
245 X$=INKEY$:IFX$=""THEN245
250 IFX$="Y"THEN265
255 IFX$="N"THEN285
260 GOTO245
265 E$(1)="GREATER THAN.."
270 E$(2)="LESS THAN....."
275 E$(3)="EQUAL TO......"
280 GOTO290
285 FORI=1TO3:FORY=1TO14:E$(I)=E
$(I)+".":NEXTY,I
290 IF FR=1 THEN FG=X-5
295 B=RND(A):C=RND(A):K=RND(10):
IFK=10THENC=B
300 IF FR=0THEN315
305 B(2)=B:C(2)=C:B(1)=RND(A/10)
:C(1)=RND(A/10):IFK=10THENC(1)=B
(1)
310 GOTO330
315 B(1)=B:C(1)=C:IF FR<>1THEN33
0
320 K=RND(2):IFK=1THEN330
325 C(1)=C(1)*2:C(2)=C(2)*2
330 CLS:PRINT@34,"LOOK AT THE TW
O VALUES BELOW.   THE FIRST NUMB
ER IS .....   THE SECOND NUM
BER."
335 PRINT@168+VL-FG,"";:PRINTUSI
NGN$;B(1);:PRINT@178+VL-FG,"";:P
RINTUSINGN$;C(1)
340 IF FR<>1THEN355
345 PRINT@198,D$;:PRINT@208,D$;:
PRINT@232-FG,"";:PRINTUSINGN$;B(
2);:PRINT@242-FG,"";:PRINTUSINGN
$;C(2);
350 B=B(1)/B(2):C=C(1)/C(2)
355 PRINT@294,"A) "E$(1)L$">"R$
360 PRINT@326,"B) "E$(2)L$"<"R$
365 PRINT@358,"C) "E$(3)L$"="R$
370 X$=INKEY$:IFX$=""THEN370
375 IFX$="@"THEN465
380 IFX$<"A"THEN370
385 IFX$>"C"THEN370
390 IF B>C AND X$="A"THEN410
395 IF B<C AND X$="B"THEN410
400 IF B=C AND X$="C"THEN410
405 NW=NW+1:PRINT@422,"SORRY, TR
```

continued on page 31

COCO3 128K

Here's a clever program to display all 64 colors available on the CoCo 3 onscreen at the same time

Color Chart for the CoCo 3

By Rick Adams and Dale Lear

One of the first things new Color Computer 3 users want to do is explore the greatly expanded selection of colors available. Frantically, they flip through the pages of the manual looking for a color chart, or a list of the color codes in numerical order with descriptions of each.

Alas, all that can be found is Sample Program 23 (which shows the colors, eight at a time), and a "color chart" on Page 295 that invites you to run the sample program and "fill in the blanks."

Until now, with the eight colors available on the Color Computer 1 and 2, there has been little ambiguity involved in interpreting the colors. There is no way, other than with a severely mis-adjusted color TV set, or a color-blind observer, that the color red would be mistaken for the color blue, for example.

With the 64 colors now available on the Color Computer 3, however, an element of ambiguity has been introduced. With 64 colors to choose from, there is not only red, there is also light red, dark red, orange-red, red-orange, purplish-red, magenta-red, red-magenta, etc. You get the idea. With so many color shades, each perhaps only slightly different from its neighbor, an element of subjectivity creeps in. As we put it: "One person's light magenta is another's pink." Also, the interpretation of the colors by the video hardware in various TVs may differ due to the adjustment of the tint and color controls, the bandwidth of the TV and many other factors.

So, we suspect that Tandy's "do-it-yourself" color chart was produced to avoid forty zillion phone calls from frantic users wanting to know, "Why does the CoCo 3 color chart in the manual say that Color Code 49 is light cyan, whereas on my TV it is pale blue?"

Don't worry, we have a solution for you. Tandy's Sample Program 23 is 44 lines of BASIC code, and shows eight colors at a time. Our version is 26 lines long (if you don't enter the comments), and shows *all 64 colors on the screen at once!*

Now, we can almost see some of you scratching your heads at this point, and thinking, "Now wait a minute. I know the CoCo 3 is a great machine, but I thought that only 16 colors were available at any one time." Yes, that's true. So, how is it that we can go right ahead and break the rules? The answer is simple: We cheat.

Down at the bottom of the BASIC program listing are a number of DATA statements, containing values that are poked into memory to load a short machine language program. This short program, for which we have provided a separate assembler listing, switches the

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.I. LOGO. *Dale, his wife, Laurel, and their six children live in Petaluma, California.*

Hint . . .

Baud Boy

If your printer is capable of accepting data at a higher baud rate than 600, or you just got that new serial-parallel converter and want to try it out, try the following poke. Location 150 in memory holds values which control how fast data is sent to your printer. The power-up value in this location is 87. If you want to change this, just enter

```
POKE150,X
```

where X equals one of the following values: 600 baud, 87; 1200 baud, 41; 2400 baud, 18; 4800 baud, 7; and 9600 baud, 1. If all you get is garbage, try adding or subtracting a value of one from the above values and re-poking them.

palette registers as a screen is drawn.

Sixteen times for each time the screen is drawn, the palette registers are reloaded with a new set of values. In this manner, every possible available color can be shown on the same screen. Another nice thing about this color chart is that it clearly shows the relationship between the "intensity" and "color phase" portions of the color values.

The color codes are six-bit values. This means that color codes 0-63 are available. For the composite color set, which is the color set those using a color TV will see, the leftmost two bits in the value are the color intensity, while the four remaining bits are the color phase.

The first column of the color chart shows color values 0 through 15, all of which correspond to color phase 0 through 15, color intensity 0. These colors are so dark that some of them may be indistiguishable from black on your TV, unless you turn up the color brightness.

The next column, which displays color codes 16 through 31, again shows values corresponding to color phase 0 through 15, except that the intensity value is 1. These colors should all be visible, yet somewhat dark.

The third column, displaying colors 32 through 47, contains colors with intensity 2. These colors appear quite bright and vibrant.

The last column, showing colors 48 through 63 (which are colors with intensity level 3), appears so bright that its colors are pastels.

You may notice some small horizontal dashes at the left edge of your TV screen. These dashes appear because the colors generated by the GIME chip in the Color Computer 3 become "unsettled" briefly whenever the contents of the palette registers are changed.

Since the palette registers are being reloaded 16 times per screen refresh (which is 960 times a second), this disturbs the GIME enough to result in this side effect. This is even more noticeable if the high-speed poke is removed from the program. You may have noticed this effect on some video games on the Atari VCS, which uses this same technique to expand the number of available colors.

Those of you with analog RGB monitors will see a set of colors on your screens that is very different from what one would see on a color TV or a composite color monitor. On a composite monitor, there are 16 distinct colors that are presented at four different luminance levels. For example, the "colors" 8, 16, 24 and 32 are the same color, but 16 is brighter than 8, 24 is brighter than 16 and 32 is brighter than 24. However, the four sets of 16 colors are presented in a very different way on RGB monitors, and the colors in each set no longer have this relationship. Figure 1 shows the colors you see on a color TV or composite monitor (this was photographed from a Panasonic CT-1300D); Figure 2 shows the colors as displayed on Tandy's CM-8 analog RGB monitor.

Why are the colors so different in the two modes? Perhaps the analog RGB output was added after the GIME chip design was already well into the final design stages.

— Ed Ellers

Listing 1: COLOR3

```
10 '============================
20 'COLOR COMPUTER 3 COLOR CHART
30 '       COPYRIGHT 1986
40 ' BY RICK ADAMS AND DALE LEAR
50 '============================
60 '
70 'SET UP PALETTES AND VIDEO
80 '
90 FOR I=0 TO 15:PALETTE I,0:NEX
T I
100 PALETTE 1,63
110 HSCREEN 2
120 '
130 'HIGH-SPEED POKE
140 '
150 POKE &HFFD9,0
160 '
170 'DISPLAY HEADING
180 '
190 HPRINT (4,1),"Color Computer
 3 - Color Chart"
200 '
210 'DRAW BOXES AND NUMBERS
220 '
230 FOR X=0 TO 3 'COLUMNS
240 HLINE (X*80+10,32)-(X*80+40,
160),PSET,B
250 FOR Y=0 TO 15 'ROWS
260 HPRINT (X*10+5,Y+4),X*16+Y
270 HLINE (X*80+10,Y*8+40)-(X*80
+40,Y*8+40),PSET
280 HPAINT (X*80+20,Y*8+36),8+X,
1
290 NEXT Y
300 NEXT X
310 '
320 'POKE IN MACHINE LANGUAGE
330 '
340 FOR I=4096 TO 4160
350 READ A
360 POKE I,A
370 NEXT I
380 '
390 'EXECUTE MACHINE LANGUAGE
400 '
410 EXEC 4096
420 '
430 'DATA FOR MACHINE LANGUAGE
440 'PORTION OF PROGRAM
450 '
460 DATA 26,80,198,46,247,255,3,
134
470 DATA 16,142,0,16,206,32,48,1
98
480 DATA 70,125,255,2,125,255,3,
42
490 DATA 251,125,255,0,125,255,1
,42
500 DATA 251,90,38,245,125,255,0
,125
```

continued on page 26

BARDEN'S BUFFER

More on PSET, PRESET and Graphics Speed

By William Barden, Jr.
Rainbow Contributing Editor

What do weather radar pictures on local TV news programs and the Color Computer have in common? More than you might think. In the last column I was nonplussed because I had coded an assembly language subroutine for setting and resetting points on a 256-by-192 graphics screen, only to find it was slower than the Microsoft BASIC PSET and PRESET! In this column I will be vindicated, to a certain extent, when I show you a Line subroutine. The whole purpose of this exercise, by the way, is to show you what's involved in CoCo graphics. The assembly language for graphics is not that easy, but once you have a few base subroutines, such as a Set Point and Draw Line, you can build upward. And we'll answer that question about TV weather radar.

PSET Revisited

In case you missed last month's column, the assembly language PSET code is shown in Listing 1. Input to PSET is an X,Y point for a high resolution screen, stored in YX in reverse order. As there are 32 bytes per row in this resolution, each byte specifying eight pixels, dividing YX by 8 gives the displacement to the row. Adding $E00 (for a disk system) points to the actual byte for the point. A point at X,Y = 100,50, for example, has a Y,X value of 50 * 256 + 100 or 12,900. Dividing 12,900 by 8 gives 1,612 for the displacement from the start of the graphics page (any remainder from the division is thrown away). A decimal value of 1612 is Hex $64C. Adding $E00 to $64C gives $144C, the actual byte location containing the bit for the pixel 100,50.

The byte location actually contains on/off status for eight pixels. To locate the proper bit, the least significant three bits of X are examined and used to index into a mask table of eight values. These three bits represent values of 0 through 7; the mask table entries of 0 through 7 contain the proper bit setting, such that the mask table value can be ORed with the pixel byte to set the proper pixel. Using the point at X,Y = 100,50 as an example, the X value in binary is 00110010. ANDing this X value with 00000111 (7) results in:

```
        00110010    X = 50
AND   . 00000111    To mask out three lower bits
        --------
        00000010    Result = 0000010 = 2
```

When this index value is added to the address of MSKTAB, the mask value at MSKTAB + 2 is read. This value, $20, is used to set Bit 5 of the byte location, regardless of the state of Bit 5 previously (set or reset).

```
        XXXXXXXX    Contents of byte location
OR      00100000    MSKTAB value, to set bit
        --------
        XX1XXXXX    Bit 5 set; others unchanged
```

The same process can be done for PRESET, except that an AND of an inverted mask retains all bits except for the one to be reset.

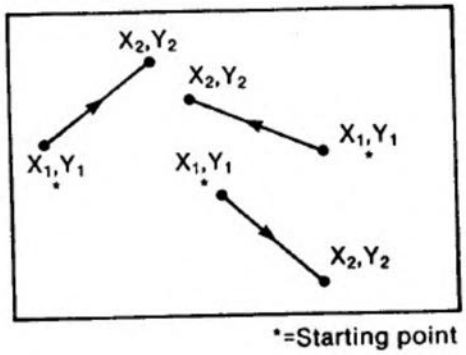
```
        XXXXXXXX    Contents of byte location
AND     11011111    PRESET MSKTAB value, to set bit
        --------
        XX0XXXXX    Bit 5 reset; others unchanged
```

Although this code is efficient and fast, I found to my chagrin that it is not as fast as the Microsoft BASIC PSET. Although my PSET code *is* more efficient, the calling sequence to pass the X,Y parameters to the assembly language subroutine from BASIC is time-consuming.

A Line Subroutine

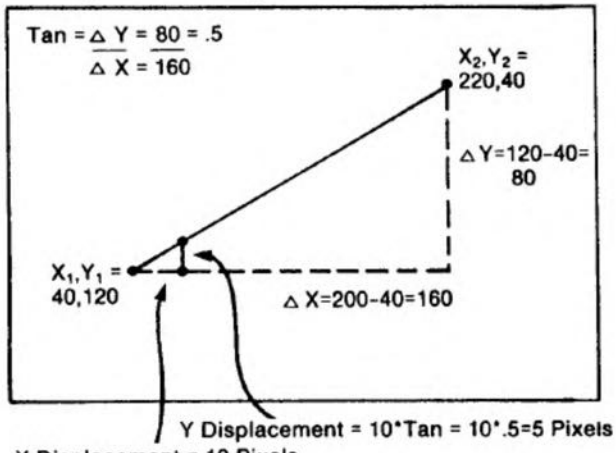
An assembly language subroutine to draw a line on the screen can utilize the PSET and PRESET subroutines. But how do we go about drawing a line on the screen? Let's consider some methods that could be used.

Figure 1: Line Start and End Points



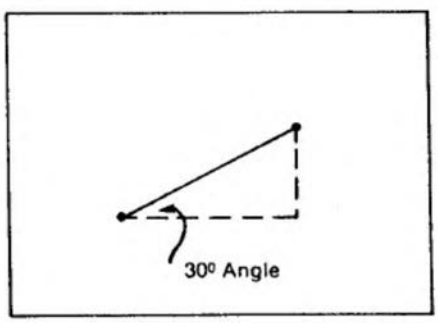
If we keep the same way of specifying lines as BASIC, we'll have a starting point called X1,Y1 and an ending point called X2,Y2, as shown in Figure 1. (We don't have to use this convention. We could specify a starting point, an angle and a line length, for example. We'll stick to the start and end point standard here, however.)

Figure 2: Plotting Points by Trigonometry



We could use trigonometry in plotting the points. The tangent of an angle is the ratio of the opposite side's length over the length of the adjacent side. Advancing one X pixel at a time, we could multiply the tan value by the new X displacement to find the corresponding Y displacement, as shown in Figure 2. But that's awfully messy, for a very good reason. It uses division to get the tan value in the first place and multiplication to get every consecutive point.

Figure 3: Increment Values Stored In a Table



Degrees	Value
1	0.017
2	0.035
3	0.052
4	0.069
5	
29	
30	0.577
31	0.601
44	
45	1.000

Tangent Values 1-45°

Another approach that could be used is to compute the tan value and use it as an index into a table of increment values, as shown in Figure 3. The table would give the amount that the Y value would be incremented for every step in the X direction.

Take an angle of 30 degrees, for example. Every step of one pixel in X causes a step of .577 in the Y direction. Keeping a running total of the accumulated Y value would define the Y pixel to be set. Again, though, this calls for fractional arithmetic which is achieved by floating-point operations, or, at the very least, "scaling."

We're on the right track, however. The ratio of Y to X can be used not to access a table of values, but directly. This is the scheme that Microsoft uses in its implementation of LINE. It's shown in the BASIC program of Listing 2. The algorithm is a variation of one called Bresenham's Line Algorithm, which you can find in books on computer graphics, such as *Computer Graphics* by Schaum's Outline Series. (One that I can heartily recommend — it's probably clearer than most.)

The BASIC code is an emulation of the assembly language code for LINE in Extended Color BASIC at $9401 through $9502.

Microsoft's LINE Algorithm

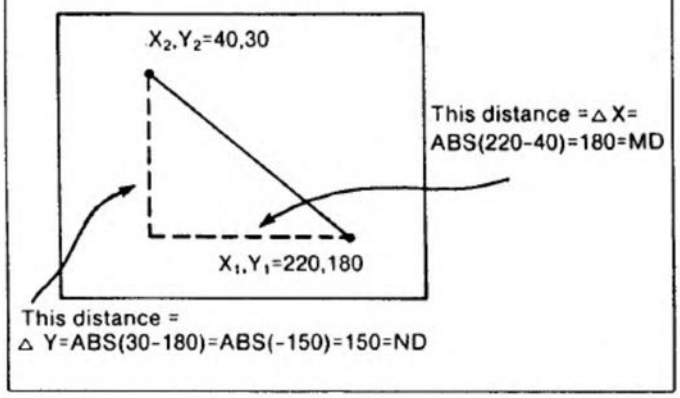
The BASIC code is entered with X1, Y1 defining the starting point, and X2, Y2 defining the ending point. Variables X and Y are set equal to X1 and Y1, the starting point. X and Y hold the coordinate of the current screen point.

First, a check is made of X2 and X1. If X2 is greater than or equal to X1, then the increment from X1 to X2 will be positive; otherwise, it is negative. Variable MA (MAJORACT) is set to 1 or 3, respectively, to record this relationship.

Next, the same check is made of Y2 and Y1, with variable NA (MINORACT) being set to 2 if Y2 is greater than or equal to Y1 or to 4 otherwise.

Next, variables MD (MAJORDELT) and ND (MINORDELT) are set to the absolute value of the actual distance in pixels from start to end, as shown in Figure 4.

Figure 4: Computing MD and ND



If MD (MAJORDELT) is greater than or equal to ND (MINORDELT), one point at a time along the X axis will be incremented. If ND (MINORDELT) is greater, one point at a time along the Y axis will be incremented. In the latter case, NA (MINORACT) is swapped with MA (MAJORACT) and MD (MAJORDELT) is swapped with ND (MINORDELT) so that the "major" action and deltas represent the governing increment, up/down or left/right. The "minor" action involves a fraction of a pixel increment.

Next MC (MAJORCNT) is set to MD (MAJORDELT). MC (MAJORCNT) holds the longest increment/decrement path.

One PSET will occur for every pixel in this path.

NC (MINORCNT) is next set to half of MD (MAJORDELT). NC (MINORCNT) is incremented by ND (MINORDELT) for each PSET. This is a form of scaling where the minor increment represents a fractional increment. When the increment is greater than MD (MAJORDELT), then the current minor coordinate will be incremented or decremented by one. For a nearly flat horizontal line, many increments will be made of NC (MINORCNT) before Y changes. For lines closer to 45 degrees, Y will be changed almost as often as X. For vertical lines, the same action is taken, but on X instead of Y. The minor increment action is really a way to step along the shorter side.

The main loop starts at Line 1000. If the MC (MAJORCNT) represents X, one increment or decrement in the X direction is done for each count in MC (MAJORCNT). For each step, PSET writes the current point. The Y NC (MINORCNT) is then adjusted by adding ND (MINORDELT). If ND (MINORDELT) is greater than MD (MAJORDELT), Y is incremented or decremented in preparation for the next PSET. If the MC (MAJORCNT) represents Y, the same actions are taken but for Y and X. The four short subroutines at the end of the code represent the increment/decrement actions to be taken for incrementing or decrementing X and Y.

Confused? Admittedly, the algorithm is a little confusing. However, if you step through the BASIC code, it's fairly easy to see what is happening. The algorithm is efficient because each point along the line is written only once. In addition, there is no complicated math — just simple additions, comparisons and subtractions.

A typical call to the subroutine is shown in Listing 2 before the code for LINE.

The Weather Radar Puzzle Explained

The idea of writing each point only once is significant. Ten years ago, I worked at a company that developed color digitizers for weather radars. The idea was to take a *News At 5* black-and-white weather radar display and convert it to color. The engineer in charge had spent a great deal of time developing a hardware algorithm to simulate a radar sweep line. (To show how design efforts can be thwarted: The sweep line was "designed out" with a great deal of effort and expense in the converter electronics. However, television stations reported that viewers didn't believe the sweepless picture was really a legitimate radar scan, without the sweep line! The decision was made to put the sweep line back in.)

The engineer's algorithm was based upon incrementing X and Y an amount such that every pixel along the line of the sweep would be filled, but there would be no "gaps." He was also anxious to avoid many overwrites of pixels more than once. Unfortunately, he didn't use the Microsoft algorithm, and the resulting display, although fast enough, was inefficient, with many overwrites. But you know these hardware guys — at nanoseconds instead of milliseconds, they can afford to be sloppy.

In any event, the algorithm *is* one of the best in efficiency. All that's left is to speed it up by our own assembly language code.

Assembly Language Code for LINE

The assembly language code that performs the same algorithm as the BASIC program is shown in Listing 3. It follows the same steps as the BASIC code in about the same order. It is similar, but not identical, to the Microsoft code in ROM that draws lines.

In the main loop of the code, the X and Y registers have been set up to contain the proper subroutine address for incrementing or decrementing X and Y. These subroutines are LIN101, LIN102, LIN103 and LIN104. The D register (A and B combined) contains the minor count, adjusted by adding the minor delta each time through the loop. The major count in MC is decremented at the end of the loop to control the number of PSETs.

An additional function has been added in this code, however. Variable FUNC controls either a PSET or PRESET — a 0 is PRESET while a 1 is a PSET. The code for PSET and PRESET follow the LINE subroutine. It is the same as that presented in the last column, with the exception of saving and restoring the CPU registers with a stack. Separate subroutines for PSET and PRESET speed up the overall execution for setting or resetting points, as no test for the function needs to be made.

Testing the Code

After all this work, it might be nice to have a Line subroutine that actually runs faster than the LINE function in Microsoft BASIC. The code here *is* faster. If used with other graphics assembly language code, it will be several times faster than calling LINE in BASIC. Even when called from BASIC, though, it is about twice as fast as LINE.

Listing 4 shows a BASIC program that illustrates the speed differences between LINE in BASIC and the code. In memory of that hardware engineer, this BASIC code simulates a radar scan in high-resolution graphics. A sweep line scans clockwise around the screen. The BASIC code to do this draws a line from the screen center to the screen edge and then erases it. There are four parts to the code, for each side of the screen.

The point at the screen center is constant, regardless of the direction of the line. We've chosen 128, 96 as the screen center.

The rightmost side is written by drawing a line from the center to a point at X=255 and a varying Y, from 0 (top-right corner) to 191 (bottom-right corner). Immediately after the line is drawn, it is erased by a PRESET. The other three sides use the same scheme — one coordinate for the side is held constant while the second varies. The sweeps of the four sides repeat continuously. One complete sweep around the screen takes about 82 seconds.

Listing 5 shows BASIC code that calls the assembly language LINE code. First, the assembly language object module is loaded from disk after memory from &H3F00 has been protected by a CLEAR. Next, the locations of X1, Y1, X2, Y2 and FUNC in the assembly language LINE subroutine are defined. This just makes it a little less work to pass parameters, using short names instead of hexadecimal addresses.

There are five parameters to be passed: the starting X and Y (X1, Y1), the ending X, Y (X2, Y2) and the functions (0 for reset and 1 for set by FUNC). These parameters are located in the middle of the assembly language Line subroutine — no great disadvantage once X1, Y1, X2, Y2 and FUNC have been defined.

The subroutine is called with the usual calling sequence. A DEF USR0 defines the starting address of &HF000, and a call is made by A=SR0(0) with a dummy argument.

All five parameters need not be defined for each call of the subroutine. The five parameters are not changed by the subroutine so they do not need to be reinitialized if they have not been changed by the BASIC code.

There's a little more work involved here in setting up the parameters, but even with the overhead, a complete sweep takes only 54 seconds.

Using the Assembly Language LINE

You can use PSET, PRESET and LINE to do your own assembly language graphics and get a significant speed increase. You are sacrificing generality, however, as the

assembly language code is designed for only high resolution two-color graphics. It's easy to see how a box function could be added to the code — four separate calls could be made for the four sides of the box. Even a filled-in box is not difficult — just draw a succession of lines from top to bottom.

BASIC graphics functions such as CIRCLE are another problem, but not impossible. A Microsoft circle is done by drawing a 64-sided polygon; again the assembly language LINE code could be used, although Microsoft uses a different approach of a sin/cosine table.

All higher-level graphics functions are built upon these "primitive" graphics operations of PSET, PRESET, LINE and a few others. If these basic functions are efficient, it will help in making higher-level operations fast as well.

CoCo 3 Report

After many phone calls checking on the availability of CoCo 3s, I was getting a little anxious. One evening I walked into my local Radio Shack, however, and glanced toward the CoCo display. This CoCo had more keys! Sure enough, it turned out to be a 3, the only one in the store and that day's new arrival. I immediately bought it. But I was worried. Would the CoCo 3 go? Would CoCo 2 users lust after it, as I had?

I found the answer at the RAINBOWfest in Princeton, N.J., on October 17-19. I was there giving a seminar on computer languages. While there, I was astounded at the number of Color Computer 3s that were being sold — every person walking out the door seemed to have a CoCo 3 box under his arm. Admittedly, the 3s were discounted, but it was apparent to me and everyone else that the CoCo 3 is going to be a hacker's dream and another Color Computer success.

Although I haven't spent a great deal of time on the CoCo 3, I can report that the disk *EDTASM* runs just fine, except for the minor inconvenience of having to restart after coming back to BASIC from the assembler.

I love my 3 and I hope you do, too. There's a lot of material that can be covered in this column about assembly language on the CoCo 3. The old high resolution mode of 256 by 192 in two colors pales by comparison to the 640-by-192 four-color modes. The additional memory of up to 512K leaves plenty of room for high-speed graphics and other applications. And even assembly language can benefit by the higher clock speed.

In future columns we'll be looking at these applications on the CoCo 3 and also looking in detail at assembly language under OS-9. The OS-9 assembler is a different animal from *EDTASM*, but the basic instruction set is the same — it's just the way OS-9 approaches things that makes it slightly more difficult. Please let me know your feelings about what topics you'd like to see covered in the column and the mix of CoCo 3/OS-9 and CoCo 2 topics. Till I hear from you, keep assembling! □

Listing 1:

```
        ORG   $3FØØ
PSET    LDD   YX       get Y to A, X to B
        LSRA           divide by 8 to get row displacement
        RORB
        LSRA
        RORB
        LSRA
        RORB
        ADDD  #$EØØ    point to actual byte
        TFR   D,Y      save in Y
        LDB   YX+1     X value to B
        ANDB  #7       get Ø - 7 value for bit position
        LDX   #MSKTAB  address of mask table
        LDA   ,Y       get byte
        ORA   B,X      set bit
        STA   ,Y       restore byte
        RTS            return
MSKTAB  FCB   $8Ø
        FCB   $4Ø
        FCB   $2Ø
        FCB   $1Ø
        FCB   $Ø8
        FCB   $Ø4
        FCB   $Ø2
        FCB   $Ø1
YX      RMB   2        Y, X in reverse order
        END
```

Listing 2: LINEBAS

```
1ØØ 'BASIC DRAW LINE SUBROUTINE
11Ø SCREEN 1,Ø
12Ø PMODE 4,1
13Ø PCLS
14Ø X1 = Ø: Y1 = 1ØØ: X2 = 255:
Y2 = 95
15Ø GOSUB 17Ø
16Ø GOTO 16Ø
17Ø 'DRAW A LINE SUBROUTINE FROM
 X1,Y1 TO X2,Y2
18Ø X = X1: Y = Y1
19Ø IF X2 - X1 >= Ø THEN MA = 1
ELSE MA = 3
2ØØ IF Y2 - Y1 >= Ø THEN NA = 2
ELSE NA = 4
21Ø MD = ABS( X2 - X1 )
22Ø ND = ABS( Y2 - Y1 )
23Ø IF MD < ND THEN T = MA: MA =
 NA: NA = T: T = MD: MD = ND: ND
 = T
24Ø MC = MD
25Ø NC = MD/2
26Ø IF ( MD AND 1 ) = 1 AND (MA
>= 3 ) THEN NC = NC - 1
27Ø 'DRAW A LINE CORE CODE
28Ø FOR I = MC TO Ø STEP -1
29Ø PSET (X,Y)
3ØØ ON MA GOSUB 36Ø, 39Ø, 42Ø, 4
5Ø
31Ø NC = NC + ND
32Ø IF NC - MD > Ø THEN NC = NC
- MD: BUMP = 1 ELSE BUMP = Ø
33Ø IF BUMP = 1 THEN ON NA GOSUB
 36Ø, 39Ø, 42Ø, 45Ø
34Ø NEXT I
35Ø RETURN
36Ø 'INCREMENT X
37Ø X = X + 1
38Ø RETURN
39Ø 'INCREMENT Y
4ØØ Y = Y + 1
41Ø RETURN
42Ø 'DECREMENT X
43Ø X = X - 1
44Ø RETURN
45Ø 'DECREMENT Y
46Ø Y = Y - 1
47Ø RETURN
```

Listing 3: LINEBIN

```
                        ØØ1ØØ ***********************************************
                        ØØ11Ø * LINE SUBROUTINE. DUPLICATES MS BASIC 'LINE'*
                        ØØ12Ø * INPUT : (X1,Y1)=STARTING POINT IN 256X192  *
                        ØØ13Ø *         (X2,Y2)=ENDING POINT IN 256X192    *
                        ØØ14Ø *         (FUNC)=Ø IF PRESET, 1 IF PSET      *
                        ØØ15Ø * OUTPUT: LINE DRAWN ON SCREEN               *
                        ØØ16Ø ***********************************************
                        ØØ17Ø
3FØØ                    ØØ18Ø          ORG     $3FØØ
3FØØ FC   3F8C          ØØ19Ø LINE     LDD     Y1       INITIALIZE X,Y BEGINNING
3FØ3 FD   3F91          ØØ2ØØ          STD     YY       X1 --> X; Y1 --> Y
3FØ6 8E   3F7C          ØØ21Ø          LDX     #LIN1Ø1  IF DELTA X > DELTA Y
3FØ9 B6   3F8E          ØØ22Ø          LDA     X2       GET END POINT FOR X
3FØC BØ   3F8D          ØØ23Ø          SUBA    X1       X2 - X1 = DELTA X
3FØF 24   Ø4            ØØ24Ø          BHS     LINØ1Ø   GO IF DELTA X POSITIVE
3F11 4Ø                 ØØ25Ø          NEGA             TAKE ABSOLUTE VALUE
3F12 8E   3F84          ØØ26Ø          LDX     #LIN1Ø3  X DECREMENTS DOWN
3F15 B7   3F94          ØØ27Ø LINØ1Ø   STA     MD+1     DELTA X=Ø TO 255
3F18 7F   3F93          ØØ28Ø          CLR     MD       MAKE DOUBLE BYTE
3F1B 1Ø8E 3F8Ø          ØØ29Ø          LDY     #LIN1Ø2  IF DELTA Y > DELTA X
3F1F B6   3F8F          ØØ3ØØ          LDA     Y2       GET END POINT FOR Y
3F22 BØ   3F8C          ØØ31Ø          SUBA    Y1       Y2-Y1 = DELTA Y
3F25 24   Ø5            ØØ32Ø          BHS     LINØ2Ø   GO IF DELTA Y NEGATIVE
3F27 4Ø                 ØØ33Ø          NEGA             TAKE ABSOLUTE VALUE
3F28 1Ø8E 3F88          ØØ34Ø          LDY     #LIN1Ø4  Y DECREMENTS DOWN
3F2C B7   3F96          ØØ35Ø LINØ2Ø   STA     ND+1     DELTA Y=Ø TO 191
3F2F 7F   3F95          ØØ36Ø          CLR     ND       MAKE DOUBLE BYTE
3F32 FC   3F93          ØØ37Ø          LDD     MD       GET DELTA X
3F35 1ØB3 3F95          ØØ38Ø          CMPD    ND       LARGER THAN DELTA Y?
3F39 24   ØB            ØØ39Ø          BHS     LINØ25   GO IF YES
3F3B 1E   12            ØØ4ØØ          EXG     X,Y      SWAP ACTION FUNCTIONS
3F3D FE   3F95          ØØ41Ø          LDU     ND       SWAP DELTAS-MAJOR
3F4Ø FD   3F95          ØØ42Ø          STD     ND       INCREMENT IS LARGEST
3F43 FF   3F93          ØØ43Ø          STU     MD       DELTA
3F46 FC   3F93          ØØ44Ø LINØ25   LDD     MD       GET MAJOR DELTA
3F49 F7   3F97          ØØ45Ø          STB     MC       STORE IN MAJOR COUNT
3F4C 54                 ØØ46Ø          LSRB             MAJOR DELTA/2
3F4D B6   3F94          ØØ47Ø          LDA     MD+1     GET MAJOR DELTA
3F5Ø 84   Ø1            ØØ48Ø          ANDA    #1       TEST FOR ODD# POINTS
3F52 27   Ø6            ØØ49Ø          BEQ     LIN028   GO IF EVEN
3F54 8C   3F84          ØØ5ØØ          CMPX    #LIN1Ø3  ODD DECREMENT=SPECIAL
3F57 25   Ø1            ØØ51Ø          BLO     LIN028   GO IF INCREMENT
3F59 5A                 ØØ52Ø          DECB             ADJUST COUNT BY ONE
3F5A 4F                 ØØ53Ø LIN028   CLRA             D=MINOR COUNT
                        ØØ54Ø * MAIN LOOP HERE - EXECUTED MAJOR DELTA TIMES
3F5B 7D   3F9Ø          ØØ55Ø LIN03Ø   TST     FUNC     TEST PRESET OR PSET
3F5E 26   Ø4            ØØ56Ø          BNE     LINØ35   GO IF PSET
3F6Ø 8D   5F            ØØ57Ø          BSR     PRESET   PRESET CALL
3F62 2Ø   Ø2            ØØ58Ø          BRA     LINØ36   CONTINUE
3F64 8D   32            ØØ59Ø LINØ35   BSR     PSET     PSET CALL
3F66 AD   84            ØØ6ØØ LINØ36   JSR     ,X       BUMP MAJOR POINT
3F68 F3   3F95          ØØ61Ø          ADDD    ND       ADD MINOR DELTA TO COUNT
3F6B 1ØB3 3F93          ØØ62Ø          CMPD    MD       TIME TO BUMP MINOR?
3F6F 25   Ø5            ØØ63Ø          BLO     LINØ5Ø   GO IF NO
3F71 B3   3F93          ØØ64Ø          SUBD    MD       GET NEW COUNT
3F74 AD   A4            ØØ65Ø          JSR     ,Y       BUMP (OR DECR) MINOR
3F76 7A   3F97          ØØ66Ø LINØ5Ø   DEC     MC       DECREMENT MAJOR COUNT
3F79 26   EØ            ØØ67Ø          BNE     LINØ3Ø   REPEAT IF NOT DONE
3F7B 39                 ØØ68Ø          RTS              RETURN FROM LINE SUBR
                        ØØ69Ø
3F7C 7C   3F92          ØØ7ØØ LIN1Ø1   INC     XX       FOR X2 > X1 CASE
3F7F 39                 ØØ71Ø          RTS
                        ØØ72Ø
3F8Ø 7C   3F91          ØØ73Ø LIN1Ø2   INC     YY       FOR Y2 > Y1 CASE
3F83 39                 ØØ74Ø          RTS
                        ØØ75Ø
3F84 7A   3F92          ØØ76Ø LIN1Ø3   DEC     XX       FOR X2 > X1 CASE
3F87 39                 ØØ77Ø          RTS
                        ØØ78Ø
3F88 7A   3F91          ØØ79Ø LIN1Ø4   DEC     YY       FOR Y2 < Y1 CASE
3F8B 39                 ØØ8ØØ          RTS
                        ØØ81Ø
                        ØØ82Ø * WORKING STORAGE AREA
3F8C                    ØØ83Ø Y1       RMB     1        STARTING POINT (INPUT)
3F8D                    ØØ84Ø X1       RMB     1
3F8E                    ØØ85Ø X2       RMB     1        ENDING POINT (INPUT)
```

```
3F8F                    00860 Y2       RMB     1
3F90                    00870 FUNC     RMB     1       0=PRESET, 1=PSET (INPUT)
3F91                    00880 YY       RMB     1       FOR PSET OR PRESET CALL
3F92                    00890 XX       RMB     1
3F93                    00900 MD       RMB     2       MAJOR DELTA, X OR Y
3F95                    00910 ND       RMB     2       MINOR DELTA, X OR Y
3F97                    00920 MC       RMB     1       MAJOR COUNT = # MAJOR PNTS
                        00930
                        00940 * PSET SUBROUTINE
3F98 34   36            00950 PSET     PSHS    D,X,Y   SAVE REGS
3F9A FC   3F91          00960          LDD     YY      Y TO A, X TO B
3F9D 44                 00970          LSRA            /8 TO GET ROW DISP
3F9E 56                 00980          RORB
3F9F 44                 00990          LSRA
3FA0 56                 01000          RORB
3FA1 44                 01010          LSRA
3FA2 56                 01020          RORB
3FA3 C3   0E00          01030          ADDD    #$E00   POINT TO ACTUAL BYTE
3FA6 1F   02            01040          TFR     D,Y     SAVE IN Y
3FA8 F6   3F92          01050          LDB     YY+1    X VALUE TO B
3FAB C4   07            01060          ANDB    #7      GET 0 - 7 VALUE FOR BIT POS
3FAD 8E   3FB9          01070          LDX     #MSKTAB ADDRESS OF MASK TABLE
3FB0 A6   A4            01080          LDA     ,Y      GET BYTE
3FB2 AA   85            01090          ORA     B,X     SET BIT
3FB4 A7   A4            01100          STA     ,Y      RESTORE BYTE
3FB6 35   36            01110          PULS    D,X,Y   RESTORE REGS
3FB8 39                 01120          RTS             RETURN
                        01130 * MASK TABLE FOR PSET
3FB9      80            01140 MSKTAB   FCB     $80
3FBA      40            01150          FCB     $40
3FBB      20            01160          FCB     $20
3FBC      10            01170          FCB     $10
3FBD      08            01180          FCB     $08
3FBE      04            01190          FCB     $04
3FBF      02            01200          FCB     $02
3FC0      01            01210          FCB     $01
                        01220 * PRESET SUBROUTINE
3FC1 34   36            01230 PRESET   PSHS    D,X,Y   SAVE REGS
3FC3 FC   3F91          01240          LDD     YY      Y TO A, X TO B
3FC6 44                 01250          LSRA            /8 TO GET ROW DISP
3FC7 56                 01260          RORB
3FC8 44                 01270          LSRA
3FC9 56                 01280          RORB
3FCA 44                 01290          LSRA
3FCB 56                 01300          RORB
3FCC C3   0E00          01310          ADDD    #$E00   POINT TO ACTUAL BYTE
3FCF 1F   02            01320          TFR     D,Y     SAVE IN Y
3FD1 F6   3F92          01330          LDB     YY+1    X VALUE TO B
3FD4 C4   07            01340          ANDB    #7      GET 0 - 7 VALUE FOR BIT POS
3FD6 8E   3FE2          01350          LDX     #MSKTA1 ADDRESS OF MASK TABLE
3FD9 A6   A4            01360          LDA     ,Y      GET BYTE
3FDB A4   85            01370          ANDA    B,X     SET BIT
3FDD A7   A4            01380          STA     ,Y      RESTORE BYTE
3FDF 35   36            01390          PULS    D,X,Y   RESTORE REGS
3FE1 39                 01400          RTS             RETURN
                        01410 * MASK TABLE FOR PRESET
3FE2      7F            01420 MSKTA1   FCB     $7F
3FE3      BF            01430          FCB     $BF
3FE4      DF            01440          FCB     $DF
3FE5      EF            01450          FCB     $EF
3FE6      F7            01460          FCB     $F7
3FE7      FB            01470          FCB     $FB
3FE8      FD            01480          FCB     $FD
3FE9      FE            01490          FCB     $FE
          0000          01500          END

00000 TOTAL ERRORS
```

One-Liner

Ribbon will generate some interesting designs. Give this one a RUN.

Charles P. Maulick
Staten Island, NY

The listing:

```
0 PCLS:PMODE3,1:SCREEN1,1:PMODE4
:Z=34:Y=0:FORX=0TO1440STEP7.5:D=
X/57.295:S=COS(D)*57.295+Z:LINE(
S,Y+35)-(S,Y),PSET:Y=Y+1:Z=Z+1:N
EXT:GOTO0
```

Listing 4: RADARBAS

```
1ØØ 'RADAR SCAN PROGRAM IN BASIC
11Ø CLEAR 1ØØ,&H3EFF
12Ø LOADM "LINE"
13Ø SCREEN 1,Ø
14Ø PMODE 4
15Ø PCLS
16Ø FOR Y=Ø TO 191: LINE (128,1Ø
Ø)-(255,Y),PSET: LINE (128,1ØØ)-
(255,Y),PRESET: NEXT Y
17Ø FOR X=255 TO Ø STEP -1:LINE
(128,1ØØ)-(X,191),PSET: LINE (12
8,1ØØ)-(X,191),PRESET: NEXT X
18Ø FOR Y=191 TO Ø STEP -1: LINE
 (128,1ØØ)-(Ø,Y),PSET: LINE(128,
1ØØ)-(Ø,Y),PRESET: NEXT Y
19Ø FOR X=Ø TO 255: LINE (128,1Ø
Ø)-(X,Ø),PSET: LINE(128,1ØØ)-(X,
Ø),PRESET: NEXT X
2ØØ GOTO 16Ø
```

Listing 5: RADARBIN

```
1ØØ ' RADAR SCAN PROGRAM USING A
SSEMBLY LANGUAGE PSET
11Ø CLEAR 1ØØ,&H3EFF
12Ø LOADM "LINE"
13Ø SCREEN 1,Ø
14Ø PMODE 4
15Ø PCLS
16Ø X1=&H3F8D: X2=&H3F8E: Y1=&H3
F8C: Y2=&H3F8F: FUNC=&H3F9Ø
17Ø DEFUSR=&H3FØØ
18Ø POKE X1,128: POKE Y1, 1ØØ
19Ø POKE X2,255:FOR Y=Ø TO 191:P
OKE FUNC,1:POKE Y2,Y:A=USRØ(Ø):P
OKE FUNC,Ø:A=USRØ(Ø):NEXT Y
2ØØ POKE Y2,191:FOR X=255 TO Ø S
TEP -1:POKE FUNC,1:POKE X2,X:A=U
SRØ(Ø):POKE FUNC,Ø:A=USRØ(Ø):NEX
T X
21Ø POKE X2,Ø:FOR Y=191 TO Ø STE
P -1:POKE FUNC,1:POKE Y2,Y:A=USR
Ø(Ø):POKE FUNC,Ø:A=USRØ(Ø):NEXT
Y
22Ø POKE Y2,Ø:FOR X=Ø TO 255:POK
E FUNC,1:POKE X2,X:A=USRØ(Ø):POK
E FUNC,Ø:A=USRØ(Ø):NEXT X
23Ø GOTO 19Ø
```

Color Chart for the CoCo 3

continued from page 19

```
51Ø DATA 255,1,42,251,191,255,18
4,255
52Ø DATA 255,186,48,137,1,1,51,2
Ø1
53Ø DATA 1,1,198,7,74,38,218,32,
198
```

Listing 2:

```
                        ØØØ1Ø *=========================
                        ØØØ2Ø * CYCLE VERSION 1.Ø
                        ØØØ3Ø * COPYRIGHT 1986 BY
                        ØØØ4Ø * DALE LEAR AND RICK ADAMS
                        ØØØ5Ø *=========================
                        ØØØ6Ø
                        ØØØ7Ø * CYCLE THROUGH 16
                        ØØØ8Ø * DIFFERENT PALETTE COMBOS
                        ØØØ9Ø * TIMED WITH THE
                        ØØ1ØØ * HORIZONTAL SYNC
                        ØØ11Ø * CLOCK, TO YIELD ALL
                        ØØ12Ø * 64 COLORS ON A SINGLE
                        ØØ13Ø * SCREEN
                        ØØ14Ø *
                        ØØ15Ø
1ØØØ                    ØØ16Ø          ORG     $1ØØØ
                        ØØ17Ø
                        ØØ18Ø * INIT VERT SYNC CLOCK
1ØØØ 1A   5Ø            ØØ19Ø CYCLE    ORCC    #$5Ø
1ØØ2 C6   2E            ØØ2ØØ          LDB     #$2E
1ØØ4 F7   FFØ3          ØØ21Ø          STB     $FFØ3
                        ØØ22Ø
                        ØØ23Ø * SET UP FOR 16 STRIPES
1ØØ7 86   1Ø            ØØ24Ø CYC1     LDA     #16
                        ØØ241
                        ØØ25Ø * INITIAL PALETTE VALUES
1ØØ9 8E   ØØ1Ø          ØØ26Ø          LDX     #$ØØ1Ø
1ØØC CE   2Ø3Ø          ØØ27Ø          LDU     #$2Ø3Ø
                        ØØ271
                        ØØ28Ø * FIRST STRIPE IS 7Ø SCAN
                        ØØ29Ø * LINES DOWN FROM TOP OF
                        ØØ3ØØ * SCREEN
1ØØF C6   46            ØØ31Ø          LDB     #7Ø
                        ØØ311
                        ØØ32Ø * WAIT FOR VERT SYNC
1Ø11 7D   FFØ2          ØØ33Ø          TST     $FFØ2
1Ø14 7D   FFØ3          ØØ34Ø CYC2     TST     $FFØ3
1Ø17 2A   FB            ØØ35Ø          BPL     CYC2
                        ØØ36Ø
                        ØØ37Ø * WAIT FOR HORIZ SYNC
1Ø19 7D   FFØØ          ØØ38Ø CYC3     TST     $FFØØ
1Ø1C 7D   FFØ1          ØØ39Ø CYC4     TST     $FFØ1
1Ø1F 2A   FB            ØØ4ØØ          BPL     CYC4
                        ØØ4Ø1
                        ØØ41Ø * COUNT STRIPE SCAN LINES
1Ø21 5A                 ØØ42Ø          DECB
1Ø22 26   F5            ØØ43Ø          BNE     CYC3
                        ØØ431
                        ØØ44Ø * WAIT FOR HORIZ SYNC
1Ø24 7D   FFØØ          ØØ45Ø          TST     $FFØØ
1Ø27 7D   FFØ1          ØØ46Ø CYC5     TST     $FFØ1
1Ø2A 2A   FB            ØØ47Ø          BPL     CYC5
                        ØØ48Ø
                        ØØ49Ø * CHANGE PALETTE VALUES
1Ø2C BF   FFB8          ØØ5ØØ          STX     $FFB8
1Ø2F FF   FFBA          ØØ51Ø          STU     $FFBA
                        ØØ52Ø
                        ØØ53Ø * BUMP PALETTE VALUES
1Ø32 3Ø   89 Ø1Ø1        ØØ54Ø          LEAX    $Ø1Ø1,X
1Ø36 31   C9 Ø1Ø1       ØØ55Ø          LEAY    $Ø1Ø1,U
                        ØØ551
                        ØØ56Ø * STRIPES ARE 8 SCAN
                        ØØ57Ø * LINES HIGH
1Ø3A C6   Ø7            ØØ58Ø          LDB     #7
                        ØØ581
                        ØØ59Ø * COUNT NUMBER OF STRIPES
1Ø3C 4A                 ØØ6ØØ          DECA
1Ø3D 26   DA            ØØ61Ø          BNE     CYC3
                        ØØ611
                        ØØ62Ø * LAST STRIPE, GO DO IT
                        ØØ63Ø * AGAIN
1Ø3F 2Ø   C6            ØØ64Ø          BRA     CYC1
                        ØØ65Ø
          1ØØØ          ØØ66Ø          END     CYCLE

ØØØØØ TOTAL ERRORS
```

Taking a Look at How Monitors Work

By Tony DiStefano
Rainbow Contributing Editor

Well, I finally got my CoCo 3. The first thing I needed to do was to plug it into a monitor. In my computer room I have various color monitors, TVs and monochrome monitors. I read through the CoCo 3 manual and found out it has three ways of connecting a display to it. The first and most common is the RF output. This is where you connect an ordinary TV to it. The second is a composite color output, sometimes known as a video output. The third is an RGB output.

Now, most people are familiar with the RF output. Many people know about video outputs, but what is this RGB stuff? It is not new to me because I use an RGB monitor for my other computers. With the right connector, a piece of ribbon wire and the right information, I connected the CoCo 3 to my Sony RGB monitor.

Ever since I wrote an article on how to connect your CoCo to a monochrome monitor, I have been getting calls about it. So, with the coming of the CoCo 3, it is time to do an "everything you ever wanted to know about monitors but were afraid to ask" article. Here it is.

I am going to start from the basics and work my way up to RGB. Let's begin with some theory on a monochrome monitor. The mono part of that word implies one color. At first, all picture tubes were white. Then green was the "in" color and then amber became popular. Whatever the actual color of the tube, it is still one color, hence monochrome. A picture tube is made of glass. Inside this tube is a vacuum. On the inside surface of the display area there is a thin coating of phosphorus. One physical property of phosphorus is that when bombarded with electrons (high voltage electricity) it glows. Inside the back end (neck) of a picture tube there are circuits that shoot electrons at the phosphorus. The construction of the tube is beyond this article, but when it is on, a stream of electrons hits the phosphorus, and where it hits, the phosphorus glows. But alone, all that does is make a glowing dot in the center of the screen. Not much good.

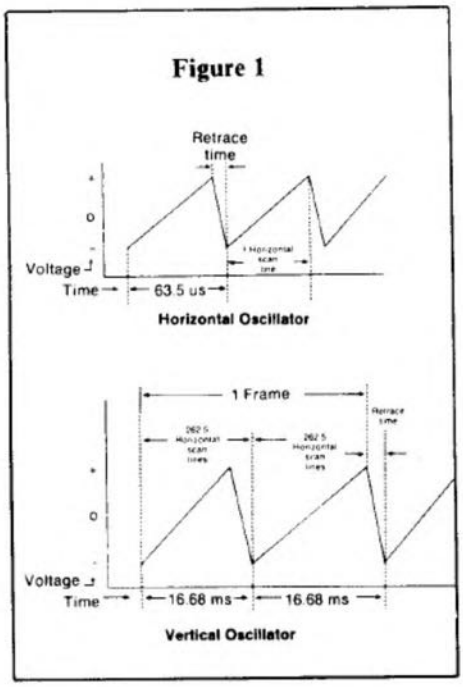
Since electrons are affected by a magnetic field, putting a magnet close to the tube will deflect our dot. The dot would move according to the strength and direction of our magnetic field. An electric current through a wire creates a magnetic field. The more current, the stronger the magnetic field. A length of wire wrapped in a coil is enough to deflect our dot anywhere on the screen. In most monitors, two coils of wire wrapped around the neck of the tube are used to move our dot around. One coil is positioned so that a varying amount of current makes the dot move sideways or horizontally. The other is positioned to give up/down or vertical motion.

Given the right amount of current and in the proper sequence, our dot now moves from right to left and from top to bottom, in the same motion as reading. Make that dot move fast enough and it appears to fill the screen with light, since phosphorus continues to glow for a short time after the dot has moved. Those lines you see on your screen are made by one moving dot.

So far, we have one moving dot that fills the screen with light. If, while moving this dot, you were to increase and decrease the number of electrons hitting the phosphorus, you would get varying amounts of light. The amount of light produced is directly proportional to the number of electrons hitting the phosphorus.

Things are shaping up to a picture. In a TV monitor, there are many signals and currents, one of which is called the horizontal oscillator. This circuit is connected to the coil that deflects the dot horizontally. Figure 1 shows the wave shape of the horizontal oscillator. It starts off negatively, deflecting the dot to the left. It increases linearly to a positive position, moving the dot smoothly across the screen. Then, it quickly jumps back to the original position. During this time the electron flow is cut off so that it will not appear on the screen. This time period is known as the retrace time, and the circuit that cuts off the electron flow is called a blanking circuit.

Figure 1

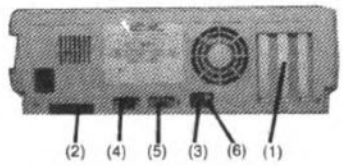


Another circuit in a TV is the vertical oscillator and yes, you guessed it, it controls the dot vertically. The wave shape of the vertical oscillator is basically, the same as the horizontal one, only much slower. Many horizontal cycles fit inside one vertical cycle (more on this later). The vertical oscillator also

SAVE $800

Reg 1999.00

$1199



Exellence in PC Business Computing

The Tandy 1000 is indicative of our pledge to making high quality affordable computers. Other PC compatibles don't offer you adapters for your peripheral devices or include a software program that may be all you need! Tandy 1000 is the basis for greater business efficiency.

25-1000

SPECIFICATIONS

Microprocessor: Intel 8088 processor. Clock speed, 4.77 MHz. Operating System: Includes Microsoft's advanced MS-DOS 2.11 with BASIC. **Memory:** 128K RAM, expandable to 640K. **Keyboard:** 90-key sculptured, including numeric keypad. Twelve programmable Special Function keys. **Video Display:** Optional high-resolution, non-glare 30.48cm monochrome (green) or color monitor. 80 or 40 characters per line by 25 lines. High-resolution monochrome and color graphics (displaying 8 colors selected from 16). **Disk Drives:** One built-in double-sided, double-density, 360K (formatted) thin-line 13-34cm mini-floppy. 48 tracks per inch. **Internal Expansion: (1)** Three user-accessible IBM PC-compatible 25cm card slots, second 360K Floppy Disk Drive. **External Connections: (2)** Standard parallel printer port. **(3)** monochrome monitor. **(4)** Light pen adapter, two joysticks. **(5)** RGBI Color Monitor, **(6)** composite video and audio. **Power:** 240 VAC, 50Hz.

Tandy 1000 with Monitor and Deskmate Software

SAVE $848.00

Reg 2347.00

1499

An offer that's unique since you'll probably never need to add to the Deskmate Software we've included here. It's a word processor, spreadsheet, inventory list and has a menu that greets you each morning with a calendar and appointment schedule. With a modem you'll be able to converse with other computers. The 30cm monochrome monitor is non-interlaced for sharp clear display.

25-1000/3211.

- ***The Complete Computer Package That Includes DESKMATE Software as a BONUS***
- ***Also Includes A Monochrome Monitor That's Phosphor Green For Easy On The Eyes Composing.***

WE SERVICE WHAT WE SELL!

Tandy ELECTRONICS

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

Nearly 350 Stores Australia-Wide

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

WHERE-IS-AS-IS-SALE SOFTWARE CLEARANCE

Deep price cuts on closeout, discontinued items and demos! NOT ALL ITEMS IN EVERY STORE – all subject to prior sale – MOST CARRY THE REGULAR TANDY LIMITED WARRANTY. Please don't blame our Managers if they don't have it, because a "Where-Is-As-Is-Sale" at 200 stores means that merchandise is liable to be almost anywhere. At these prices it's worth a bargain hunt – try all Tandy stores in your area.

SPECTRUM ANALYSER

The perfect program for the hi-fi buff! Colour bar graphs display the power dispersion of your system over 9 octaves. In ⅓ octave segments. Test your stereo's performance. 26-3156 ..... WAS 34.95 ..... NOW 19.95

COLOR LOGO LAB

This version of the "turtle graphics" program is designed for the classroom situation where children can learn computer programming under supervision. 26-2770 ..... WAS 289.95 ..... NOW 99.95

SUPER LOGO DISK

An updated version of the Color logo that will take your child through more advanced echelons of computer programming. "Turtle graphics" teaching method is unsurpassed. 26-2716 ..... WAS 199.95 ..... NOW 89.95

TYPING TUTOR

A fun way to learn to type or improve your current level of performance. This program includes a ten page instruction manual and self-tests. 16K min required. 26-7961 ..... WAS 34.95 ..... NOW 19.95

ATOM

This game is an educational tool designed to introduce your child to the exciting sub-atomic world of elements. The object is to build an atom out of the 54 elements. 26-3145 ..... WAS 39.95 ..... NOW 19.95

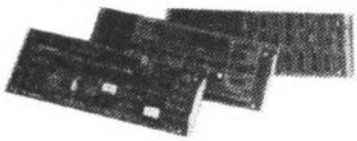
GOMOKU AND RENKU

A classic Oriental game that requires the player to place five men in a row. If you've got the patience and the time, this is an extremely worthwhile program to master. 26-3069 ..... WAS 49.95 ..... NOW 24.95

COLOR ROBOT BATTLE

By programming the two robots, you can learn about computer programming. Players are encouraged to challenge others, and thereby improve their programming skills. 26-3070 ..... WAS 69.95 ..... NOW 29.95

Let your Tandy 1000 grow



Memory Expansion Board. User-installable memory board expands your machine to 256K. Add additional memory if you choose and expand to 384K. Includes direct memory access controller. Was 499.95. 25-1004 **NOW 149.95**

Tandy 1000 Disk Drive Kit. Upgrade your Tandy 1000 with an additional 360K of storage. Mounts internally and installation is recommended (not incl.). Comes complete with full instructions. Was 399.95 24-1005 **Now 249.95**

Hard Disk Controller Board. Lets you add a hard disk to your Tandy 1000. Compatible with most 1000 software which transfers to hard disk. With cable for use with 10, 15 or 35-megabyte hard disks. Was 549.95. 25-1007 **Now 299.95**

Memory Expansion Board. Uses an expansion slot to upgrade a Tandy 1000 (with 384K of memory) to 512K. Also provides sockets for an additional 128K. For all Tandy 1000/PC compatibles. Was 399.95. 25-1009 **Now 149.95**

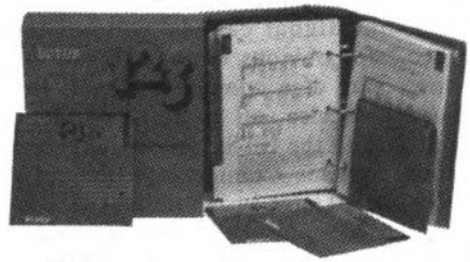
Digi-Mouse Controller/Calendar Board. Dual-purpose board includes controller for Digi-Mouse and perpetual time-date clock. Features a program and documentation on how to use Digi-Mouse. Was 299.95. 25-1010 **NOW 149.95**

Digi-Mouse. 2-Button Mouse. For fast and accurate cursor control. Simply plugs in. Great for graphics, text and other applications. Requires Mouse controller, calendar board. 26-1197 **149.95**

Sorry no rainchecks on these items.

Business Applications Software

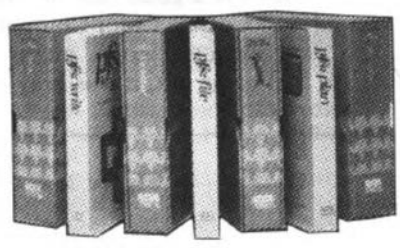
46.6% OFF!



The Remarkable Lotus 1-2-3 Software

One of the most widely used programs. Go from spreadsheet to graphics to information management. Also sorts data from your spreadsheet. Reg 1499.00. 25-1145 **Now 799.95**

SAVE MORE THAN **50% OFF!**



PFS: file. Filing applications.	Was 229.95	25-1140	**Now 99.95**
pfs: report.	Was 229.95	25-1141	**Now 99.95**
pfs: write. Advanced writing	Was 229.95	25-1142	**Now 99.95**
pfs: graph. Quality graphics	Was 229.95	25-1143	**Now 99.95**
pfs: plan. Spreadsheet	Was 229.95	25-1144	**Now 99.95**
Friday. Mailing and filing	Was 749.95	25-1149	**Now 399.95**
MS-Word. Word processor.	Was 749.95	25-1153	**Now 399.95**
Scripsit 1000/3000.	Was 699.95	25-1153	**Now 399.95**
Demon Attack.	Was 54.95	25-1110	**Now 29.95**

Sorry no rainchecks on the items.

has retrace time and vertical blanking circuits. Due to its nature, one horizontal cycle is called a scan line, and one vertical cycle is called a frame.

When our dot is not doing horizontal retrace or vertical retrace it appears on the screen. This is known as active video. It is during this time that our dot gives the viewer useful information. This information can be a picture like ordinary TV, or computer generated characters. In either case, the video signal is proportional to the brightness of the picture. A higher signal produces a brighter dot and a lower signal produces a softer dot.

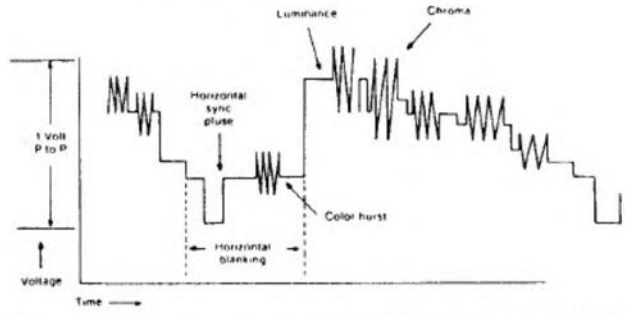
In order for a picture to appear on a video monitor, three signals are needed; horizontal, vertical and video signals. It is not efficient to run three signals and a ground return to a TV receiver or monitor. A method was developed to combine these three signals into one. Instead of supplying complete horizontal and vertical wave shapes, the source need only send a pulse signifying the start of every horizontal line and the start of every vertical frame. These pulses are known as sync pulses. The rest of the wave shape is then regenerated inside the monitor. It is then up to the monitor to make sure that the internal horizontal and vertical oscillators keep up with the sync pulses.

These sync pulses and video signals are mixed together in a specific way to form one signal called "composite video," for obvious reasons. Figure 2 shows part of a composite signal. In North America, all composite video conforms to the NTSC (National Television Systems Committee) standard; more on that later. In a monitor, circuits are made to separate the video information from the sync signals, and are then translated to drive currents that connect to the coils and the picture tube.

Up till now, I've been talking about monochrome (black and white) pictures. But, there is a good reason why Tandy calls our CoCo a Color Computer — it can display a color picture. When TV first came out, it was only in black and white. When color came out, a method had to be developed so that a color signal would be compatible with a black and white TV.

It was up to the NTSC to develop a composite signal that would carry the extra color signal and still be compatible with the older black and white signal. In 1953, the NTSC established the color television standards. In these standards, the signal is to have 525 line interlaced scan. The horizontal scan frequency is 15.734 kHz; the vertical frequency is 59.94 Hz. The color information is contained in a 3.579545 MHz subcarrier. The phase angle of the subcarrier represents the color, and the amplitude of the carrier represents the saturation. Figure 3 shows one horizontal line. Notice the color frequency burst just after the horizontal sync. The phase difference between this reference burst and the actual signal describes what color that particular part of the screen should be. The amplitude of the color signal represents how much of that color to put on the screen.

Figure 3: One Horizontal Line



On the monitor side, color is quite complicated to reproduce. You have to start with a completely new tube. Instead of smooth monochrome phosphorus, the tube has to be striped with alternate red, green and blue phosphorus. The smaller the stripe the better the picture quality.

When a color composite video signal enters a color monitor, it is first stripped of its sync signals, and then the monochrome (called luminance) and color signals (called chroma) are split into three signals, the red content, the green content and the blue content. Three separate electron beams are used to display the three colors on the screen. The beam carrying the red content has to hit all the red strips. The green hits the green strips and so on. If a beam that has red information hits any other color than red, a wrong color results. It requires a lot of electronic circuits to keep this from happening. That is not the worst part; the color frequency carrier is 3.58 MHz. In order to isolate the color carrier from the monochrome signal, a filter is used that removes any frequency higher than 3.58 MHz. This seriously limits the resolution of a color signal. In fact, the resolution of a color signal, at absolute best, is about 400 lines. That is OK for the CoCo and CoCo 2 but is not good enough for the CoCo 3.

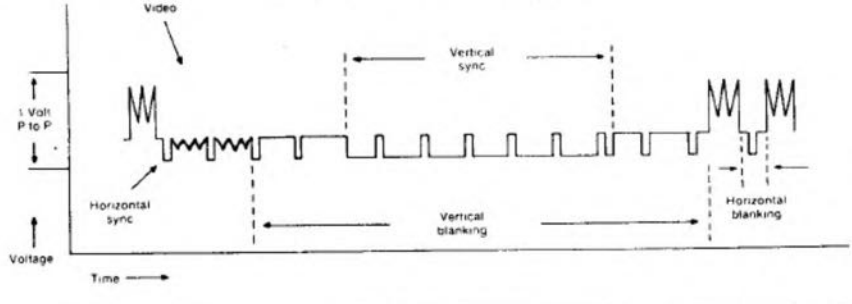
When you put a color signal in a monochrome monitor, the color information shows up as dots on the screen. Figure 3 shows that. The frequency of the color signal is 3.58 MHz. A monochrome monitor with a 20-MHz bandwidth has no filter to remove the color carrier. The monitor will have no problems displaying the color carrier — as an annoying monochrome mess of dots.

Now comes the CoCo 3. It has a resolution of 640-by-192. That is very nice but have you ever seen a 640-by-192 screen on a regular composite monitor? Believe me, it's not a pretty sight. What is Tandy to do? The only reasonable thing is to get rid of that color carrier and put out the color information separately. Now that is a great idea and for once Tandy did it right! The CoCo 3 has an output known as an RGB output. That's right, RGB stands for Red, Green and Blue. No color carrier, no filters and no sync pulses, just clean color.

Wait a minute, that won't work without sync pulses. So Tandy added some

continued on page 31

Figure 2: A Composite Signal



Taking a Look at How Monitors Work

continued from page 30

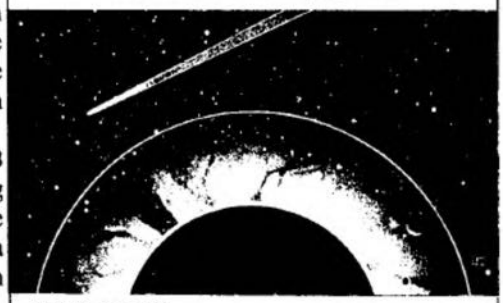
more lines and added sync pulses. In fact, the CoCo 3's RGB output is the best color picture ever for a CoCo! The clarity is limited only by the resolution of the monitor.

You don't have to have a Tandy RGB monitor. However, if you plan on going out to buy a brand X monitor at some discount mail order house, here are a few tips to help you get started with fewer headaches.

First, when you select a model you want (or can afford), make sure that it is an RGB analog monitor with negative or composite sync (like my Sony) with a horizontal frequency of 15.7 kHz and a vertical frequency of 60 Hz. Also make sure that the bare connector to the monitor is available. You will also need a connector for the CoCo 3 side of it. That requires a 10-pin female socket connector for flat ribbon cable. And don't forget to get three or four feet of 10-conductor flat ribbon cable. Use the pinout supplied in the CoCo 3 manual and match the pinout of the RGB monitor manual to it. Now plug it in and watch it go!

Let the Laser Battle Begin

continued from page 15



```
179Ø LL=9Ø
18ØØ PMODE3,1:SCREEN1:POKE 65314
,248
181Ø R=RND(14Ø):X=22Ø
182Ø RETURN
183Ø ' ENEMY SHOOTS AT SHIP
184Ø PMODE3,1:COLOR 3,1
185Ø LINE(X,R+7)-(5,R+7),PSET
186Ø PLAY"T1ØØ;L255;CCCCACCCCA"
187Ø COLOR 1,Ø:LINE(X,R+7)-(5,R+
7),PSET
188Ø IF ABS((Y+3)-R)<6 THEN GOSU
B 1Ø8Ø
189Ø SN=RND(21Ø)
19ØØ RETURN
191Ø ' END OF GAME
192Ø FOR T=1 TO 5Ø
193Ø CIRCLE(127,96),T*2,,.8
194Ø PMODE4,1:SCREEN1,1
195Ø PLAY"O2;L"+STR$(T)+";G"
196Ø CLS4
197Ø PRINT@75,"game over";
198Ø PRINT@36Ø,"YOUR SCORE";SC;
199Ø PRINT@394,"WAVE NO.";W;
2ØØØ NEXT T
2Ø1Ø PRINT@485,"PRESS ANY KEY TO
 PLAY";
2Ø2Ø IF INKEY$=""THEN 2Ø2Ø ELSE
RUN
```

Understanding Relationships Between Fractions, Decimals and Whole Numbers

continued from page 17

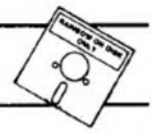
```
Y AGAIN !!";:FORI=1TO1ØØØ:NEXTI:
PRINT@422,STRING$(22,32);:GOTO37
Ø
41Ø NR=NR+1
415 IF B>C THEN SL=291:GOTO43Ø
42Ø IF B<C THENSL=323:GOTO43Ø
425 IF B=C THEN SL=355:GOTO43Ø
43Ø NC=NC+1:PRINT@418,"VERY GOOD
!  THAT IS CORRECT!";:PRINT@483,
"PRESS <ENTER> TO CONTINUE!";
435 X$=INKEY$
44Ø PRINT@SL,AR$;
445 FORI=1TO1Ø:NEXT:PRINT@SL,"
";:FORI=1TO1Ø:NEXT
45Ø IFX$="@"THEN465
455 IFX$<>CHR$(13)THEN435
46Ø GOTO29Ø
465 CLS:PRINT@1Ø1,"YOU TRIED"NC+
NW"PROBLEMS &":PRINT@165,"ANSWER
ED"NC"CORRECTLY"
47Ø PRINT@229,"WHILE DOING"NW"WR
ONG."
475 NQ=NC+NW:IF NQ=ØTHEN NQ=1
48Ø MS=INT(NC/NQ*1ØØ)
485 PRINT@293,"YOUR SCORE IS"MS"
%."
49Ø PRINT@357,"ANOTHER TRY (Y/N/
C) ?";
495 X$=INKEY$:IFX$="Y"THEN RUN
5ØØ IFX$="N"THENCLS:END
5Ø5 IFX$="C"THEN29Ø
51Ø GOTO495
```

CORRECTION

Mission: Hold the Bridge

Apparently we fouled up the line numbering - it was all there, only slightly re-arranged. Sorry for the inconvenience.

```
1ØØ TX(WT)=TX(WT)-TS:IF TX(WT)<=
Ø THEN TX(WT)=Ø:M=M-1:TS=WT:GOTO
 26Ø
1Ø5 IF MF=1 THEN GOTO 11Ø ELSE I
F MF=Ø THEN 5Ø
11Ø ' FIRE ROUTINE
115 X=X+6
12Ø H=Ø+X*MA/36Ø
125 V=16Ø-SIN(X/57.29578)*(16Ø/2
)
13Ø H=INT(H):V=INT(V)
135 PSET(H1,V1,5)
14Ø PRESET(H,V)
145 IF V>161 THEN X=Ø:PSET(H,V,5
):MF=Ø:GOTO 25Ø
15Ø H1=H:V1=V
155 GOTO 9Ø
16Ø GOTO 16Ø
165 ' DRAW, DEFINE, & GET TANK
17Ø PMODE 4,1:PCLS 5:SCREEN 1,1
175 DRAW"BM5Ø,5ØC8S4R7ER4FDGL4HU
BD2RL4GLG2E2RDLGDFR15EUHL15R14UR
F2H2LHL4BDBR5BL14BDRFDGR4HUERFDG
R3HUERFDGR4HUEC5
18Ø GET(5Ø,45)-(76,57),T,G
185 ' SET UP TANK X COORDINATES
19Ø TX(1)=255+6
195 FOR N=2 TO 7
2ØØ TX(N)=TX(N-1)+2Ø
2Ø5 NEXT N
21Ø IF RE=1 THEN RE=Ø:RETURN  EL
SE 22Ø
215 REM 'SOFTWARE CO. TITLE
22Ø IF SS=1 THEN PCLS5:GOTO 225
ELSE IF SS=Ø THEN PCLS5:GOTO 285
225 FORC=1TO2ØSTEP3:LINE(Ø+C,Ø+C
)-(255-C,191-C),PRESET,B:NEXT C:
DRAW"C8S24BM3Ø,5ØU3R3D3L3U3BFDRU
LRBEBRD2FREU2LD2LU2LBR4R3DL2RDLR
2DL3U3R3BRR2FGFLHDLU3RBDRBR2BUR3
DL2R2D2L3UR2L2U2BR4R3DL2RDLR2DL3
U3R3BRBDD2RURDRU2HLGBRRBEBRR3DL2
R2D2L3UR2L2U2"
23Ø DRAW"BM3Ø,55R3DL2R2D2L3UR2L2
U2R3BRR3D3L3U3BFDRULBUBR3R3DL2RD
LDLU3R3BRR3DLD2LU2LUR3BRD3ERFU3L
DLULBR4BDD2RURDRU2HLGBRRBR2UR2FG
FLHDLU2BRRBUBR2R3DL2RDLR2DL3U3"
235 DRAW"S16BM8Ø,14ØR2DL2DU2BR3R
2DLFHLDU2BR3R2L2DRLDR2BRR2UL2UR2
BRR2L2DRLDR2BRU2F2U2BRR2LD2BR2R2
UL2UR2"
24Ø IF PEEK(6528Ø)=126 OR PEEK(6
528Ø)=254 THEN PCLS5:GOTO 285 EL
SE 24Ø
245 ' HIT OR MISS ROUTINE
25Ø IF H>TX(WT) AND H<TX(WT)+26
THEN K=K+1:GOTO 26Ø ELSE FOR Y=1
 TO 4:CIRCLE(H,V),Y,Ø,1,.5,1:NEX
T Y:MF=Ø:FORY=1 TO 4:CIRCLE(H,V)
,Y,5,1,.5,1:NEXT Y:MF=Ø:X=Ø:GOTO
 5Ø
255 ' DESTROY TANK
26Ø FOR B=1 TO RND(6):PUT(TX(WT)
,TY)-(TX(WT)+26-B,TY+12-B),T,PSE
T:NEXT B:IF TX(WT)=Ø THEN LINE(Ø
,188)-(63,191),PSET,BF:LINE(TX(W
T),TY)-(TX(WT)+26,TY+12),PSET,BF
265 TS=WT:SS=Ø
27Ø TX(WT)=255:SS=Ø:WT=WT+1:GOSU
B 385
```



KISSable OS-9

Debunking the Myth of OS-9 User Hostility

By Dale L. Puckett
Rainbow Contributing Editor

Since January's RAINBOW is dedicated to beginners, we'll take a fresh look at the basics of OS-9 this month and try to eliminate some of the apprehension that surrounds Color Computer owners trying to use OS-9 for the first time. We'll wrap up the column with a few tips submitted by readers.

Myth Versus Reality

OS-9 is the victim of a myth in Color Computer circles. Those who perpetuate that myth would have you believe that OS-9 is difficult to use and impossible to understand. Not so!

The ironic thing about this myth is the fact that it most likely exists because of the tremendous computing power built into OS-9. Many options come with this power; options that give you the opportunity to make many choices.

Many people become overwhelmed when they face too much freedom or too many choices. They think that just because a choice exists, they have to make it. They lose sight of the fact that no action — or choice — is always an option. A professional operating system like OS-9 usually delivers a popular default action when you decide not to make a choice.

To illustrate our point we will try to give several examples that compare OS-9 to the Disk Extended BASIC used by the Color Computer. To do this, we'll show you how to do a few trivial tasks using several different tools. For example, let's imagine we want to print a message on our Color Computer screen. With Disk Extended BASIC we could type a simple command to do the job immediately after we turn on our computer.

```
PRINT "HELLO, I'M A COLOR
COMPUTER!"
HELLO, I'M A COLOR COMPUTER
```

As soon as we type the command line, our Color Computer will print the second line on the screen. Disk Extended BASIC is a language and one of the verbs in that language is PRINT. Each time Disk Extended BASIC sees the word PRINT it looks at the string of characters or variable following that word and prints it.

If you wanted to print the same line on the screen several times, you could type in a short program and run it.

```
10 FOR X = 1 TO 10
20 PRINT "HELLO, I'M A COLOR
COMPUTER!"
30 NEXT X
RUN
HELLO, I'M A COLOR COMPUTER
HELLO, I'M A COLOR COMPUTER
```

BASIC09 is another computer language that runs under the OS-9 operating system. Using an advanced design, it compiles each line of your program into an intermediate or I-code as you type it in. As a result it can run your programs four to five times faster than Disk Extended BASIC. Additionally, programs written in BASIC09 are much easier to read and understand because they do not require line numbers and they let you use long variable names that convey a meaning. Reading a BASIC09 program is almost like reading the solution to a problem written in English. For example:

```
FOR LINE := 1 TO 10
  PRINT "HELLO, I'M A COLOR
  COMPUTER!"
NEXT LINE
```

When we run this program, BASIC09 produces the same results on our screen as the Disk Extended BASIC program. If we wanted a stand-alone program, we could write the same program using the C language compiler that runs under OS-9.

```
main( )
{
 int line;
 for(line = 0; line != 10;
 line++)
 printf("HELLO, I'M A COLOR
 COMPUTER! \n");
}
```

"Just think of an operating system as a traffic cop on a busy corner . . . OS-9 directs the flow of information inside your computer."

Disk Extended BASIC, BASIC09 and C are all languages. While the syntax of each language is a bit different, the result is the same. Each language lets you do the same job.

OS-9 is not a language. Rather, it is an operating system. And the many utility commands that come with it let you do many things. For example, the OS-9 echo utility lets you simulate the first Disk Extended BASIC PRINT command line above. Even though it is a command that tells the OS-9 operating system to do something, it delivers a result just like the PRINT verb in Disk Extended BASIC. Try it. At the OS-9 prompt, type:

```
OS9: echo HELLO, I'M A COLOR
COMPUTER ENTER
HELLO, I'M A COLOR COMPUTER
```

Note that we did not type the exclamation point in our example. It is a special character in an OS-9 command line and is used to set up a pipeline. We'll talk about pipelines later.

By repeating the echo command line above 10 times in an OS-9 procedure file we could even simulate the other programs. However, since the OS-9 Shell is not a complete programming language and does not allow FOR-NEXT loops we would need to type the echo command line 10 times. Let's try it!

```
OS9: build TenTimes ENTER
? load echo ENTER

? echo Hello, I'm A Color
Computer ENTER
? echo Hello, I'm A Color
Computer ENTER
? ...
? unlink echo ENTER
? ENTER

OS9: TenTimes ENTER
Hello, I'm A Color Computer
Hello, I'm A Color Computer
...
OS9:
```

It may not be elegant. But, it works and serves to illustrate the point that OS-9 is not really that complicated. It's just another way of getting a job done.

OS-9 Can Do Graphics Too!

The OS-9 module that writes to your Color Computer screen can also do graphics. Just to prove the old adage that the more things change, the more they stay the same, let's look at three ways to draw a box near the edge of your CoCo graphics screen. We'll use Disk Extended BASIC first.

```
10 PMODE 1,1
20 PCLS
30 SCREEN 1,1
40 LINE (10,10) - (10,176),
PSET
50 LINE (10,176) - (240,176),
PSET
60 LINE (240,176) - (240,10),
PSET
70 LINE (240,10) - (10,10),
PSET
80 GOTO 80
```

As an alternative, we could have replaced lines 40 though 70 with Line 40 below. But, we wanted to keep our example programs parallel.

```
40 LINE (0,0) - (255,191),
PSET,B
```

To write the same program in BASIC09 we would use the following lines:

```
run gfx("mode",0,4)
run gfx("clear")
run gfx("color",7)
run gfx("move",16,16)
run gfx("line",16,176)
run gfx("line",240,176)
run gfx("line",240,16)
run gfx("line",16,16)
```

Note that when you work with graphics under OS-9, the module named grfo must be available in memory or in your current execution directory. We'll talk more about modules in memory and the OS-9 directories later. For now, you should know that the gfx in the BASIC09 program lines above is the name of a BASIC09 module that interprets the high-level language parameters within parentheses and sends out the proper control codes to the OS-9 screen driver module. The move command above places OS-9's invisible graphics cursor at a position 16 pixels up from the bottom of the screen and 16 pixels to the right of the screen's left edge. We drew our box in from the edge of the screen so it would be easier to see it on a monochrome monitor. All of this implies that you should be able to send these codes to the screen driver module from the OS-9 command line, and you can.

The easy way to draw the box above using OS-9 would be to use a series of command lines containing the display utility in a procedure file.

```
OS9: build box ENTER
? load grfo
? load display ENTER
? display F 0 4 ENTER
? display 10 4 ENTER
? display 11 7 ENTER
? display 15 10 10
? display 16 10 B0 ENTER
? display 16 F0 B0 ENTER
? display 16 F0 10 ENTER
? display 16 10 10 ENTER
? unlink display ENTER
? ENTER

OS9: box ENTER
```

You could also type display followed by each of the hexadecimal numbers used above in one command line. We formatted the OS-9 procedure file the way we did to make it parallel in structure to the BASIC09 program. The line with display 15 10 10 places the invisible cursor used by the OS-9 graphics driver 16 pixels up and 16 pixels to the right of the lower left-hand corner of the OS-9 graphics screen, just like the move command in the BASIC09 program. Notice also that the OS-9 display command uses hexadecimal numbers, while the BASIC09 program used decimal numbers. The Location B0 is 176 pixels up from the bottom of the screen. The Location F0 is 240 pixels to the right of the left edge of the screen. Notice how these numbers compare to the decimal numbers in the BASIC09 and Disk Extended BASIC programs.

```
OS9: display F 0 4 10 4 11 7 15
10 10 16 10 B0 16 F0 B0 16 F0 10
16 10 10
```

You can also use the OS-9 display utility with the proper control codes to move the graphics cursor, set individual pixels on the screen to a specific color, draw circles and paint an area of the screen with a specific color.

Now that we have shown you that OS-9 is merely another way to give directions to your Color Computer, we'll move on to show you how to get started using OS-9. Hopefully, we'll be able to show you how to avoid a few trouble spots along the way.

OS-9 is an Operating System

Before we move on we should take a few moments to talk about operating systems. In the language of a systems designer, an operating system controls the low level processes within your computer. It gives your applications programs a way to talk to and control your hardware. It also manages your memory and other finite resources within your computer.

These processes are nothing more than short programs that happen to be running and doing a job for you. They may be putting characters in a disk file or they may be sending a series of characters to a printer. Your system resources include external devices like your terminal, printer and disk drives. They also include things inside your computer like memory and the microprocessor's time.

Just think of an operating system as a traffic cop on a busy corner. The traffic cop directs the flow of automobiles on a busy street. OS-9 directs the flow of information inside your computer, making sure that the right data gets to the right place at the right time.

Getting Started With OS-9

There are two ways to bring OS-9 to life on your Color Computer. The method you use depends on which Disk

Extended BASIC ROM is in your CoCo. If you have Disk Extended BASIC 1.0 then you must use the special OS-9 boot disk that comes with your OS-9 system disk. You must insert it in Drive 0 and type RUN"*". A few seconds later the screen will instruct you to insert the OS-9 master disk in Drive 0 and press any key to continue.

If you have a Color Computer with Disk Extended BASIC 1.1, then all you need to do is put the OS-9 master disk in Drive 0 and type DOS, a Disk Extended BASIC command that does the same thing as the RUN"*" command above.

After typing DOS, OS-9 will ask you the date and time. OS-9 uses the date and time when it saves a file. And if you forget the date or want to know what time it is, OS-9 can retrieve that information and print it on the screen. If you are using versions 1.00 or 1.01 of OS-9, setime, the OS-9 utility command that asks you for the time must be run each time you start OS-9. It starts the pseudo clock that runs within OS-9. If this clock is not running, OS-9 is not able to run multiple tasks.

If you want to know the current date while running OS-9, you can ask for it by using the date utility command that comes with your system. Simply type: OS9: date and press ENTER. OS-9 will print the date on your screen for you. If you need to know the time, you can ask OS-9 to print that by typing OS9: date t and pressing ENTER.

If you type this command right after you start the Color Computer, OS-9 will look in its module directory for a module named date. Since, you haven't loaded that module, OS-9 will not find it. Because it could not find the module in memory, OS-9 will look for it in its current execution directory. This directory is almost always /d0/CMDS on standard OS-9 systems. If you haven't deleted the file named date from your disk, OS-9 will find it, load a module named date into memory and execute it. After it does all of this, you will see the date appear on the screen.

If you knew you were going to ask for the date and time every few minutes, you could load the date command in memory. After you do this, it will appear to run instantly (OS9: load date). In fact, you can load a number of program modules into memory and have them available instantly. However, if you try to load too many programs into the crowded 64K workspace available with OS-9 Level I, you will quickly run into a problem — you won't have enough memory left in your computer to run the large programs that do most of your real work. BASIC09, for example, takes up approximately 22,000 bytes of memory.

All of this means you have to write your programs in a number of small modules while programming with BASIC09 on an OS-9 Level I system. You must then load each module from a disk when you need it. Modularity is a big plus for BASIC09, but loading a module from disk every time you need to run it can be a big pain.

OS-9 Level II Solves Many Memory Problems

If you already own a CoCo 3 with 512K of memory, rest easy. When OS-9 Level II becomes available, many of the problems feeding the myth that OS-9 is hard to use and understand will disappear. I have been running OS-9 Level II with only 240K of memory for more than four years and I have never run into a problem.

"Modularity is a big plus for BASIC09, but loading a module from disk every time you need to run it can be a big pain."

OS-9 Level II helps solve memory problems by setting up a separate 64K of workspace for each task running on your computer. Let's briefly compare OS-9 Level I and OS-9 Level II.

With OS-9 Level I you can only access 64K of memory. Part of this is due to the fact that the 6809E processor in your Color Computer can only access 64K of memory. The rest is due to the fact that OS-9 Level I does not know about memory management beyond the 64K boundary. Enter OS-9 Level II.

OS-9 Level II works with two different types of workspace. For starters, it sets aside a system workspace that holds all of the device descriptors and device drivers you need to access your hardware as well as the file managers, kernel and other internals that OS-9 needs to manage your computer's resources. And here's another bonus — that same system workspace is used for all the data memory needed by OS-9.

After setting up its own system workspace, OS-9 Level II then sets up a separate workspace for each process you start. Remember, a process is simply one of your programs that happens to be running. The important fact to remember is that each of these user workspaces can be up to 64K long. Let's get specific and show what this really means to you.

What happens when you run BASIC09 from within OS-9 Level I? In a few words, memory space is cramped and you have very little room for programs. For example, in my system I have 145 pages, or just over 36K, of memory free immediately after I start OS-9. After loading BASIC09, I have 55 pages, or just over 13K, of free memory left. That doesn't give me a lot of space to work.

Now, let's see what happens when I run BASIC09 with OS-9 Level II. After loading BASIC09 it just sits there in memory. It will be available almost immediately, but it won't be using any data workspace until I start it as a process by running it from the command line. When I do run it, I will have nearly 64K, or 256 pages, of memory available for it and the programs it is running. This means BASIC09 will be able to use approximately 64K-22K, or nearly 42K, of memory for its programs and data. In reality, there is closer to 63.5K of memory available in each workspace because the top memory locations are mapped to the hardware. But in any case, 42K of memory is much better than 13K.

Making a New OS-9 Disk

One of the most important lessons you can learn when you start to work with OS-9 is to always work with a copy of the master disk you purchased from Tandy. Never work with the original disk because accidents can happen — even to experienced hackers. So when you first run OS-9, make a backup of that precious master disk. It isn't hard to do and it shouldn't take too much time. Think of the time as a valuable investment.

First, you will need to format a new disk. To begin, type OS9: load format free and press ENTER. Now, take the OS-9 system master disk out of Drive 0, insert the new disk in the drive and type OS9: format /d0 and press ENTER.

The format program asks you if you really want to format the disk in Drive 0. After you confirm that the disk in Drive 0 is indeed your new disk and not your master system disk, press Y for yes and the format program will go to work.

Format will then ask you to name the new disk. You can make up any name here because that name will be replaced when you back up the master system disk on the new disk. After you type the

name, format will verify the data it placed on the new disk and you will soon see the OS-9 prompt again. When you do, type OS9: free /d0 and press ENTER.

The free utility command should report that you have a total of 630 sectors on the disk and that 620 sectors are available to use. If format found any bad sectors on the new disk it will report a number smaller than 630. If this happens, do not attempt to back up the system master disk on to it. The OS-9 backup utility requires that the format of the disk you are using to hold the backup is formatted exactly like the original disk you are backing up. If the two formats are not identical, the backup command will not work. Don't even try.

Assuming that your new disk is good, take out the new disk and put the master system disk back in Drive 0. Then type OS9: unlink format free and press ENTER.

This command line removes the two utility programs you loaded earlier and frees more memory for you to use when you do the actual backup. Now type OS9: load backup and press ENTER.

After backup is loaded you must take the original master disk out of Drive 0 and put the newly formatted disk in that drive. Then type OS9: backup s /d0 #32K and press ENTER.

When the OS-9 backup utility asks if you are ready to back up from /d0 to /d0 answer with a Y for yes. The program will instruct you to get the destination disk and press any key. Since you already have it in the drive, go ahead and strike any key. OS-9 asks you if it is OK to write over the disk in the drive. Again, press Y for yes.

You'll then be asked to get your source disk ready. Remove the backup disk from the drive, insert the master system disk and then press any key. You will have to repeat the steps above several times, swapping the disks when the backup program asks you to, until the original disk is completely copied onto the new disk.

Now that you have a new copy of the system master disk you should store the original master disk in a safe place. The important thing to remember about backup is that you can only back up a disk to another disk of identical size. You cannot back up up a 35-track disk to a 40-track drive. Nor can you back up a single-sided, 40-track disk to a double-sided, 40-track disk.

To back up all the files on a disk of one format onto a disk formatted differently, you must use the OS-9 dsave utility command or one of the many alternative copy commands from third-party vendors. Without going into a lot of detail, here is an OS-9 command that will do the job: OS9: dsave /d1 /d0 ! Shell and press ENTER.

You'll notice that this command assumes you have two disk drives in operation. After you have used OS-9 a few hours, you will discover that two disk drives are indeed a necessity — not a luxury. The exclamation point in the command line above causes dsave to send its output to the OS-9 command interpreter, which is named Shell. It is an example of the pipelines we mentioned earlier.

Listing 1: fixtime

```
**************************************************
*
* FIXTIME  -  COPYRIGHT (c) 1986 by S. B. GOLDBERG
*
* Updates time to help correct clock for disk usage.
*
* Counter update is $49 from start of module.  Use
* Debug to change timing count, if necessary.  In-
* crease the count to slow the clock, decrease the
* count to speed up the clock.  Check clock operation
* and change in source code when it keeps good time
* and re-assemble object module.  Do NOT unlink fixtime
* while testing, or you'll have a total system crash.
*
* Fixtime can't be unlinked if part of OS9Boot file.
* Use OS9Gen to add fixtime to boot file and execute
* from your startup file.  Do NOT use Cobbler after
* executing fixtime, the CRC and header will not be
* correct!!!
*
         ifp1
         use   /dØ/defs/os9defs
         endc
*
****************************************
* SET COUNT TO KEEP YOUR CLOCK ON TIME *
****************************************
count    equ   8    disk access count
vector   equ   $Ø1ØA   NMI vector address
*
         mod   len,name,prgrm+objct,reent+1,entry,dsiz
*
         rmb   2ØØ    stack
dsiz     equ   .
*
name     fcs   /fixtime/
         fcc   /(c) 1986 S.B.GOLDBERG/
********************************
*
* INITIALIZE AND QUIT
*
entry    ldx   vector   get NMI vector
         stx   1+jump,pcr   save it
         leax  fix,pcr   time correction address
         stx   vector   put in NMI vector address
         lda   #$3d   new entry offset
         sta   name-3,pcr   put in header
noerr    clrb   clear error flag
         os9   f$exit   quit
********************************
*
* THIS DOES THE ACTUAL WORK
*
counter  fcb   count   disk access counter
fix      dec   counter,pcr   update time?
         bne   jump   not yet
         lda   #count   yes, counter value
```

Customizing Your Disks

One of the most important advantages of OS-9 is the fact that it lets you customize your system to your heart's content. Unfortunately, this ability also makes a tremendous contribution to the myth that OS-9 is difficult to use and hard to understand.

Make the pledge right now to stick with the basics until you are ready to start modifying your system. Practice running the utility commands that are stored in the /d0/CMDS directory of your working system disk. Follow the directions in the OS-9 manuals or *The Complete Rainbow Guide To OS-9* carefully. After you understand what is happening when you run each command, you can move forward freely and modify your computer as you like.

Many of the problems you'll encounter if you are running OS-9 Level I revolve around the severe memory constraints forced on you by the limited 64K workspace. In fact, many of the error messages you receive when you start to work with OS-9 procedure files will occur because there is simply not enough memory to load in the module required to do a specific task.

Another error message you may see quite often at first is Error 216 — File Not Found! This error pops up a lot for beginners because they do not fully understand the OS-9 file system.

The important thing you must understand is that OS-9 always maintains two working directories. One of these directories is called the current execution directory. The other is called the current data — or working — directory.

OS-9 always looks in the current execution directory when it is looking for a file that contains a program it needs to run. Likewise it usually stores all data files and looks for procedure files in the current data directory.

Here's the trick. The current execution directory and current data directory are seldom stored in the same physical position on two different disks. This means that even though these directories may have the same names on both disks, they are often not located on the same track or sector. Because of this, OS-9 will not be able to find your current directories if you swap disks without telling it.

Here's the solution. If you remove one disk from a drive and insert another, always type:

```
OS9: chd /d0/MyDirectory ENTER
OS9: chx /d0/CMDS ENTER
```

Note that the directory names in these two command lines are simply exam-

```
          sta   counter,pcr   reset counter
          leax  pack,pcr   address for time package
          os9   f$time   get time
          lda   #6Ø   time check constant
          inc   5,x   add a second
          cmpa  5,x   totals minute or more?
          bhi   newtime   no, set new time
          clr   5,x   yes, make seconds zero
          inc   4,x   add one minute
          cmpa  4,x   hour or more?
          bhi   newtime   no, set new time
          clr   4,x   yes, zero minutes
          inc   3,x   add an hour
newtime   os9   f$stime   set new time
*******************************************
* You MUST use '>ØØ' to force extended address
*
jump      jmp   >ØØ   goto NMI handler
*
*******************************************
pack      fcb   ØØ,ØØ,ØØ,ØØ,ØØ,ØØ
          emod
len       equ   *
          end
```

Listing 2: reboot

```
/*
 * OS9 ReBoot
 *
 * Copyright 1986 by R.M. Santy
 */
#include <stdio.h>
#include "module.h"
#include "os9defs.h"

#define SECTOR 256
#define SECS_TRACK 18
#define TRACK_SIZE (SECTOR * SECS_TRACK)
#define BOOTSIZE TRACK_SIZE
#define TRACK34 ((long) TRACK_SIZE * 34)

#define SYSERR -1

/*
  Track 34 of boot disk is loaded
  into the following buffer.
 */
char bootstrap[BOOTSIZE];
/*
  If a hard disk has a bootable partition,
  the byte starting byte address is placed
  in the following 24 bit buffer.
 */
char offset[3];
/*
  Any failure causes the following message
  to be displayed.
 */
char *usage = "Usage: ReBoot /devname\n";
/*
  The device descriptor's address is copied
  here.
 */
mod_dev *device;
/*
  The device driver's address is copied
```

ples and you need to type the names of the actual directories stored on the disk you have inserted.

Hopefully, we have given you enough information to get you started and pointed out a few of the pitfalls to avoid when you first start running OS-9. Hang in there and practice. Stick with the simple utilities until you thoroughly understand what is happening when you run them. After you conquer a command, move on to another. Before long, you'll be able to control your Color Computer like you never could before.

And, if the Not Enough Memory Errors are driving you crazy, just remember that OS-9 Level II and 512K of usable memory at your fingertips is just around the corner with the new CoCo 3. I have a hunch that the new visual shell that Microware and Tandy are developing will bring intuitive computing and applications to OS-9. Things should really be fascinating in about a year.

CoCo SIG Database Expanding

The OS-9 database in RAINBOW's CoCo SIG on Delphi is really expanding. I understand that the complete library of the OS-9 Users Group should be available soon. And even without the Users Group Library, the number of files has expanded dramatically in the last few months.

You can now find beginners' tutorials and help with downloading files in a series of excellent articles in the database. In addition to the articles, you will find several dozen programs including a disassembler, Steve Bjork's bouncing ball demos and many utilities. You will even find drivers and descriptors for the J&R Banker RAMdisk and the Speech/Sound Cartridge from Tandy. Plus, you'll find a number of files that show you how to patch several OS-9 programs. Take a look; you'll like what you see!

Tips From Bob Rosen

Bob Rosen reports that to use a monochrome monitor on the Color Video Composite output of the CoCo 3 you can type:

```
WIDTH 80: PALETTE 8, 255:
PALETTE 0, 0
```

To have your CoCo 3 tell you who wrote the Microware Disk patch, type:

```
WIDTH 40 : CLS 9 : CLS 100
```

And, finally, Rosen researched the pinout of the RGB monitor jack and

```
  here.
 */
mod_exec *driver;
/*
  The bootable diskette or hard disk
  partition's file number.
 */
FILE *disk;
/*
  The bootable diskette or hard disk
  partition's device name.
 */
char devname[32];
/*
  The bootable diskette or hard disk
  partition's driver name.
 */
char drivername[32];
int temp;
/*
  Reboot main program.
 */
main(ac,av)
int ac;
char *av[];
  {
  /*
    Reboot has no default argument.
   */
  if (ac != 2)
    failed("Wrong number of arguments");
  /*
    Argument 1 is the diskette or hard disk
    partition name.
   */
  strcpy(devname,av[1]);
  /*
    Try to link to the device.  Success if
    the device is already in the module
    directory.
   */
  device = modlink(&devname[1],DEVIC,OBJCT);
  /*
    Verify existence.
   */
  if (device == SYSERR)
    {
    /*
      No problem, try to load it.
     */
    device = modload(&devname[1],DEVIC,OBJCT);
    /*
      Verify existence.
     */
    if (device == SYSERR)
      /*
        Now we have a problem.  The device
        descriptor is not in the execution
        directory!
       */
      failed("Loading descriptor");
    }
  /*
    Ok, we will access the boot device
    directly.
   */
```

shares it here. Looking at the outside of the RGB out jack on the bottom of the computer, you'll see the following pinout:

```
 9  7  5  3  1
10  8  6  4  2
```

Here are the connections:

1. Ground	6. Polarity (No Pin)
2. Ground	7. Sound
3. Red	8. Hsync
4. Green	9. Vsync
5. Blue	10. No connection

OS9Gen Tips From Walt Weber

We received a nice note with several tips from Walt Weber in Marysville, Wash., recently that repeats some advice we have mentioned several times in the past but which bears repeating. If you own several versions of OS-9, make sure you use only the utility commands that came with each version. The same is especially true if you are using the ASM command that comes with OS-9. You need to make sure that you use the OS9DEFS files that came with the version of OS-9 you are currently using. If you don't, tricky bugs can sneak into your object code that will be almost impossible to find.

Weber has done a lot of experimenting with the OS9Gen command. In fact, he has written a program named *GenMod* which is available in the DL6 section of the OS-9 SIG on CompuServe. It adds three options to an existing OS9Gen command and fixes a few bugs he located in the original. *GenMod* makes it easier to change your OS9Boot file if all you need to do is delete and/or add a module to it.

While he was experimenting with OS9Gen he learned that the OS-9 kernel gets located at $F000 and is $F00 bytes long in Version 1.00. However, in Version 2.00.00, it is located at $EF00 and is $F80 bytes long. Unfortunately, it seems these locations and lengths are hard coded into Cobbler and OS9Gen. Version 1.00. of these commands marks the 15 sectors of Track 34 allocated in the disk allocation map of the diskette. Version 2.00.00 marks all 18 sectors, even though only 16 contain the kernel.

This means that if you boot up in Version 1.00, but use OS9Gen from Version 2.00, the wrong data will be written to the disk on Track 34. Booting up in Version 2.00.00 and using OS9Gen from Version 1.00 won't work either.

Weber also found a bug in Version 2.00.00 of OS9Gen. It seems that it will rewrite Track 34 if it determines that an OS-9 kernel is on the track. But, a bug in both Cobbler and OS9Gen can cause problems in the disk allocation map

```
strcat(devname,"@");
/*
  Need to find the device's driver now.
  Get the bas address of the descriptor.
 */
temp = device;
/*
  Copy the name of the driver from the
  descriptor.
 */
strhcpy(drivername,device->m_ddname+temp,32);
/*
  Hard disk descriptors that support booting

  will have their partition offsets
  copied.
 */
_strass(offset,device->m_control,3);
/*
  Now link to the driver.
 */
driver = modlink(drivername,DRIVR,OBJCT);
/*
  Verify.
 */
if (driver == SYSERR)
  {
  /*
    Driver not in memory, try to load
    it.
   */
  driver = modload(drivername,DRIVR,OBJCT);
  /*
    Verify.
   */
  if (driver == SYSERR)
    /*
      Driver not in execution directory either!
     */
    failed("Loading driver");
  }
/*
  Ok, now open the boot device.
 */
disk = open(devname,1);
if (disk == SYSERR)
  failed("Opening device");
/*
  Seek to sector Ø to satisfy the RBF
  manager's thirst for its contents.
 */
if (lseek(disk,ØL,Ø) == SYSERR)
  failed("Seek to Sector Ø");
/*
  Fake read.  RBF will get a copy of
  the identification sector here.
 */
if (read (disk,bootstrap,256) == SYSERR)
  failed("Reading Sector Ø");
/*
  Ok, now seek to the bootstrap on
  track 34.
 */
if (lseek(disk,TRACK34,Ø) == SYSERR)
  failed("Seek to Track 34");
/*
```

when a kernel is already on Track 34. Here is Weber's patch. The Cobbler patch begins at an offset of 23F from the beginning of the module.

```
OLD: EC 84 81 4F 10 26 00 A6 C1 53
     10 26 00 A0 A6 04
NEW: CC 4F 53 10 A3 84 10 26 00 A4
     A6 04 30 C8 52 12
```

Here is the OS9Gen patch. It begins at an offset of 47F.

```
OLD: EC 84 81 4F 10 26 01 06 C1 53
     10 26 01 00 A6 04
NEW: CC 4F 53 10 A3 84 10 26 01 04
     A6 04 30 C9 02 00
```

Additionally, you must change the A7 at an offset of 4A9 to an A6, and the 1D at an offset of 4AD to an 18.

Fixtime

Stephen B. Goldberg, of 695 Plainview Road, Bethpage, NY 11714, has contributed another interesting utility program. He wrote Fixtime because he got fed up with having the system clock end up 20 or 30 minutes slow after a long session at his CoCo. Fixtime is his attempt at keeping the clock on time. It does violate several rules of OS-9 programming. It is self-modifying and it loads and stores to fixed addresses. But it works. It must be loaded before you run it and Goldberg reports that he thinks it is a good idea to add it to your OS9Boot file after you have debugged it so that it won't get unlinked accidently. If the Fixtime does get unlinked after you run it, your system will crash. Also, after you have run Fixtime you cannot use Cobbler to generate a new OS9Boot file.

ReBoot

Bob Santy of Medford, Mass., and Greg Law both caught me when I stated that you couldn't remove the floppy disk driver and descriptor when running OS-9 from a hard disk. I stand corrected. While writing this column, I often need to transfer files from a disk someone mails me to my hard disk.

Because of this constant use, it never occurred to me that I could run the system without the floppy drivers — even though I could always load them in and use them when needed. The fact that the Tandy hard disk I used for several months seldom found the /h0 descriptor like it should until it had been running for several hours, also sidetracked me.

As Santy pointed out in his letter, the boot module stored with the kernel and init on Track 34 of a Color Computer OS-9 disk is completely self-contained and capable of reading the system into memory without the CCDisk driver and /d0 device descriptor. This means if you are using a hard disk you can save a lot of memory by not loading the floppy drivers. This can be a big help to you if you want to run the Tandy/Microware C compiler on the Color Computer.

Santy contributed the reboot utility program listed this month. It requires the pathname of the device you want to boot as an argument. He wrote it for use with the hard disk he purchased from Software Support of Ashland, Mass. That disk has bootable partitions and reboot works well on that device.

Even if Santy's utility will not work with your hard disk, it is full of excellent C code that may help you when you need to write other C utilities. It is an outstanding example of how good comments make a program easier to understand, and it is a valuable contribution. □

```
      Read the entire track into the buffer.
     */
    if (read (disk,bootstrap,BOOTSIZE) == SYSERR)
      failed("Reading Track 34");
    /*
      Verify that the track contains a bootstrap.
     */
    if (strncmp(bootstrap,"OS",2) == 0)
    {
    /*
      Ok, all is well.  The bootstrap is in memory
      and we are all set to execute it.
     */
#asm
 orcc #$50       Disable interrupts
 leax offset,y   Copy hard disk partition offset
 ldu #$3800      Hard disk boot module is
 ldb #3          Coded to find the partition
oloop
 lda ,x+         Offset at $3800 in RAM
 sta ,u+
 decb
 bne oloop
 lda >$3800      Check to see if
 cmpa #$FF       Booting floppy
 beq fboot       Floppy boot!
 lda #$22        Hard disk boot!
 sta >$FF7F      Select slot 3 of Multi-Pak
fboot
 clr >$78        Warm start indicator
 clra            Set DP to 0
 tfr a,dp
 leay bootstra,y Get base of boot
 jmp 2,y         Execute bootstrap
#endasm
    }
   failed("Disk not bootable!");
   }

/*
   Any failure uses this exit routine.
   Display specific error message and
   exit.
 */

failed(msg)
char *msg;
   {
   printf("ERROR: %s\n",msg);
   printf(usage);
   exit(errno);
   }
```

BITS AND BYTES OF BASIC

The First Days With CoCo 3: Experimentation and Discovery

By Richard A. White
Rainbow Contributing Editor

The old gray box was quiet as its plugs and multipack interface were pulled. Certainly, this had happened many times before in preparation for trips to users group meetings or the vacation cottage in Michigan. If it suspected that its successor had arrived, it did not let on.

And it had plenty of reason to doubt. In its nearly six years of existence, it had seen reams of words about other computers pour through its keyboard. The actual presence of a Model 100 had failed to change the old gray box's preeminent position. The Tandy 1000 was talked about, but never appeared on site, and dreaming about a "new CoCo" had been going on for over two years.

But, recently, a message appeared. You've seen them, the "while you were out" type, short and to the point:

Date: 10/10 Hour: 3:45
Name: Don Eaker
Telephoned: have good news

Don runs the Fairfield, Ohio, Tandy Computer Center and has a reputation for making things happen. His is the only Radio Shack facility in the southern Ohio area whose listing in the telephone book is in boldface type. Less than an hour later, CoCo 3, Serial Number 1001394, left its last Tandy home to go on active duty.

With the old gray box on the sidelines, it was simple enough to plug 1394 in its place and fire it up. It was feeding an amber monochrome monitor, since the analog RGB monitor had not yet showed up. Marty Goodman had already alerted us to the inadequacies of a monochrome monitor handling a color composite video signal, and the 32-character screen display was certainly less than good. No surprise here, so let's move on to the 40-character screen by typing `WIDTH 40`. The screen cleared to one color and what looked like the `OK` appeared in the upper left-hand corner.

The `O` was partly lost off the left edge of the screen. Adjusting the width knob on the back of the monitor brought the entire character onto the screen. There was still the black grid in the background that Marty had described a few issues ago.

Typing `WIDTH 80` produced the 80-character screen, which was also laced with background lines. Now came the discovery that makes computers so much fun. When I typed `CLS5`, the background cleared to a uniform shade and the characters were clearly readable.

With the legibility problem solved, it was time to move on to finding out what characters were available. Actually, all the characters and their codes are in the manual that comes with the machine. But there is a saying, "When all else fails read the manual." All else had not failed. A one-liner did the job and gave me a tool to probe speed in printing to the screen and observe screen scrolling:

```
10 FOR X=32 TO 255 :PRINT CHR$
(X); :NEXT :PRINT :GOTO10
```

This puts each printable character to the screen, then does a line feed and starts over again. The primary addition is a set of foreign and special characters that are not available from the keyboard. The old CoCo, colored block graphics are not available in the 40- and 80-character text screens.

The users group meeting the next day provided a chance for picky people to pass judgment on the characters and screen scrolling. A number were long-time *Wordpack* users. The consensus was that this was a good character set and that the scrolling was also good. And this came from one user who has complained long and loud about the jumpy scrolling of an IBM PC.

In the early days of the CoCo, much was made of the "high-speed pokes." `POKE 65495,0` doubles the microprocessor clock speed whenever it accesses the upper 32K where the BASIC ROMs are located. `POKE 65497,0` doubles the clock speed without qualification. Only the earliest models have trouble with the `POKE 65495,0`. The old gray box will tolerate it only if there is no disk ROM pack installed.

Since CoCo 3 supports the 1.8-MHz clock, only the `POKE 65497,0` is used. Certain functions, including musical tone generation, the cassette port and serial port are clock-rate dependent and the `POKE 65496,0` is used to return the machine to the slower clock to achieve proper timing. Some people have been able to use the serial port at the higher clock rate. To date I have not heard of successful cassette operations at a high clock rate. You can easily see the effect of the high speed clock by running the screen scrolling program previously mentioned to get a feel for screen writing and scrolling at low speed. Then `POKE 65497,0` and run the program again. It really speeds things up. I suspect that most CoCo 3 software will be designed to use the high clock rate.

The next thing to do was to turn everything off and plug in the multipack interface that held the disk drive controller. I booted back up and issued a `DIR` command. I got back an SN Error. BASIC had not recognized the drive controller. Eventually, I dispensed with the interface and then could boot into Disk Extended BASIC Version 2.0 or 2.1. I have controllers with both versions 1.0 and 1.1 available. The CoCo 3 converts these to versions 2.0 and 2.1. However, when I tried to access the disk, an I/O Error resulted. Finally, it dawned on me that these were older controllers that require 12 volts, which is not supplied by either CoCo 2 or CoCo 3. Even worse, these controllers almost worked. They work well enough to trash any disk you try them on. So beware!

The solution to this problem is to use the older controllers in a multipack interface. But, if CoCo 3 does not see the Disk BASIC when it is in a multipack interface, we are back to square one. The problem is in the interface, and

Richard White lives in Fairfield, Ohio, has a long background with microcomputers and specializes in BASIC programming. With Don Dollberg, he is the coauthor of the TIMS *database management program.*

Radio Shack has said the PAL chip would need to be replaced. (See Marty Goodman's article on Page 98 of this issue.) However, we had assumed that interfaces would work with 128K machines and that the upgrade was needed to operate with 512K. We now know that the upgrade is needed for an interface to be operable at all. A disk controller with a Radio Shack DOS ROM that works in a CoCo 2 works in the CoCo 3.

One result of these doings is that the old gray box is back in its accustomed place performing its usual duties, while 1394 is on another table making pretty pictures. This arrangement works better for doing this column, since I can experiment on the CoCo 3 and then immediately move over to write about what I found.

At this point, I wanted to see some graphics. Rick Adams and Dale Lear had two programs on October's RAINBOW ON TAPE. The tape interface worked. The *Tunnel* program is not much on a black-and-white monitor, but *Wheel* does what it is intended to do. Since I had seen what I wanted to see on the monitor, it was time to hook 1394 to a color TV.

It quickly became apparent why Radio Shack has stayed with the 32-character CoCo screen so long. The 40- and 80-character text screens and the new high resolution graphics screens are 64 pixels wider than the old CoCo screens. If the TV does not properly center the screens, the edge of the left side is lost. In my case, one to two characters on the left edge of the 40-character screen and more on the 80-character screen were off the screen. The TV at the Computer Center was only somewhat better.

Interestingly, 80-character text is readable on my color TV, but it is fairly new and still has good contrast. And the quality is not what one needs for continuous use. Expect 80-column text to be illegible on old sets with poor contrast and other ills. I quickly found myself returning to the 32-character screen to enter programs. The 32-character screen looks fine on a color TV. It is only on monochrome monitors that one has trouble. This is not to say that programmers should not use the 40-character screen. It does mean that programs must forgo using a couple of columns at least on the left side of the screen.

The cause of the the centering problem lies in the computer, since the screen is offset to the left on both the TV and monitor. I am told that the same thing happens when an MS-DOS machine runs on a TV. I expect that much new programming will use CoCo 3 modes, partly because of the ability to put text onto graphics screens.

Tunnel in the 16-color graphics mode on the color TV was something else entirely. The pastel colors blending into each other gives an effect totally lost in black and white.

Looking for something more led me to Sample Program 24 in the manual. It draws 80 circles on the screen and fills each with a random color. It then draws some colored bars at the top and bottom of the screen. Then the program goes into a loop and randomly changes the colors in the palette. The result is spectacular with all elements changing colors at the same time. And it's fast. I put a high speed poke into the program, but took it out because the changes came too fast.

On a reasonable quality TV, CoCo 3 is a spectacular graphics machine. On top of this it appears that the new high resolution graphics commands are a bit easier to understand and use than the original Extended BASIC commands.

While the original Extended BASIC commands remain and should run all older Extended BASIC programs, their capability is enhanced so they can use any of the computer's 64-color set. These colors are now available to all text and graphics modes. The job is done through use of the palette, which is as good a place as any to start.

In the CoCo 3, as in previous CoCos, each color is referred to by number. Previous CoCos supported eight colors plus black in the low resolution graphics and two or four color subsets of these colors in the higher resolution modes. These restraints still apply, except that the available colors are determined by reference to specific slots in the palette that are initialized with numbers that refer to colors available on previous CoCos. These default settings can be changed from the keyboard or from a BASIC program changing the available colors in any text or graphics mode at any time.

PALETTE is used in two ways. The first is to configure the colors in the slots for either a composite monitor or TV set: PALETTE CMP. This is the default from a cold start. Whenever PALETTE CMP is issued, the default palette colors are reestablished. PALETTE RGB sets up the machine to use the RGB analog monitor to be released by Tandy. The RGB monitor interpretation of colors is different from that of a composite video monitor. When you cold start a CoCo 3 with an RGB monitor, you will have to enter the PALETTE RGB command from the keyboard. The command could also be the first line of a BASIC program. I suspect that there will be a way to configure the computer for an RGB monitor from an OS-9 start-up file.

The second use is to change the color assigned to a particular slot or palette register. The syntax is PALETTE pr, cc where pr stands for palette register or slot and cc means color code.

There are 16 palette registers numbered 0 to 15. The computer refers to the palette registers to determine which colors are to appear on the screen. When a palette register's color assignments change, all locations on the screen referring to that slot change. Here is a BASIC program that will change the screen color showing all available colors:

```
10 HSCREEN 2
20 HCLS 2
30 FOR X=0 TO 63 : PALETTE 2,X
: FOR Y=1 TO 300 :NEXT Y,X
```

HSCREEN 2 sets the computer for the 320-by-192, 16-color graphics. HCLS2 clears the graphics screen to the color in Palette Register 2. In Line 30, X is incremented from 0 to 63 and the Color Codes are loaded into Palette Register 2 with PALETTE 2,X. FOR Y=1 TO 300 :NEXT Y is simply a delay loop so the colors can be observed.

The manual does not list all the available colors. This is probably because of differences in the ways different monitors and TVs display colors. What may be yellow on one machine could be yellow-orange on another, while a pastel green might be shifted to yellow. What they call buff looks like white to me. A program, Sample Program 23, is provided to present each color and its number in groups of eight. With this program, you can view all the colors and list them as they appear with your equipment. A picture of the first screen shown by this program should help you adjust your TV or monitor to display colors in a similar manner. There appear to be 16 basic colors with pastel variations. In the first cycle, the colors are intense with intensity diminishing in higher numbered cycles. The only colors I really missed were dark brown and a selection of grays. A picture of the old gray box might need to have a bluish cast. And it certainly will be possible to draw an electric pink Cadillac.

The startup color assignments are listed in the manual and are reproduced in Table 1.

The colors in slots 0 through 8 include the colors used by the traditional CoCo low resolution text screen, but color number does not correspond to slot

Slot	Color	CMP Code	RGB Code
0	Green	18	18
1	Yellow	36	54
2	Blue	11	9
3	Red	7	36
4	Buff	63	63
5	Cyan	31	27
6	Magenta	9	45
7	Orange	38	38
8	Black	0	0
9	Green	18	18
10	Black	0	0
11	Buff	63	63
12	Black	0	0
13	Green	18	18
14	Black	0	0
15	Orange	38	38

Table 1: Startup Color Assignments

	CLS	ATTR Foreground	ATTR Background
Color 0	----	Slot 8	Slot 0
Color 1	Slot 0	Slot 9	Slot 1
Color 2	Slot 1	Slot 10	Slot 2
Color 3	Slot 2	Slot 11	Slot 3
Color 4	Slot 3	Slot 12	Slot 4
Color 5	Slot 4	Slot 13	Slot 5
Color 6	Slot 5	Slot 14	Slot 6
Color 7	Slot 6	Slot 15	Slot 7
Color 8	Slot 7	----	----
Defaults —	Foreground:	Slot 8	Black
	Background:	Slot 0	Green

Table 2: Color Slot Numbers

number. Black is in slot 8 and all colors are in slots numbered one slot lower than the color number. I am not satisfied that this confusion is needed.

Things get worse with the 40- and 80-column text screens. CLS works just like it does for the 32-column screen. A new command, ATTR c1,c2,B,U is available. How ATTR does what it does is not immediately obvious. Central to understanding is that c1, the foreground color, uses one set of palette slots while c2, the background color, uses an entirely different set of slots. Further, c1 uses higher numbered slots, while c2 uses lower numbered slots. It seems backward. Reread this paragraph and make sure you understand the arrangement. Perhaps a table will help. Table 2 appears on Page 300 in the manual. The discussion of ATTR in the front of the manual does not even mention the table, leaving the user to wallow in confusion.

To finish off ATTR, B is a switch to start text blinking and then stop blinking; U is a switch to start and stop underlining.

At this point we have enough information to explain why CLS5 clears the screen on a monochrome monitor and makes the lettering legible. Color 5 references Slot 4, whose default color is buff (which is white for most purposes). The default foreground color is black. The composite video signal carries only black and white information, which is what the black and white monitor wants to see. ATTR 2,4 accomplishes the same thing. Available background colors do not include black in the default situation. Try this:

```
PALETTE 0,0
ATTR 1,0
```

This assigns the color black to Slot 0; green is in Slot 9, which is used as Foreground Color 1. The result is green letters on a black background. ATTR 3,0 provides buff letters on black. Both displays have color fringes in a TV screen, but don't look bad from a distance. Neither color is readable on the 80-character screen, and dark on light is better on the 40-character screen. PALETTE CMP undoes all the damage we did with this experiment.

PRINT, PRINT TAB and PRINT USING work on the 40- and 80-column screen just as they do on the 32-column screen. PRINT @n, is only used on the 32-column screen. On the 40- and 80-column screens use LOCATE x,y to position the cursor to column x and row y. The next PRINT statement will begin printing at that location. The column can be 0 to 39 on the 40-column screen and 0 to 79 on the 80-column screen. On either screen, rows range from 0 to 23.

Note the difference between the ways PRINT@n, and LOCATE x,y work. LOCATE x,y merely positions the cursor at a particular column position on a specified line. When using PRINT @n, each screen location has a number. The range is 0 to 511 on the 32-character screen. Further, PRINT @n, expects there to be a string or variable following the comma, which is to be printed. You can use PRINT @n, like LOCATE x,y, if you follow the comma with a null string and a semicolon, like PRINT @n,"";.

As noted, the left one or two character positions on the 40-column screen are lost when using CoCo 3 on a color TV.

Good programming practice will be to start all printing at Column 3. This will generalize your programs so they will work on monitors or TVs. LOCATE 2,y would be used prior to each PRINT statement that is to print to the left side of the screen.

Alternately, you can use PRINT TAB(4);"TEXT". This prints the text at the same position as LOCATE 2,y. The implication is that the computer uses 80-column locations even when working on a 40-column screen.

This essentially covers the operation of the 40- and 80-column text screens. They represent a major improvement over previous CoCos. Next month we will move on to the new graphics capabilities.

What you need to know to upgrade your Multi-Pak Interface for use with your new CoCo 3

A PAL for Your CoCo 3

By Marty Goodman

Reference has been made to the fact that you will need to upgrade your Multi-Pak Interface if you want to use it with your new CoCo 3. There has been a lot of confusion about this. In this article, I hope to clarify matters regarding this upgrade, explaining why it is necessary, how to do it, and where and how to get the needed parts. Much of my information for this article comes from *Tandy Technical Bulletin CC:29*, dated August 14, 1986. This bulletin is distributed to Tandy Computer Service Centers. Some of it comes from personal tests, and some from personal communication with Mark Siegel of Tandy Corporation.

There are two different types of Multi-Pak Interfaces (MPI) for the CoCo. The "old MPI" was the first MPI sold by Tandy. It is physically larger than the newer ones, and is Radio Shack Catalog No. 26-3024. This model was first sold with a battleship gray case, though later it was sold with a white case.

Later on, Tandy redesigned the MPI, making it smaller and much less expensive to produce. This newer, slimmer, trimer MPI is Radio Shack Catalog No. 26-3124. The catalog number can be found on a sticker on the bottom of your Multi-Pak.

When folks first got their CoCo 3s, brought them home, and plugged them into their old Multi-Paks (with the disk controller plugged into Slot 4 as recommended by Tandy), they discovered one of three things. Some found that their MPIs appeared to work perfectly. Others found that their disk controller would be recognized by the CoCo 3 only if it was plugged into Slot 1. Still others found that regardless of what slot they plugged their disk controller into, it simply would not be seen by the CoCo 3. This different behavior is explained by the fact that different releases of the same catalog number MPI had slightly differently programmed PAL chips in them.

The critically important thing to note here is that *regardless* of how your MPI behaves with your CoCo 3, it is *necessary* to upgrade it either by replacing the PAL chip (in the case of the older MPIs) or by adding in a satellite board (in the case of the newer ones). Both the technical bulletin and Mark Siegel, personally, were quite firm in this recommendation.

Note that earlier, several folks alleged that this upgrade was needed only if you were going to install 512K in your CoCo 3. That is false. You need to do this upgrade regardless of the amount of memory in your CoCo 3.

At the Princeton RAINBOWfest, I heard that all local Radio Shack retail outlets had been briefed in the problem with the Multi-Pak. However, I called five Radio Shack Computer Centers in the San Francisco Bay area, and five Radio Shack stores with computer divisions. No one had heard of this problem. Because of this, I urge you, the RAINBOW reader, to tell *your* local Computer Center or division personnel about this problem.

By informing the local stores, you may save other CoCo 3 purchasers some grief. You might even consider urging your local Radio Shack store employee to buy a copy of this issue of RAINBOW for reference on this matter.

If your Multi-Pak is still under warranty (if it has been less than 90 days since it was bought or last repaired) I have been told by Mike Ward that you can get it upgraded for free. Please check out this possibility if your MPI falls into that category. I do not know how much, if anything, Tandy ordinarily charges for labor on upgrading the Multi-Pak. The parts fee is $7.50. You may want to inquire about this before deciding to do the upgrade yourself. Usually Tandy's minimum labor charge is $15. Even if Tandy is providing the labor for the upgrade for free, you still may want to upgrade the Multi-Pak yourself, because that way it will not be out of your hands for the one to two weeks that such repairs often take.

Upgrading Your Old Multi-Pak

If you own one of the older MPIs (Catalog No. 26-3024) it will be relatively easy for you to do the upgrade yourself. All you need is a 20-pin PAL chip. To order it, just go to your local Radio Shack store and tell them you want to order, from National Parts, Part Number AXX 7123, and also tell them that that part is for Catalog No. 26-3024. It should cost $7.50. It usually takes about two weeks for the part to arrive at your local Radio Shack store.

Be sure the MPI is unplugged, then open the case of your old Multi-Pak. This, of course, will involve breaking the warranty sticker that covers one of the four screws.

Remove the black cardboard shield that covers the space around the four card sockets. You will now see the circuit board. Note that only one chip on that board is socketed. All others are soldered directly to the board. The chip you are interested in is the socketed one, a 20-pin chip called U8. Carefully remove this chip and replace it with the new chip you just bought. In doing this you must be careful not to bend pins on the new chip, and you must be careful to put in the new chip with its notch pointing the same way as the notch on the chip you just removed. Also, observe the usual anti-static precautions for handling delicate chips. The new chip will come to you on a piece of black anti-static foam. Place the old chip that you removed into this foam, and save it in a safe place. I will explain why later.

After successfully replacing the old PAL chip, put your MPI back together again and the upgrade is complete. Be

careful about positioning the plastic extension of the slot select switch as you are reassembling the Multi-Pak.

Upgrading Your New Multi-Pak

If you own one of the newer MPIs (Catalog No. 26-3124), I recommend you take your MPI to Tandy and have them upgrade it for you. This is because the upgrade involves cutting a trace on the circuit board, and then delicately soldering seven wires of that satellite board to various integrated circuit chip pins on the board. If you feel totally comfortable with doing such work, what follows is a brief outline of the upgrade process.

Go to your local Radio Shack and order from National Parts a Satellite Board, Part No. AXX 7119 (for Catalog No. 26-3124). This should also cost $7.50.

When you get the satellite board, open up your MPI, being careful to note the length of the screws that hold it together and what holes they go in. Be sure, that your MPI is unplugged. Now:

1) Unscrew and free the printed circuit board. This involves removing three screws that hold it down, removing all metal clips that hold the shield to it, and removing the shield itself. Carefully note the position of the metal clips for reassembly. You need not disconnect the transformer.

2) Locate IC6, the monster 64-pin chip. Cut the trace that connects Pin 52 of IC6 to Pin 19 of IC1 (the 74LS245 chip used to buffer the data lines).

3) Position the satellite board over IC 6, with its components up, and with its wires facing the card edge.

4) Locate IC4. This is one of three 74LS367 buffer chips. Solder the three yellow wires from the satellite board to pins 3, 9, and 11 of IC4. It does not matter which yellow wire goes to which of the three pins on IC4.

5) Solder the white wire from the satellite board to Pin 52 of IC6 (the big IC).

6) Solder the blue wire from the satellite board to Pin 19 of IC1 (the 74LS245).

7) The remaining red and black wires from the satellite board need to be hooked up to +5 volts and ground, respectively. This can be gotten from the power supply pins of any of the chips on the board. Tandy recommends using pins 16 and 8 of IC5, respectively. (Red wire to Pin 16 of IC5 and black wire to Pin 8 of IC5.)

8) Replace the shield on the circuit board. Using two of the three screws that held it to the case, reattach it. The screws to use are the one near the power switch and the one near the selector switch. Now close up the case, being careful to make the extension of the selector switch fit properly.

Problems You Must Know About

After you have completed either of these upgrades (or had Tandy do them for you), the Multi-Pak will now work fine with your CoCo 3 with nearly all Tandy hardware and software. *But*, the fix causes some serious problems with some third-party products. After upgrading your Multi-Pak, it will *not* work with the *CoCo Max* Hi-Res joystick interface. Even if you use that upgraded Multi-Pak with an old CoCo 1 or 2 (the *CoCo Max* software itself will not run on the CoCo 3), the *CoCo Max* hardware card will cease to work. The PBJ Word-Pak I and II will also cease to work with Multi-Paks that have been upgraded for the CoCo 3. The Microworks DS-69A digitizer will also not work with upgraded Multi-Paks.

The reason for this is that the upgrade locks out address space $FF80 through $FF9F. The upgraded Multi-Pak can no longer access those addresses. This lockout is done because the GIME chip in the CoCo 3 uses those addresses, and the upgrade makes sure that no other device will affect those locations. The third-party devices I just mentioned all use port addresses in the $FF90 to $FF9F range, and thus are locked out by the Multi-Pak upgrade.

Note that the PBJ Word-Pak I and II are no longer in production, and the current model (PBJ's Word-Pak RS) uses ports at $FF76, 7, 8, and 9. Thus, it is not affected by the Multi-Pak upgrade. I have spoken with Bob Lentz of Microworks, and he is in the process of fixing his DS-69A digitizer to work on the CoCo 3. At present, I have no word on whether or not a fix for *CoCo Max* will be available.

A Possible Solution for Hackers

If you own one Multi-Pak and occasionally want to use it with a CoCo 2 with some of the hardware that is now locked out, there are a few approaches to fixing this problem. In all cases you need to be a little bit of a tinkerer.

If you have a new, small MPI (Catalog No. 26-3124) the fix is quite easy. Open the Multi-Pak and remove the circuit board and shield. Remove the white and blue wires of the satellite board from where they are soldered to the ICs on the main board. Now, send wires from Pin 52 of IC6 and Pin 19 of IC1 to the two poles of a DPDT switch. On one of the two positions, short the two connections together. On the other, hook up the blue and white wire from the satellite board. Refer to my description of the upgrade of the new Multi-Pak to clarify just what wires go where. In effect, what you are doing is switching the satellite board in and out of the circuit. Mount the DPDT switch on the case of the Multi-Pak. You now have a Multi-Pak that can be switched between CoCo 2 and CoCo 3 modes. In CoCo 2 mode, it will work with a CoCo 2 and any of the add-on, third-party hardware I mentioned above. In CoCo 3 mode, it will work properly with a CoCo 3.

If you own one of the older Multi-Paks (Catalog No. 26-3024) you might consider rigging up a "PAL switcher." This switcher is easier to build than you might think, because most of the pins on the PAL chip are either power or inputs. Indeed, only pins 14, 15, 16, and 18 are outputs. Armed with this knowledge, you can piggyback or wire in parallel all but those four pins, then switch them using a 4PDT switch.

Ghost Busting

The reason for the Multi-Pak fix for the CoCo 3 is at least twofold.

First, the older PAL chips used to decode the software slot select port for the Multi-Pak "ghosted" from $FF7F to $FF9F. That is, when a value was written to $FF7F, it appeared at $FF9F also, and vice versa. This ghosting caused no problems with the CoCo 2, but it plays havoc with the CoCo 3, which occasionally wants to write to $FF9F when talking to its GIME chip. Similarly, any attempt at slot selection with a ghosting Multi-Pak will send spurious data to the GIME chip. Thus, the decoding of the software slot selection port had to be made more complete.

Second, Tandy felt it necessary to lock out the $FF80 through $FF9F range ($FFA0 and up are already locked out by the programming of the old Multi-Pak PAL chips). This was to protect the GIME chip from conflicts with information from other external devices that might be addressed in that range.

I have been asked many times already why one should upgrade one's existing Multi-Pak if it happens to be one that appears to work fine with the CoCo 3. I can't fully answer that question at this time. I must emphasize, however, that both the Tandy technical bulletin and top Tandy officials strongly insist that regardless of whether or not your Multi-Pak *appears* to work properly with your CoCo 3, you must get it upgraded to assure proper operation.

The CoCo ROS, Part II: Building the ROS Circuit

By Dennis H. Weide

Last month, I introduced you to the CoCo ROS and gave you a brief explanation of what it can do. This month, I'd like to show you the actual ROS circuit and give some tips on how to build and test it. I'll explain how the circuit works and how to program it.

The ROS Design

The idea for the ROS circuit came from the book *TRS-80 Models I,III, & Color Computer Interfacing Projects*, by William Barden, Jr. It's an excellent book on interfacing projects and shows various methods for interfacing the CoCo. Borrowing from his idea, I've increased the capacity of the original circuit by adding a second 8255 Processor Peripheral Interface chip and providing the proper address, read/write and chip select (CS) decoding for it.

Figure 1 is the programmable peripheral interface circuit used by the ROS. We'll refer to it as the Robot Operating System Signal Processor circuit or simply the ROSSP. It uses two 8255 PPI chips whose data buses are connected in parallel. The chip select (CS)

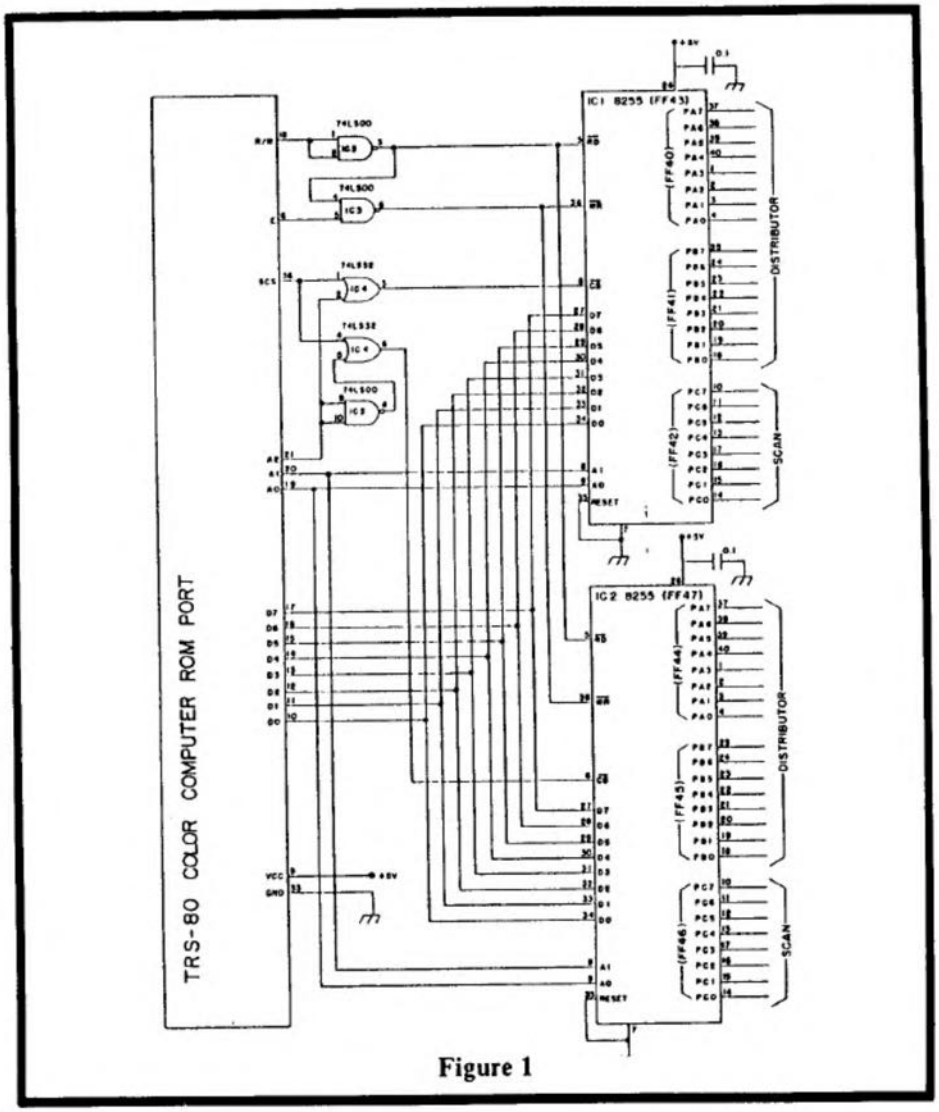


Figure 1

Dennis Weide is a communications technician for AT&T communications in Albuquerque, New Mexico, where he programs AT&T and IBM PCs. He enjoys making toys and teaching computer programming.

Figure 2: PPI Control Word Layout

Bit	Function		
0	Port C (lower)	1=Input	0=Output
1	Port B	1=Input	0=Output
2	Mode select	0=Mode 0	1=Mode 1
3	Port C (upper)	1=Input	0=Output
4	Port A	1=Input	0=Output
5,6	Mode select	00=Mode 0 01=Mode 1 10=Mode 2	
7	Mode set flag	1=Active	

Examples:

Dec Value	Binary Value	Input Ports	Output Ports
128	10000000	None	A,B,C
129	10000001	C(lower)	A,B,C(upper)
130	10000010	B	A,C
131	10000011	B,C(lower)	A
136	10001000	C(upper)	A,B,C(lower)
144	10010000	A	B,C
146	10010010	A,B	C
154	10011011	A,B,C,	None

Decimal value to be poked into &HFF43,&HFF47

lead uses the A2 Address Lead and three gates to select one of the PPI chips. Address leads A0 and A1 allow you to select one of three addresses per chip to read or write. For the ROSSP, we've chosen two addresses on each chip for outputs (distributors) and one for inputs (scans). Later, I'll show you how to program the PPI chip for all inputs or all outputs.

A Look at the PPI

The 8255 PPI is a 40-pin DIP IC that has three programmable modes. The mode I've chosen (Mode 0) allows for two output ports (ports A and B) and one input port (Port C). The outputs are latched and must be set and reset as desired. The inputs follow the state of the peripheral. To program the PPI, poke the control word for the proper mode according to Figure 2. The addresses for each PPI chip are shown in Figure 1 in parentheses. The control word is address $FF43 for IC 1 and $FF47 for IC 2. To program each PPI for two outputs and one input port, POKE &HFF43,&H89 and POKE &HFF47,&H89. To read the input ports, print PEEK(&HFF42) and print PEEK(&HFF46). To write to the output ports, POKE &H*XXXX*,*YYY* where *XXXX* is the hexadecimal address to load and *YYY* is the value (0-255) to write. For example, to write to address $FF40 Port PA3 only, POKE &HFF40,8; 8 is the binary value of Bit 3. You can use the AND, OR and NOT functions to control multiple actions from the same address. Figure 3 shows how I wired the inputs and outputs for use in the ROSSP. Each 8255 PPI chip requires one circuit of Figure 3. Only one bit of each chip is shown wired although all bits must be. For IC 2, the pin designations change from 1-8 to 10-16. All other designations stay the same. Besides the ROSSP, this circuit has many other useful applications.

A Look at the Octal Buffers

For buffering between the 8255s and the peripheral equipment, I've used 74LS240s and 74LS244s. These 20-pin DIP octal buffers/line drivers are capable of handling 40 milliamps of current; enough to drive high-resistance relays or LEDs. The 74LS240 buffer inverts the input signal; the 74LS244 doesn't. Using these two buffers, you can design many different circuit configurations. For the ROSSP, the enable leads (pins 1 and 19) are grounded to enable all outputs at all times. In Figure

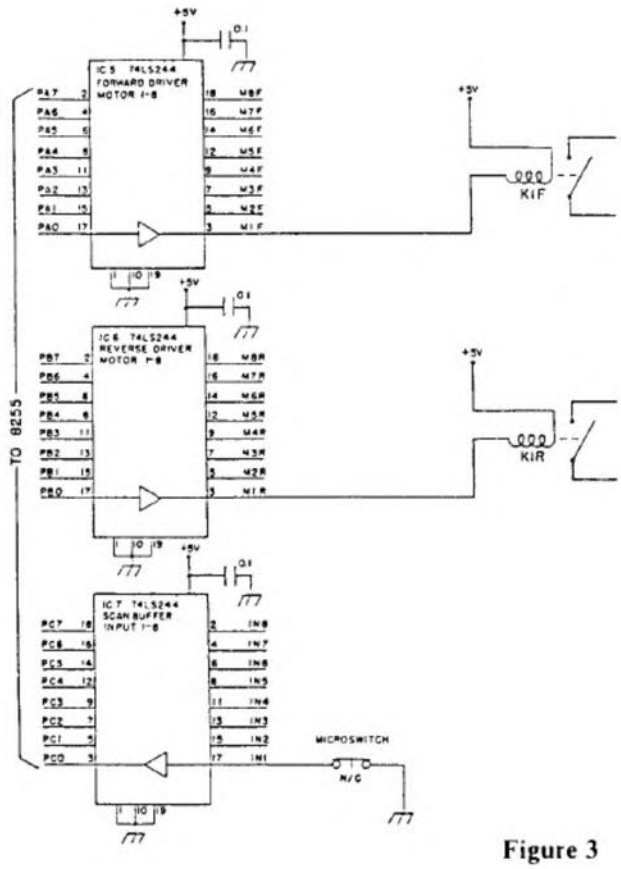


Figure 3

3, the 74LS240's drive relays direct. A high on any input to the buffer causes a low output, which in turn operates the relay. The ROS program is designed so that only one relay can operate at a time, thus allowing the entire ROSSP to be powered from the computer. Use relays with a high resistance winding so that current requirements are kept as low as possible. If you build a separate +5-volt power supply, be sure to keep a common ground between the computer and the ROSSP.

To operate any particular relay, write the bit value of that relay to the address of that port. When any port of the 8255 is poked to zero, all relays are released. When it's poked with a value between 1 and 255, the corresponding relays will operate.

Building the ROSSP

Building the ROSSP circuit is easy. We won't go through a step-by-step description of how to build it because construction isn't critical. I'll give you whatever hints are necessary to help you along. If you're inexperienced in circuit construction, I recommend you study the article and schematics carefully before beginning. If necessary, find someone in your local CoCo club to guide you along. After you've built this project, you'll be ready for almost anything.

If possible, use wire-wrap connections and 30-gauge wire. You can use point-to-point soldering, but wire wrapping is faster and easier. Keep the leads from the ROM port to the PPIs as short as possible. Use .1mfd capacitors across the power leads of all ICs to prevent high-speed switching errors. All other wiring is non-critical.

I used a double-sided Vector 3795 board with a pin spacing of .1 inch. I had to cut the pins on each side of the board to make it a 40-pin board. Any double-sided circuit board with a pin spacing of .1 inches and at least 40 pins can be modified to work. If you use the Vector board, you'll have to add on a piece of perfboard to fit the entire circuit. Use IC sockets and IDs for the bottom of the sockets to assist in wiring. Check all connections carefully and confirm your wiring with an ohmmeter before installing the ICs or plugging the board into the computer. When the wiring is correct, you're ready to plug it in and test it.

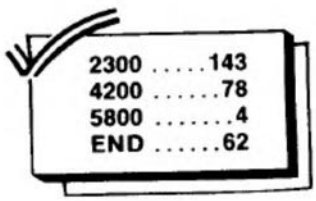
Testing the ROSSP

Using the program listing and last month's logic tester, you can test the circuit. First, turn off the computer. Then plug the ROSSP into the ROM port. Be sure that the circuit board fits properly into the port. A shorted pin could destroy the 6809E microprocessor chip. Connect the logic tester to +5 volt (red lead) and ground (black lead) of the ROSSP then turn the computer power on. Load and run the BASIC program.

Follow the instructions on the screen to test the two 8255 PPI chips and the four 74LS240 buffer chips. Test the 74LS244 buffer chips using the PEEK command and grounding the input leads one at a time. A grounded input lead will cause the corresponding bit to be reset when the PEEK command is executed. If no input leads are grounded, the PEEK command will return a value of 255.

Next month, we'll take a look at the ROS program and discuss how to connect the circuit to the Robotix R-2000 kit. We'll also take a look at a sample ROS program. Until then, have fun with this one.

□

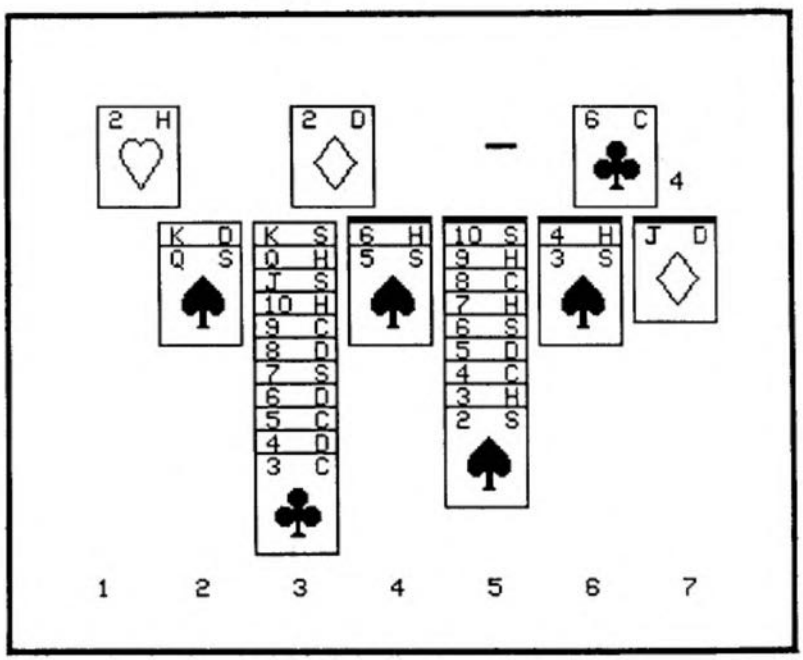


The listing: ROS TEST

```
100 '      PROGRAM LISTING 1
200 '      ROS TEST PROGRAM
300 '      BY DENNIS H. WEIDE
400 '      (C) 1986
500 '      FOR TESTING THE ROSPPI
 CIRCUIT
600 '
700 '
800 CLS:POKE &HFF43,137:POKE &HF
F47,137
900 POKE &HFF40,0:POKE &HFF41,0
1000 POKE &HFF44,0:POKE &HFF45,0
1100 FOR X=0 TO 7
1200 READ Y:PA(X)=Y:NEXT X
1300 FOR X=0 TO 7
1400 READ Y:PB(X)=Y:NEXT X
1500 FOR X=0 TO 7
1600 READ Y:PC(X)=Y:NEXT X
1700 PRINT
1800 '      SET PIN NUMBERS
1900 DATA 4,3,2,1,40,39,38,37,18
,19,20,21,22,23,24,25,3,5,7,9,12
,14,16,18
2000 CLS:PRINT:PRINT
2100 PRINTTAB(6)"ROSPPI TEST PRO
GRAM"
2200 PRINT:PRINTTAB(4)"1. TEST 8
255 PPI'S "
2300 PRINT:PRINTTAB(4)"2. TEST 7
4LS240 BUFFERS"
2400 PRINT:PRINTTAB(4)"3. END CI
RCUIT TEST"
2500 PRINT:PRINTTAB(4)"ENTER ONE
 OF THE ABOVE";
2600 INPUT Q
2700 ON Q GOTO 3000,3500,2900
2800 GOTO 2000
2900 CLS:END
3000 AD=&HFF40:IC=1:P=1:GOSUB 43
00
3100 AD=&HFF41:IC=1:P=2:GOSUB 43
00
3200 AD=&HFF44:IC=2:P=1:GOSUB 43
00
3300 AD=&HFF45:IC=2:P=2:GOSUB 43
00
3400 GOTO 2000
3500 IC=5:AD=&HFF40:GOSUB 6100
3600 IC=6:AD=&HFF41:GOSUB 6100
3700 IC=8:AD=&HFF44:GOSUB 6100
3800 IC=9:AD=&HFF45:GOSUB 6100
3900 GOTO 2000
4000 '
4100 '      8255 TEST SUBROUTINE
4200 '
4300 CLS:PRINT"SET LOGIC TESTER
TO RED"
4400 FOR X=0 TO 7
4500 IF P=1 THEN 4600 ELSE IF P=
2 THEN 4700
4600 PRINT@64,"ADDR="HEX$(AD)"
 IC="IC"    PIN="PA(X):GOTO 4800
4700 PRINT@64,"ADDR="HEX$(AD)"
 IC="IC"    PIN="PB(X)
4800 PRINT@128,"OUTPUT IS LOW (R
ED)"
4900 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
5000 POKE AD,2^X
5100 PRINT@128,"OUTPUT IS HIGH (
GREEN)"
5200 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
5300 POKE AD,0
5400 PRINT@128,"OUTPUT IS LOW (R
ED)"
5500 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
5600 NEXT X
5700 RETURN
5800 '
5900 '      74LS240 TEST SUBROUTI
NE
6000 '
6100 CLS:PRINT"SET LOGIC TESTER
TO GREEN"
6200 FOR X=0 TO 7
6300 PRINT@64,"ADDR="HEX$(AD)"
 IC="IC"    PIN="PC(X)
6400 PRINT@128,"OUTPUT IS HIGH (
GREEN)"
6500 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
6600 POKE AD,2^X
6700 PRINT@128,"OUTPUT IS LOW (R
ED)"
6800 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
6900 POKE AD,0
7000 PRINT@128,"OUTPUT IS HIGH (
GREEN)"
7100 PRINT:INPUT"PRESS <ENTER> T
O CONTINUE";Q
7200 NEXT X
7300 RETURN
```

The Solitary Endeavor

By Tudor P. Jones



Solitaire is a computer version of the old standby favorite card game whose object is to put shuffled playing cards back into order.

Twenty-eight cards are dealt to the columns to start the game, and others are turned from the remainder, or display, and added to the columns or stacks as they appear. If they cannot be legally entered onto the columns or stacks, then they have to be left in the display until they are exposed once again.

The bottom card on any column may be moved either to its appropriate stack to continue the sequence, or to the bottom of another column. The visible display card can also be moved to its stack, or to the bottom of any column, but only in accordance with the usual rules. The columns must be in descending sequence and alternating color (diamond or heart on a club or spade).

One card is dealt face up by the computer to begin Column 1, and six more face down in a row, to begin columns 2 to 7. Another card is dealt face up on Column 2, and five more face down on columns 3 to 7. This is continued until there are seven piles of cards increasing in number from one to seven from left to right, with only the top card of each pile exposed.

As each ace becomes available, it should be moved to begin the stacks above the columns. Each stack is built up in suit and sequence to the king.

The 24 remaining cards are turned over three at a time. The third card is visible in the display. The number alongside the display indicates which card is visible.

After making any opening moves that are possible, press the space bar to turn over the next three cards. When all of the cards have been dealt this way to the display, another press of the space bar turns all of them over, and deals the top three cards of those remaining, with the third card being visible in the display, and available for play.

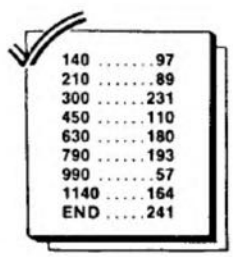
Press the down arrow and a column number to move a card from the display to that column. Any illegal move generates a NO message. The left arrow moves a card from the display to the stacks. Press the up arrow and column number to move a card from the bottom of a column up to the stacks. The right arrow plus two column numbers (from and to) moves a card or cards from column to column. A whole column, or part of a column, may be moved to another following the descending/alternating rule.

When a column is cleared to reveal the top of a face-down pile, the top card of this pile is turned over by the computer. The heavy line over a column indicates that unexposed cards are lying beneath the top card.

When a pile has been cleared to leave a space, it can only be filled with a king or a column of cards built on a king.

As in regular solitaire, the game can not always be completed. Sometimes, the cards are shuffled so it is impossible for the four stacks to be finished. Failure, therefore, is not always a sign of bad play. On the other hand, some games may be lost through bad play that would otherwise have come out.

(Questions about this program may be directed to Mr. Jones at 2338 Ryder Street, Ottawa, Ontario, Canada K1H 6X6, 613-731-3365. Please enclose an SASE when writing.) □



The listing: SOLTAIRE

```
10 CLS:CLEAR:DIMDECK(52),COL(7,2
0),C$(4),S$(4),N$(14),YC(7),STAC
K(4):R=RND(-TIMER):GOSUB121Ø:GOT
O19Ø
2Ø N$=INKEY$:IF N$=""THEN2ØELSER
ETURN
3Ø LINE(238,9)-(25Ø,32),PRESET,B
F:N1=INT(D/1Ø):N2=D-N1*1Ø:IF N2=
1THEN N2=14
4Ø IF N1=1THEN N1=14
5Ø IF N1=ØTHEN6ØELSE DRAW"BM238,
32;XN$(N1);BR4XN$(N2);":RETURN
6Ø DRAW"BM238,32;XN$(N2);":RETUR
N
7Ø IF CARD<14THEN RANK=CARD:SUIT
=1:RETURN
```

```
8Ø IF CARD<27THEN RANK=CARD-13:S
UIT=2:RETURN
9Ø IF CARD<4ØTHEN RANK=CARD-26:S
UIT=3:RETURN
1ØØ RANK=CARD-39:SUIT=4:RETURN
11Ø LINE(XC,YC)-(XC+11,YC+1),PSE
T,B:RETURN
12Ø LINE(2ØØ,Ø)-(232,39),PRESET,
BF:RETURN
13Ø LINE(XC,YC)-(XC+32,YC+39),PR
ESET,BF:RETURN
14Ø LINE(XC,YC)-(XC+32,YC+39),PS
ET,B:POKE2ØØ,XC+5:POKE2Ø2,YC+8:D
RAW"XN$(RANK);":POKE2ØØ,XC+24:PO
KE2Ø2,YC+8:DRAW"XS$(SUIT);":POKE
2ØØ,XC+17:POKE2Ø2,YC+32:DRAW"XC$
(SUIT);":IF SUIT=2OR SUIT=4THEN
PAINT(XC+11,YC+22),Ø,Ø
15Ø RETURN
16Ø IF COL(F,1)=ØORCOL(F,2)=ØTHE
N17ØELSERETURN
17Ø LINE(XC,43)-(XC+32,44),PRESE
T,B:RETURN
18Ø XC=165:YC=1:GOSUB13Ø:DRAW"BM
174,2Ø;U6F4D2U6BR5R2FD4GL2HU4E":
FORN=1TO5ØØ:NEXT:GOSUB13Ø:GOTO28
Ø
19Ø CLS:PRINT@1Ø,"'SOLITAIRE'":P
RINT@35,"PRESS:-":PRINT@64,"SPAC
EBAR TO DISPLAY NEXT CARD.  DOWN
 ARROW & COLUMN NO. TO MOVE    C
ARD FROM DISPLAY TO COLUMN.":PRI
NT@16Ø,"LEFT ARROW TO MOVE FROM
DISPLAY    TO TOP STACK."
2ØØ PRINT@224,"UP ARROW AND COLU
MN NO. TO MOVE    CARD FROM COLU
MN TO TOP STACKRIGHT ARROW AND C
OLUMN NUMBERS     TO MOVE CARDS
FROM COL TO COL":PRINT@352,"'Q'
TO QUIT AT ANY TIME.        MOVE
 ALL CARDS TO TOP STACKS TO    W
IN."
21Ø PRINT@453,"(DECK BEING SHUFF
LED)."
22Ø FORI=1TO4:STACK(I)=Ø:NEXT:FO
RI=1TO7:YC(I)=45:FORJ=1TO2Ø:COL(
I,J)=Ø:NEXT:NEXT
23Ø FORI=1TO52:DECK(I)=I:NEXT:FO
RI=1TO51:J=RND(52):N=DECK(J):DEC
K(J)=DECK(I):DECK(I)=N:NEXT:PRIN
T@453,"    PRESS <ENTER>.    ":G
OSUB2Ø
24Ø PMODE4,1:COLORØ,1:PCLS(1):SC
REEN1,1
25Ø DRAW"BM14,19Ø;XN$(14);":J=14
:FORI=2TO7:POKE2Ø2,19Ø:J=J+38:PO
KE2ØØ,J:DRAW"XN$(I);":NEXT
26Ø J=Ø:K=24:FORI=1TO7:J=J+1:FOR
 N=1TO J:K=K+1:COL(I,N)=DECK(K):
DECK(K)=-1:NEXT:NEXT:XC=-36:YC=4
5:FORI=1TO7:CARD=COL(I,I):GOSUB7
Ø:XC=XC+37:GOSUB14Ø:IF I=1THEN N
EXT ELSE LINE(XC,43)-(XC+32,44),
PSET,B:NEXT
27Ø CARD=DECK(3):GOSUB7Ø:XC=2ØØ:
YC=Ø:GOSUB14Ø:D=3:GOSUB3Ø
28Ø XC=165:YC=1:GOSUB13Ø:IF STAC
K(1)+STACK(2)+STACK(3)+STACK(4)=
52THEN3ØØELSE XC=166:YC=15:GOSUB
11Ø
29Ø GOSUB2Ø:IF N$="Q"THEN1ØELSEI
F N$=CHR$(32)THEN32ØELSEIF N$=CH
R$(1Ø)THEN53ØELSEIF N$=CHR$(9)TH
EN76ØELSEIF N$=CHR$(94)THEN39ØEL
SEIF N$=CHR$(8)THEN119ØELSEGOTO2
9Ø
3ØØ DRAW"BM9Ø,11Ø;F4NE4D6BR1ØH2U
6E2R4F2D6G2NL4BR9H2U8BR8D8G2NL3B
R3BE1ØD7F3E2F2E3U7BR7D1ØBR7U1ØF8
D2U1Ø":GOSUB2Ø:GOTO19Ø
31Ø REM START OF 'SPACE BAR'
32Ø DRAW"BM167,13;E4NL4NH4NU4NE4
NR4NF4D4":XC=2ØØ:YC=Ø:GOSUB13Ø
33Ø IF DECK(1)=-1THEN18Ø
34Ø D=D+3:IF DECK(D)>ØTHEN37Ø
35Ø D=D-1:IF DECK(D)>ØTHEN37Ø
36Ø D=D-1:IF DECK(D)<ØTHEN D=Ø:X
C=2ØØ:YC=Ø:GOSUB13Ø:GOTO33Ø
37Ø CARD=DECK(D):GOSUB7Ø:XC=2ØØ:
YC=Ø:GOSUB14Ø:GOSUB3Ø:GOTO28Ø
38Ø REM START OF 'UP ARROW'
39Ø DRAW"BM17Ø,13;U5L2E4F4L2D5L4
":XC=183:YC=15:GOSUB11Ø
4ØØ GOSUB2Ø:IF N$="Q"THEN28ØELSE
IF N$<"1"OR N$>"7"THEN4ØØ
41Ø F=VAL(N$):IF COL(F,1)=ØTHEN1
8Ø
42Ø IF F=1THEN F=14
43Ø I=2Ø:DRAW"BM186,13;XN$(F);":
IF F=14THEN F=1
44Ø I=I-1:IF COL(F,I)=ØTHEN44ØEL
SE CARD=COL(F,I):GOSUB7Ø
45Ø IF RANK<>STACK(SUIT)+1THEN18
Ø
46Ø STACK(SUIT)=STACK(SUIT)+1:XC
=SUIT*38-24:YC=Ø:GOSUB13Ø:GOSUB1
4Ø
47Ø XC=F*37-36:YC=YC(F):GOSUB13Ø
48Ø COL(F,I)=Ø:IF YC(F)>45THEN Y
C(F)=YC(F)-9
49Ø GOSUB16Ø
5ØØ IF COL(F,1)=ØTHEN YC=45:GOSU
B13Ø:GOTO28Ø
51Ø CARD=COL(F,I-1):GOSUB7Ø:YC=Y
C(F):GOSUB14Ø:GOTO28Ø
52Ø REM START OF 'DOWN ARROW'
53Ø DRAW"BM172,13;H4R2U5R4D5R2G4
":IF DECK(1)=-1THEN18Ø
54Ø CARD=DECK(D):GOSUB7Ø:IF RANK
<>1THEN61Ø
55Ø GOSUB 12Ø
56Ø STACK(SUIT)=STACK(SUIT)+1:XC
=SUIT*38-24:YC=Ø:GOSUB13Ø:GOSUB1
4Ø
57Ø IF D=ØTHEN37Ø
58Ø I=D:D=D-1:CARD=DECK(D):GOSUB
7Ø:XC=2ØØ:YC=Ø:GOSUB3Ø:IF D>ØTHE
N GOSUB14Ø
59Ø IF DECK(I+1)=-1THEN DECK(I)=
-1:GOTO28Ø
6ØØ DECK(I)=DECK(I+1):I=I+1:GOTO
59Ø
61Ø XC=183:YC=15:GOSUB11Ø
62Ø GOSUB2Ø:IF N$="Q"THEN28ØELSE
IF N$<"1"OR N$>"7"THEN62Ø
63Ø F=VAL(N$):IF F=1THEN F=14
64Ø DRAW"BM186,13;XN$(F);":IF F=
14THEN F=1
65Ø IF RANK=13AND COL(F,1)<>ØTHE
N18Ø
66Ø IF RANK=13 AND COL(F,1)=ØTHE
N GOSUB12Ø:COL(F,1)=DECK(D):XC=F
*37-36:YC=YC(F):GOSUB14Ø:GOTO57Ø
67Ø C1=RANK:S1=SUIT:I=2Ø
68Ø IF COL(F,1)=ØTHEN18Ø
69Ø I=I-1:IF COL(F,I)=ØTHEN69Ø
7ØØ CARD=COL(F,I):GOSUB7Ø:N=SUIT
+2:IF N>4THEN N=N-4
71Ø IF N=S1 OR SUIT=S1 THEN 18Ø
72Ø IF C1<>RANK-1 THEN 18Ø
73Ø GOSUB12Ø
74Ø YC(F)=YC(F)+9:XC=F*37-36:YC=
YC(F):GOSUB13Ø:COL(F,I+1)=DECK(D
):CARD=DECK(D):GOSUB7Ø:GOSUB14Ø:
GOTO57Ø
75Ø REM START OF 'RIGHT ARROW'
76Ø DRAW"BM167,11;U4R5U2F4G4U2L5
":XC=183:YC=15:GOSUB11Ø
77Ø GOSUB2Ø:IF N$="Q"THEN28ØELSE
IF N$<"1"OR N$>"7"THEN77Ø
78Ø F=VAL(N$):IF F=1THEN F=14
79Ø DRAW"BM186,13;XN$(F);":IF F=
14THEN F=1
8ØØ XC=165:YC=3Ø:GOSUB11Ø
81Ø GOSUB2Ø:IF N$="Q"THEN28ØELSE
IF N$<"1"OR N$>"7"THEN81Ø
82Ø T=VAL(N$):IF T=1THEN T=14
83Ø DRAW"BM169,28;XN$(T);":IF T=
14THEN T=1
84Ø J=2Ø:IF COL(T,1)=ØTHEN1Ø6Ø
85Ø J=J-1:IF COL(T,J)=ØTHEN85Ø
86Ø CARD=COL(T,J):GOSUB7Ø:HRANK=
RANK:HSUIT=SUIT:HYC=YC(F):HJ=J:I
=2Ø
87Ø I=I-1:IF I=ØTHEN18Ø
88Ø IF COL(F,I)=ØTHEN87Ø
89Ø CARD=COL(F,I):GOSUB7Ø:IF HRA
NK=RANK+1 THEN 92Ø
9ØØ I=I-1:IF I=ØTHEN18Ø
91Ø HYC=HYC-9:IF HYC=36THEN18ØEL
SE89Ø
92Ø N=SUIT+2:IF N>4THEN N=N-4
93Ø IF N=HSUIT OR SUIT=HSUIT THE
N 18Ø
94Ø HI=I:YC(F)=HYC
95Ø J=J+1:COL(T,J)=COL(F,I):COL(
F,I)=Ø:I=I+1:IF COL(F,I)>ØTHEN95
Ø
96Ø I=HI:XC=F*37-36:IF COL(F,1)=
ØTHEN LINE(XC,45)-(XC+32,183),PR
ESET,BF
97Ø GOSUB16Ø
98Ø IF HYC>45THEN YC(F)=HYC-9
99Ø LINE(XC,HYC)-(XC+32,183),PRE
SET,BF:I=2Ø
1ØØØ I=I-1:IF I=ØTHEN YC(F)=45:Y
C=45:GOSUB13Ø:GOTO1Ø3Ø
1Ø1Ø IF COL(F,I)=ØTHEN1ØØØ
1Ø2Ø CARD=COL(F,I):GOSUB7Ø:YC=YC
(F):GOSUB14Ø
1Ø3Ø J=HJ:YC(T)=YC(T)-9:XC=T*37-
36
1Ø4Ø IF COL(T,J)=ØTHEN28Ø
1Ø5Ø YC(T)=YC(T)+9:CARD=COL(T,J)
:GOSUB7Ø:YC=YC(T):GOSUB13Ø:GOSUB
14Ø:J=J+1:GOTO1Ø4Ø
1Ø6Ø I=2Ø:HYC=YC(F)
1Ø7Ø I=I-1:IF I=ØTHEN18Ø
1Ø8Ø IF COL(F,I)=ØTHEN1Ø7Ø
1Ø9Ø CARD=COL(F,I):GOSUB7Ø:IF RA
NK=13 THEN 112Ø
11ØØ I=I-1:IF I=ØTHEN18Ø
111Ø HYC=HYC-9:IF HYC=36THEN18ØE
LSE1Ø9Ø
112Ø YC(T)=45:YC(F)=45:HI=I:J=Ø:
XC=F*37-36:LINE(XC,45)-(XC+32,18
3),PRESET,BF:XC=T*37-36:YC=45
113Ø J=J+1:COL(T,J)=COL(F,I):COL
(F,I)=Ø:CARD=COL(T,J):GOSUB7Ø:GO
SUB14Ø:I=I+1:IF COL(F,I)=ØTHEN11
4ØELSE YC(T)=YC(T)+9:YC=YC(T):GO
SUB13Ø:GOTO113Ø
114Ø XC=F*37-36:I=HI-1:IF COL(F,
1)=ØTHEN YC=45:GOSUB13Ø:GOSUB17Ø
:GOTO28Ø
115Ø IF COL(F,2)=ØTHEN GOSUB17Ø
116Ø CARD=COL(F,I):GOSUB7Ø:YC=YC
(F):GOSUB14Ø
117Ø I=I+1:IF COL(F,I)=ØTHEN28ØE
LSE COL(F,I)=Ø:GOTO117Ø
118Ø REM START OF 'LEFT ARROW'
119Ø DRAW"BM171,13;H4E4D2R5D4L5D
2":IF DECK(1)=-1THEN18Ø
12ØØ CARD=DECK(D):GOSUB7Ø:IF RAN
K=STACK(SUIT)+1THEN55ØELSE18Ø
121Ø N$(Ø)="BUU4ER2FD4GL2H":
N$(1)="U4E2F2D2L3R3D2":
N$(2)="BU5ER2FDGL2GD2R4":
N$(3)="BR3L2HBU4ER2FDGLRFDG":
N$(4)="BR3U6G3DR4":
N$(5)="BR3L2HBU5R4L4D2R3FD2G":
N$(6)="BR3L2HU4ER2FBD2BL3R2FDG"
122Ø N$(7)="BU6R4DG3D2":
N$(8)="BR3L2HUEHUER2FDGL2R2FDG":
N$(9)="BUFR2EU2L3HUER2FD4":
N$(1Ø)="R2LU5LRUBR5R2FD4GL2HU4E"
:N$(11)="UDR3U6L2R4":
N$(12)="BR1HU4ER2FD4GLBUF2":
N$(13)="U6BR4G3F3":
N$(14)="R2LU6DLRBRBD5"
123Ø S$(1)="U6D3R4U3D6":
S$(2)="BR3EGL2HU4ER2FBF":
S$(3)="U6R3FD4GL3":
S$(4)="BU1FR2EUHL2HUER2F"
124Ø C$(1)="HUH2UH2UH2UHU3EUE2R3
F2E2R3F2DFD3GDG2DG2DG2DG":
C$(2)="L3ER2HU5G3L3H2U3E2R3FEH2U
3E2R3F2D3G2FER3F2D3G2L3H3D5F2L2"
:C$(3)="H3UH2UH3E3UE2UE3F3DF2DF3
G3DG2DG3":
C$(4)="L2EU7G3L2H2U4EUE7F7DFD4G2
L2H3D7FL2"
125Ø RETURN
```

Minimize bug problems and increase your programming expertise

ESCAPE FROM THE BUG ZONE

BY EUGENE VASCONI

Picture, if you will, a dimly lit workshop. The only perceptible activity centers around quickly moving fingers performing a computeresque symphony on the keyboard. Atop the paper-clogged table rests a smoking beaker of cocoa (what else?); in the distance, thunder rumbles. Line after line is entered from voluminous pages of THE RAINBOW, still warm after the journey from Prospect, Ky.

Meet Mr. I.N. Putter, an unsuspecting participant in this pseudo-Frankenstein scenario, who is unaware of his imminent, never-to-be-forgotten voyage into "The Bug Zone." (Music, please.)

Those of us having an affinity for the Color Computer have the chance to conquer The Bug Zone. Since we create most of them ourselves, we can quite easily eradicate our bug problems provided we are armed with the right ammunition: knowledge. Debugging is in itself an art, just like creating original programs. As such, it can be quite rewarding and a valuable aid toward better understanding programming.

This is written for less experienced computer users who are still refining their basic skills. It offers certain techniques about debugging and should make the prospect of entering listings and handling the debugging process less frightening.

What can be done to help Mr. Putter avoid an unnecessary journey into The Bug Zone? The most obvious ammunition is some understanding of how BASIC vernacular is applied to make the computer perform a particular task. For that, he must become familiar with the language: Where do O's go in a PLAY statement? How big a number can be used in a circle coordinate? When can remarks be eliminated? Attempting to enter a listing without at least an elementary understanding of the parameters is like playing a trumpet and not knowing what the valves are for.

Assuming we have done our homework, most bugs can be squashed during the actual typing phase. Listings such as those in the RAINBOW are very helpful, since they reflect what should appear on the computer screen (the 32-character width). As you enter the listing, pay particular attention that what you've typed matches the ends of the magazine lines. Make your information match the RAINBOW's exactly. Any variation could mean you've made an error. Take a moment after every line to check this. Use a marker to help avoid jumping to another line. Try a pen clipped to the side which points to the proper spot; as you continue, move it down the page. This technique can prevent line jumping or typing in the same information twice. As we all know, wading through tightly packed lines for hours can be hazardous to your eyesight; this will help.

Pay attention to addresses. A "28" typed in place of a "280" will eventually cause the program to act strangely or give a UL Error when the CoCo tries to locate an address that doesn't exist.

Even if there is no machine language in the program, always save it to tape before you run it. You may have typed in an accidental poke that will lock the unit, and your typing efforts will have been wasted when the power must be turned off to free the machine. An even wiser practice is to make in-progress saves while entering, so if the power goes off you won't have lost everything. I learned this lesson when hours of typing during a storm ended in a one-second power failure. My CoCo, trained as it was, got scared and forgot everything I had been teaching it. I now save, even when taking a short break. Keep a spare work tape around for this purpose.

If the program doesn't run properly go about your debugging and, especially if machine language is used, mark your corrections in the magazine, power down, reload and then edit. Again, save before trying. In this way, you'll be certain everything is correct from scratch and not helped by a past input from you or the program, which may be masking a problem.

Trips into The Bug Zone are most frequently scheduled by Syntax Errors. They are the most common and usually the easiest to find except for the ones pointed out by the computer that you check 30 times to no avail. Don't worry, they're there. They hide in things like those formulas the geniuses create with 200 parentheses and 50 multiplications. Remember, you need just as many ('s as you need)'s for it all to work. You don't need to understand the formula to be sure it is entered correctly, just count. Pay particular attention to semicolons, colons, periods and commas; they can look alike. This is an area many programmers need to watch, and I have been a violator as well. Let's start avoiding similar-looking letter/number combinations where possible, like `O(0)=I(1)` or `S$(8)=MN$(I)`. Of course, certain information must be programmed according to basic requirements such as `PLAY "L8B05"`, but notice how many of the following look similar to type-weary eyes especially on a television plagued with RF interference:

```
O - 0 - Q
I - 1
B - 8
S - $
N - M
G - 6
* - +
```

This is one area where thoughtful programming can assist the variety of debugging talents found in the CoCo community and make entering listings much easier.

TM Errors are solved simply by finding the place where a $ or string designation was either omitted or added by mistake. $ can only designate string material like `A$="DOG"` not `A$=76`. There are a few modifications to this rule, but basically when acquiring the TM Error, look for a dropped $.

The two errors that strike most fear into human hearts are the OD Error and

the FC Error. The first because it tells us we must wade through those 800 data lines we barely got through in the first place, and the second because it can be caused by something far away from the line listed as the error line.

The OD or (Out of Data) Error is not a particularly overwhelming problem. First, by yelling "out of data" the computer is saying that it wanted to read more than you gave it. That tells us we have one problem: The read amount doesn't match the amount of data available. It isn't necessary to understand the program to find the fix, simply check the read amount. If it says `10 FOR X=1 TO 99:READ A$(X):NEXT X`, we know we need 99 pieces of data to satisfy the computer. Be certain the 99 is correct. Now go to all those data lines and simply count the numbers between the commas; they should total 99. If not, you missed one somewhere, so double-check the magazine listing for proper alignment at the edges of the margins. This should help; if it doesn't, you will need to go line by line to find the problem.

Sometimes you get to the end and still can find no omitted data number. Is the computer wrong? Don't bet on it. The problem here is usually a period typed in place of a comma. This would make a statement like: `DATA 77,54,32,71. 10,` which has five separate numbers, turn into: `DATA 77,54,32.71,10,` which has only four. That little period is almost invisible among myriad lines of numbers, and most people would rather hear fingernails on a chalkboard than go through them one by one.

So, do it the easy way. If you find yourself in this position, check each `DATA` line with `EDIT`. If we enter `EDIT` and search for the period, `S.`, the cursor will move to the end of a good line or stop on the period we've been looking for. Change it and you're on your way. Don't forget to mark it in the book as a problem, in case you need to find it again.

We now come to the infamous FC (Function Call) Error. This accounts for most of the one-way trips into The Bug Zone.

Let's look at a few possibilities with an FC Error in Line 100: `100 PLAY"T255L355;O1;ABCDEF;XZ$;"`. Everything in the line looks OK. The 'O' is an 'O' and not a zero, quotes are there, 255 is the maximum legal amount for tempo and length, 1 is legal for the octave, the note names are proper, and the semicolons are really semicolons. So why still an FC Error in 100?

The only culprit left to cause a problem is `XZ$`, so now we must find `Z$` and check it out. If it's bad, the computer came to Line 100 in good faith and did what it was told, but when it smacked into `Z$`, it found someone asked it to do something illegal for the `PLAY` command. Presto — The Bug Zone.

To find `Z$`, we have to look at Line 50: `50 Z$="CCCAAABBBQ4ACEGF"`. Looks OK, but wait. The `Q` before the `4` doesn't make sense; it should be an `O` for octave. The mistake could just as easily have been any letter other than A through G or a command not allowed in `PLAY`. So, we see here how a mistake way up in the program can affect a line anywhere.

The `DRAW` command is another good place for problems, since countless letter/number combinations are used to create a particular screen image. How about an FC Error in Line 100 again: `100 DRAW"BM125,95;S63;E30R10D9 L12;XA$;"`.

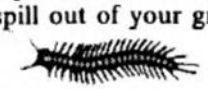
Aha, the obvious problem is the illegal number for the `SIZE` command. It can only be a maximum of 62. We'll fix it and all will be well. Unfortunately, we still have an FC Error, so let's find where `A$` is created: `10 A$="U4R3G9L 3U7H3U5R4785F9R6"`.

Remember, you must know something about the parameters allowed in `BASIC` commands. If you do, you find that `A$` calls for a line to be drawn 4,785 points to the right. When you check the listing you find out it should have been `R47D5`.

FC bugs can take on a wide variety of disguises: `PUT` parameters must equal their repsective `GET` partners, `PRINT @XXX`, can be no more than 511, and line commands must not exceed 256 horizontal and 192 vertical. The possibilities are endless, and we can't cover them here. The solution is to break the problem down into small parts starting with the called line number, then search for the other areas brought into play to make that line work. By doing this, you'll eventually escape The Bug Zone.

A final hint concerning breaking the large program down into small parts. Don't be afraid to insert a `GOTO` or `STOP` command in a listing to test the waters. Things go by rapidly when the computer works, so you need to center in on your problem.

If, for example, a `PAINT` command makes color spill out of your graphics try this:



```
10 DRAW"BM125,96R40D40L40U38"
20 PAINT(130,100),4,3
```

Put `15 GOTO 15` in and check whether the line makes a complete container for the paint. You won't see it until it is too late. Then fix the `DRAW` statement and remove the `GOTO`. If the container is complete and there is still a problem, double-check the paint coordinates by changing `PAINT` to a `PSET(130,100)`, put in `25 GOTO 25` and see where the dot sets. If the listing is correct you won't need this, but mistakes do happen.

For number variables like `10 X=X*2` or `200 LINE(X,30)-(X+7,120), PSET`, the program will work for a moment unitl X gets too big to be legal. If it's part of a larger program, put a stop on Line 201 and ask the computer to `?X`. What if it says `X=250`? No good, because the second part of the line command adds 7 to X and therefore equals 257, which is not allowed. Your problem then becomes the `X=X*@` statement, which maybe needed to be an `X=X+2`.

So, don't be discouraged by an occasional voyage into The Bug Zone. Those more experienced have spent many an hour saying nasty things to their CoCo while learning to discover elusive bugs. Just remember to:

1. Learn the rules
2. Compare with the listing
3. Save to tape before running
4. Check each line before continuing
5. Double-check address numbers
6. Be careful of similar characters
7. Narrow down the problem
8. Don't be in a hurry

You will soon find you've acquired a key that unlocks the many doors you may encounter on your next trip into The Bug Zone.

Eugene Vasconi is a helicopter pilot in San Antonio, Texas, as well as a musician and free-lance television producer. His major interests on the CoCo are graphics and music.

(Questions about this article may be directed to the author at 12474 Starcrest #204, San Antonio, TX 78216, 512-496-5783. Please enclose an SASE for a reply when writing.)

Create your own zany puzzles

CoCo-Nect-A-Dot

By Eric White

From time to time, I enjoy exploring the graphics capabilities of the CoCo. The Color Computer's Extended BASIC is a very powerful construction set. It can provide hours of fun and occasionally frustration. With it you can successfully create a variety of interesting and useful programs.

CoCo-Nect-A-Dot is one such program, using commands such as DRAW, LINE, PAINT, GET and PUT as the main building blocks. Explored in this program is the incorporation of a simple data interpreter. By coding each dot puzzle in a standard format, the puzzle can be displayed by simply processing the puzzle data through the interpreter. Once the user interface is finished and operating, additional puzzles can be created by entering more puzzle data.

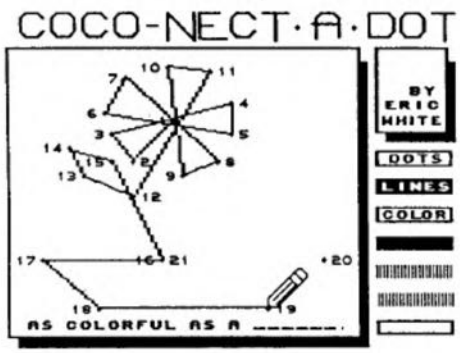
CoCo-Nect-A-Dot accepts input from either the joystick port or keyboard. Press the firebutton on the desired joystick to select it as the input device. I will be using the keyboard throughout the rest of the program operation instructions. The active keys are the arrow keys and the space bar.

There are three menu options available: Dots, Lines and Color. To select a new puzzle, move the pencil with the arrow keys until the point of the pencil is on the box marked Dots, then press the space bar.

The Lines menu item is automatically selected when the program first starts, and after each new puzzle has been constructed. Lines is the drawing mode used to connect the dots. Position the pencil point where you want to tack a line down and press the space bar. Next, move the pencil to where you want the line to stop and press the space bar a second time.

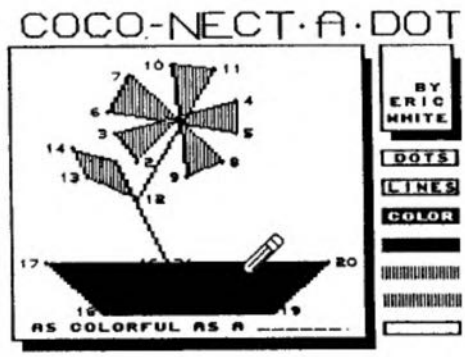
Once your puzzle is complete, you may want to add a little color. Move the pencil point to the box marked Color and press the space bar. A flashing box will surround one of four color bars located at the bottom right of the screen. Use the up- and down-arrow keys to move the flashing box to the desired color and press the space bar to select that color. Next, position the pencil point where you want to add color and press the space bar to fill the area.

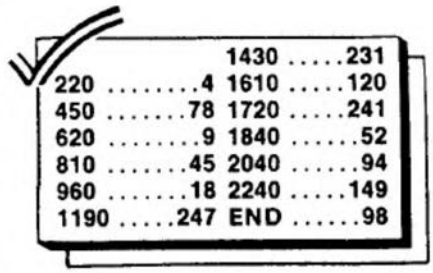
The program consists of nine sections. Lines 50 to 390 are for device input, lines 400 to 650 are pencil position update, lines 660 to 980 draw the main screen, and lines 990 to 1170 are the puzzle data interpreter. Lines 1180 to 1320 do Hi-Res text printing, lines 1330 to 1630 are the menu options, lines 1640 to 1760 have the puzzle data, lines 1770 to 2200 are the Hi-Res character data and program variables, and lines 2210 to 5000 are the start-up display.



To create a new puzzle, simply add puzzle data in this order: x position (0-255) for hint message, y position (0-191) for hint message, and the hint message string. Next, each point of the puzzle is described by the x (0-255), and y (0-191), R or L (which puts the dot's number on the right or left of the dot), and 0,0,END (which is used to end the puzzle's data).

Note: The x coordinate must be divisible by 4 and the y coordinate must be divisible by 3, so the pencil point can be placed exactly on the dots. With a little planning, almost any simple shape can be encoded into a *CoCo-Nect-A-Dot* puzzle.





```
                    1430 .....231
220 .....4          1610 .....120
450 .....78         1720 .....241
620 .....9          1840 .....52
810 .....45         2040 .....94
960 .....18         2240 .....149
1190 .....247       END .....98
```

The listing: COCONECT

```
1Ø ' 85Ø3.1Ø/23:ØØ MOD:86Ø7.26
2Ø CLEAR2ØØ:PCLEAR8
3Ø 'POKE65495,Ø'SPEED POKE
4Ø GOTO 176Ø
5Ø '
6Ø 'JOYSTICK ROUTINES
7Ø '
8Ø PLAY"L15ØABCDEFG":A$=INKEY$
9Ø FR=Ø:P=PEEK(6528Ø):AR=JOYSTK(
Ø):BR=JOYSTK(1):AL=JOYSTK(2):BL=
JOYSTK(3)
1ØØ IF KJ<>1 THEN IF INKEY$<>""
THEN KJ=1:GOTO 8Ø
11Ø IF KJ<>2 THEN IF P=126 OR P=
254 THEN KJ=2:GOTO 8Ø
12Ø IF KJ<>3 THEN IF P=125 OR P=
253 THEN KJ=3:PLAY"L15ØABCDEFG":
GOTO8Ø
13Ø ON KJ GOTO 19Ø,26Ø,31Ø
14Ø '
15Ø 'DEVICE 1 = KEYBOARD
16Ø 'DEVICE 2 = RIGHT JOYSTICK
17Ø 'DEVICE 3 = LEFT  JOYSTICK
```

```
18Ø '
19Ø 'KEYBOARD DETECTION
2ØØ IF PEEK(345)=247 THEN POKE 3
45,255:FR=1:RETURN
21Ø IF PEEK(343)=247 THEN POKE 3
43,255:A=A-1
22Ø IF PEEK(341)=247 THEN POKE 3
41,255:B=B-1
23Ø IF PEEK(342)=247 THEN POKE 3
42,255:B=B+1
24Ø IF PEEK(344)=247 THEN POKE34
4,255:A=A+1
25Ø RETURN
26Ø 'RIGHT JOYSTICK DETECTION
27Ø A=AR:B=BR
28Ø IF (P AND 3)<3 THEN FR=1
29Ø RETURN
3ØØ '
31Ø 'LEFT  JOYSTICK DETECTION
32Ø '
33Ø A=AL:B=BL
34Ø IF (P AND 3)<3 THEN FR=1
35Ø RETURN
36Ø '
37Ø PCOPY5TO1:PCOPY6TO2:PCOPY7TO
3:PCOPY8TO4:RETURN
38Ø FORU=1TO4:PCOPY U TO U+4:NEX
T:RETURN
39Ø PUT(21Ø,79)-(253,119),K,PSET
:RETURN
4ØØ '
41Ø 'DRAW & GET PENCIL
42Ø '
43Ø PMODE4,1:PCLS1:COLORØ
44Ø DRAW"BMØ,22;"+P$
45Ø GET(Ø,Ø)-(23,22),P
46Ø PCLS:COLOR1:DRAW"BMØ,22;U5E1
7R2F3D2G17L5":PAINT(2,17),1,1
47Ø GET(Ø,Ø)-(23,22),M
48Ø GOSUB37Ø:X=99:PRINT@96+32+32
,"";:GOSUB227Ø
49Ø PRINT:PRINT"    PRESS ANY KE
Y TO BEGIN    ":EXEC44539
5ØØ PMODE4:SCREEN1,1
51Ø IF FR=1 THEN 52Ø ELSE 55Ø
52Ø IF X>194 THEN 134Ø
53Ø PLAY"T2ØØAB":IF L=Ø AND TP=Ø
 THEN L1=X:L2=Y:L=1:GOTO55Ø ELSE
 IF TP=1 THENGOSUB37Ø:PMODE3,1:
PAINT(X,Y),CO,1:PMODE4,1:GOSUB38
Ø:GOTO56Ø
54Ø IF L=1 THEN PMODE3,5:COLOR1:
LINE(L1,L2)-(X,Y),PSET:PMODE4:L=
Ø
55Ø GOSUB9Ø:SCREEN1,1:IF A=C AND
 B=D THEN 51Ø
56Ø C=A:D=B
57Ø X=C*4:IF X>232 THEN X=232
58Ø IF X<Ø THEN X=Ø
59Ø Y=D*3:IF Y<24 THEN Y=24
6ØØ IF L=1 AND TP=Ø THEN PCOPY5T
O1:PCOPY6TO2:PCOPY7TO3:PCOPY8TO4
:PMODE3,1:COLOR1:LINE(L1,L2)-(X,
Y),PSET:PMODE4,1:GOTO62Ø
61Ø PMODE4,5:GET(OX,OY-22)-(OX+2
3,OY),S,G:PMODE4,1:PUT(OX,OY-22)
-(OX+23,OY),S,PSET
62Ø PUT(X,Y-22)-(X+23,Y),M,OR
63Ø DRAW"BM=X;,=Y;C@"+P$
64Ø OX=X:OY=Y
65Ø GOTO55Ø
66Ø '
67Ø 'DRAW SCREEN
68Ø '
69Ø PMODE4,5:PCLS1
7ØØ BD=1:S=18:MX=8:MY=1:DRAW"S15
":M$="COCO  NECT  A  DOT":GOSUB1
25Ø
71Ø DRAW"BM81,8;S4U6BM166,8;"+D$
+"BM198,8;"+D$
72Ø PMODE3,5:COLOR1
73Ø LINE(Ø,2Ø)-(2ØØ,185),PSET,B
74Ø LINE(6,186)-(2ØØ,191),PSET,B
F
75Ø LINE(2ØØ,28)-(2Ø5,191),PSET,
BF
76Ø LINE(21Ø,2Ø)-(25Ø,68),PSET,B
77Ø MX=214:MY=113:M$="COL":GOSUB
119Ø
78Ø MX=236:MY=113:M$="OR":GOSUB1
19Ø
79Ø MX=214:MY=97:M$="LINES":GOSU
B119Ø
8ØØ MX=218:MY=81:M$="DOTS":GOSUB
119Ø
81Ø LINE(251,28)-(256,73),PSET,B
F
82Ø LINE(216,69)-(256,73),PSET,B
F
83Ø MX=23Ø:MY=42:M$="BY":GOSUB12
3Ø
84Ø MX=216:MY=5Ø:M$="ERIC":GOSUB
123Ø
85Ø MX=214:MY=59:M$="WH":GOSUB12
3Ø
86Ø MX=228:MY=59:M$="I":GOSUB123
Ø
87Ø MX=234:MY=59:M$="TE":GOSUB12
3Ø
88Ø BD=Ø:PMODE3,5:COLOR1:LINE(21
Ø,128)-(253,135),PSET,BF
89Ø COLOR2:LINE(21Ø,144)-(253,15
1),PSET,BF
9ØØ COLOR3:LINE(21Ø,16Ø)-(253,16
7),PSET,BF
91Ø COLORØ:LINE(2ØØ,15)-(2Ø8,18)
,PSET,BF:COLOR1:LINE(21Ø,176)-(2
53,183),PSET,B
92Ø LINE(21Ø,79)-(253,87),PSET,B
93Ø LINE(21Ø,95)-(253,1Ø3),PSET,
B
94Ø LINE(21Ø,111)-(253,119),PSET
,B
95Ø GET(21Ø,79)-(253,119),K,G
96Ø PUT(212,96)-(251,1Ø2),L,NOT
97Ø GOSUB1ØØØ:GOTO43Ø
98Ø RETURN
99Ø '
1ØØØ ' READ DATA & PLOT POINTS
1Ø1Ø '
1Ø2Ø DN=1
1Ø3Ø PMODE4,5
1Ø4Ø COLORØ
1Ø5Ø READ MX,MY,M$:IF M$="OUT"TH
EN RESTORE:GOTO1Ø5Ø ELSE GOSUB11
9Ø
1Ø6Ø READ PX,PY,OF$:IF OF$="END"
 THEN 115Ø
1Ø7Ø IF OF$="R "THEN OF=8
1Ø8Ø IF OF$="L "THEN OF=-6:IF DN
>9 THEN OF=-12
1Ø9Ø DRAW"BM=PX;,=PY;"+D$:PX=PX+
OF:PY=PY-1:DRAW"BM=PX;,=PY;A1BD3
"
11ØØ IF DN<1Ø THEN DRAW N$(DN):G
OTO113Ø
111Ø DN$=STR$(DN)
112Ø DRAW N$(VAL(MID$(DN$,2,1)))
+"BL6BU6"+N$(VAL(RIGHT$(DN$,1)))
113Ø DN=DN+1
114Ø GOTO1Ø6Ø
115Ø 'FINISHED READING DATA
116Ø DRAW"A@"
117Ø RETURN
118Ø '
119Ø 'HI-RES PRINTING SHOP
12ØØ 'NEEDS M$="MESSAGE"
121Ø '      MX=X POS
122Ø '      MY=Y POS
123Ø DRAW"S4":S=8
124Ø PMODE4,5:COLORØ
125Ø FORM=1 TO LEN(M$)
126Ø N=ASC(MID$(M$,M,1))-64
127Ø IF N=-32 THEN MX=MX+6:NEXTM
:RETURN
128Ø IF N=Ø THEN LINE(MX,MY+4)-(
MX+6,MY+4),PSET:MX=MX+8:NEXT:RET
URN
129Ø IF N=-18 THEN PSET(MX+2,MY+
4,Ø):NEXTM:RETURN
13ØØ DRAW"BM=MX;,=MY;A1C@"+A$(N)
131Ø BX=MX+1:DRAW"BM=BX;,=MY;A1C
Ø"+A$(N)
132Ø MX=MX+S:NEXTM:RETURN
133Ø '
134Ø ' MENU
135Ø '
136Ø IF X<21Ø THEN55Ø
137Ø IF Y>79 AND Y<88 THEN 142Ø
138Ø IF Y>96 AND Y<1Ø4 THEN 149Ø
139Ø IF Y>111 AND Y<12Ø THEN 153
Ø
14ØØ GOTO55Ø
141Ø '
142Ø 'DOTS
143Ø '
144Ø PLAY"T2ØØCD":PMODE4,5:SCREE
N1,1:GOSUB39Ø
145Ø PUT(212,8Ø)-(251,86),L,NOT
146Ø COLOR1:LINE(2,21)-(198,184)
,PSET,BF
147Ø L=Ø:GOSUB1ØØØ:GOSUB39Ø:PUT(
212,96)-(251,1Ø2),L,NOT:GOSUB37Ø
:PMODE4:SCREEN1,1:TP=Ø:GOTO57Ø
148Ø '
149Ø 'LINES
15ØØ '
151Ø PLAY"T2ØØCD":L=Ø:TP=Ø:PMODE
4,5:GOSUB39Ø:PUT(212,96)-(251,1Ø
2),L,NOT:GOSUB37Ø:GOTO57Ø
152Ø '
153Ø 'COLOR
154Ø '
155Ø PLAY"T2ØØCD":OO=B:PMODE4,5:
GOSUB39Ø:PUT(212,112)-(251,118),
L,NOT:GOSUB37Ø
156Ø L=Ø:TP=1:PMODE3
157Ø GOSUB9Ø
158Ø IF K>1 THEN B=INT(B/16) ELS
E B=B AND 3
159Ø COLOR1:LINE(2Ø9,B*16+127)-(
255,B*16+136),PSET,B:COLORØ:LINE
(2Ø9,B*16+127)-(255,B*16+136),PS
ET,B:OB=B
16ØØ IF FR=Ø THEN 157Ø
161Ø CO=PPOINT(235,B*16+129)
162Ø B=OO:GOTO57Ø
163Ø '
164Ø ' DOT DATA
165Ø '
166Ø DATA 4,175,WHEN YOU WISH AP
ON A @@@@.,1ØØ,36,R ,14Ø,148,R ,
3Ø,72,L ,16Ø,72,R ,52,148,L ,1ØØ
,36,L ,Ø,Ø,END
167Ø DATA 16,175,YOU ARE MY SWEE
T @@@@@.,1Ø8,167,R ,168,99,R ,17
2,67,R ,14Ø,48,R ,1Ø4,67,L ,76,4
8,L ,44,67,L ,48,99,L ,1Ø8,167,L
,  Ø,Ø,END
168Ø DATA 12,175,LIKE A BOLT OF
@@@@@@@@.,176,36,L ,48,95,L ,1ØØ
,1Ø3,R ,28,169,L ,164,1Ø6,R ,12Ø
,87,L ,176,36,R ,Ø,Ø,END
169Ø DATA 44,175,LITE AS A @@@@.
,56,16Ø,L ,172,1Ø4,R ,8Ø,36,L ,5
6,16Ø,R ,16Ø,32,R ,8Ø,36,R ,172,
1Ø4,L ,Ø,Ø,END
17ØØ DATA 12,175,SHINES LIKE A @
@@@@@@.,24,68,R ,99,16Ø,L ,64,68
,L ,99,36,L ,136,68,R ,99,16Ø,R
,184,68,L ,24,68,L ,8Ø,36,L ,12Ø
,36,R ,184,68,R ,Ø,Ø,END
171Ø DATA 22,175,YOUR MY @@@ @@
@@@.,124,132,L ,148,72,L ,164,82
,R ,152,1Ø8,R ,132,118,L ,156,56
,R ,36,56,L ,68,132,R ,184,132,R
 ,136,148,L ,56,148,R ,16,132,R
,68,132,L ,Ø,Ø,END
172Ø DATA 22,175,YOU @@@@@ UP MY
 LIFE.,72,32,R ,128,32,R ,156,1Ø
4,R ,88,1Ø4,L ,72,148,L ,84,16Ø,
R ,76,168,L ,124,168,R ,116,16Ø,
L ,128,148,R ,112,1Ø4,R ,44,1Ø4,
L ,72,32,L ,Ø,Ø,END
173Ø DATA 12,176,AS COLORFUL AS
A @@@@@@.,96,6Ø,L ,72,84,R ,6Ø,6
8,L ,128,5Ø,R ,128,68,R ,56,56,L
 ,68,36,L ,12Ø,84,R ,1ØØ,92,L ,9
2,3Ø,L ,116,32,R ,72,1Ø4,R ,44,9
2,L ,36,76,L ,6Ø,84,L ,88,14Ø,L
,2Ø,14Ø,L ,52,168,L ,148,168,R ,
```

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 to the joystick port of your CoCo Max. Comes with programs to start you thinking		$39.95

SOFTWARE

Magazines: 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.		
Back Issues: Australian CoCo Magazine. (Aug '84 to now.) **Please Note:** Some months out of stock.	Australian CoCo 1986 Sept 1984 — 1985	$3.75 $3.00
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	$10.00 $42.00 $75.00
Back issues of CoCoOz are always available	Each DISK Subscription on disk, 12 months	$10.95 $118.26
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	$10.00
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 $39.75 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	$39.75
Books: HELP: A quick reference guide for CoCo users.		$9.95
MiCo HELP: A quick reference for owners of MC 10 computers.		$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.

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 — 16K

ALIEN ..... STUART SANDERS
QWERL ..... DARRELL BERRY
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 ..... WAL STEPHENSON
BREAKOUT ..... WHY/BILT

Best of CoCoOZ #9 GAMES — 32K

TRIOMINO ..... BOB DELBOURGO
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
LABYRINT ..... JAMES REDMOND
TANK ..... C. STEWART

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

Best of #11 EDUCATION 3 — Disk only CHATWIN MANOR ..... BOB HORNE

TAPE $10 each

DISK $16 each

GOLDSOFT

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

ORDER FORM

Goldlink	12 Months	$ 39.75 .....
Australian CoCo	12 Months	$ 39.95 .....
	6 Months	$ 24.75 .....
	1 Month	$ 4.50 .....
CoCoOz on Tape	12 Months	$ 75.00 .....
	6 Months	$ 42.00 .....
	1 Month	$ 10.00 .....
CoCoOz on Disk	12 Months	$118.25 .....
	6 Months	$ 59.50 .....
	1 Month	$ 10.95 .....
Softgold	12 Months	$ 35.00 .....
	6 Months	$ 21.35 .....
	1 Month	$ 3.75 .....
Softgold on Tape	6 Months	$ 42.00 .....
(CoCo Progs)	1 Month	$ 10.00 .....
Softgold on Disk	6 Months	$ 59.50 .....
(CoCo Progs)	1 Month	$ 10.95 .....
Golddisk		
(T1000 Quarterly)	1 Month	$ 10.95 .....
MiCoOz	12 Months	$ 75.00 .....
(MC–10 Progs)	6 Months	$ 42.00 .....
	1 Month	$ 10.00 .....

Or, Charge my credit Card Monthly
(Tapes &/or Disks only)

CoCoOz on Tape	$ 10.00
CoCoOz on Disk	$ 10.95
Softgold on Tape	$ 10.00
Softgold on Disk	$ 10.95
Golddisk (Quarterly)	$ 10.95

Additional Requirements

.....

.....

.....

Sub No: .....

Name: .....

Address: .....

.....

Phone No. .....

Credit Card No. .....

Please find enclosed:–
CHQ / Money Order / Cash

Please charge my:–
Mastercard / Bankcard / Visa

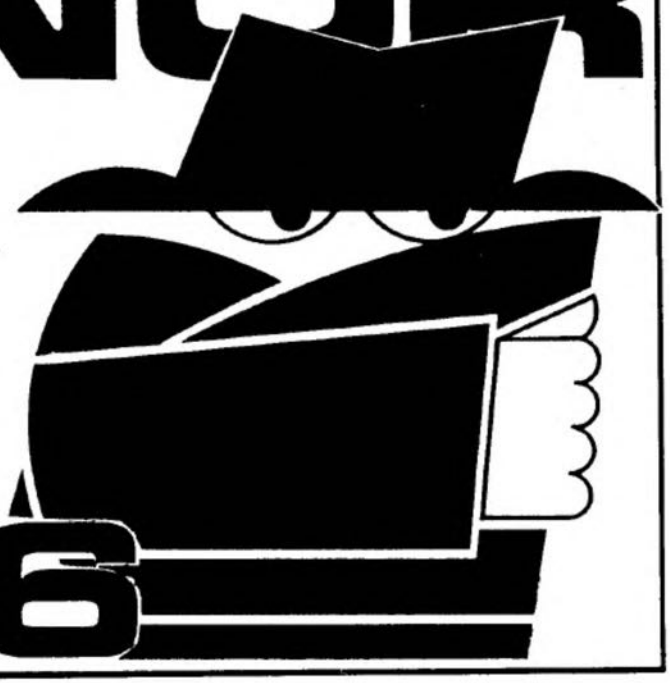
Authorised Amount: .....

Signed: .....

CHATWIN MANOR

By Bob Horne

BEST OF CoCoOz #11
EDUCATION 3



Available from:
GOLDSOFT
PO Box 1742,
SOUTHPORT

$16
DISK
ONLY

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	02 649 3793
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSKO MCKAY	02 624 3353
CHATSWOOD	BILL O'DONNELL	02 419 6081
COLYTON	HERMAN FREDRIKSSON	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
MINTO	GRAHAM POLLOCK	02 603 5028
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 PRATT	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 6549
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 6549
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
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
CLAYFIELD	JACK FRICKER	07 262 8869
COLL'WOOD PK	AND'W SIMPSON	07 288 5206
IPSWICH	MICK MURPHY	07 271 1777
PINE RIVERS	BARRY CLARKE	07 204 2806
SOUTH WEST	BOB DEVRIES	07 373 7816
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	071 71 5301
CAIRNS	JEFF LARSEN	070 54 7127
DALBY	MERRICK TANSKY	074.62.3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 51 0015
GYMPIE	BERT LLOYD	071 8219100
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	DEBBIE DORFIELD	074 65 3177
TOOWOOMBA	LEN GERSEKOWSKI	076 35 8264
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064

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:

DEVONPORT	JEFF BEST	004 24 6759
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
UPR F'TREE GLY	RORY DOYLE	03 758 2671

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
MAFFRA	MAX HUCKERBY	051 45 4315
MOE	JIMMY WELSH	051 27 6984
MORNINGTON	MICHAEL MONCK	03 789 7997
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	LEIGH DAVIES	051 74 5552
WONTHAGGI	LOIS O'MEARA	056 72 1593
YARRAWONGA	KEN SPONG	057 44 1488

WA:

PERTH	IAIN MACLEOD	09 448 2136
GIRRAWHEEN	HANK VILLEMSEN	09 342 7639
KALGOORLIE	TERRY BURNETT	090.21.5212

CANADA - CoCo:

Ontario	Richard Hobson	416 293 2346

special interest groups

TEACHERS' INTEREST GROUP

BRISBANE	BOB HORNE	07 281 8151

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
SYDNEY	GRAHAM POLLOCK	02 603 5028
WARRNAMBOOL	GARY FURR	055 62 7440

TANDY 1000 / MS DOS:

NSW:

GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY WEST	ROGER RUTHEN	047.39.3903
WYONG	JOHN WALLACE	043 90 0312

QLD:

BRISBANE

NORTH	BRIAN DOUGAN	07 30 2072
SOUTH	BARRY CAWLEY	07 390 7946

GOLD COAST	GRAHAM MORPHETT	075 51 0015

SA

PORT LINCOLN	BILL BOARDMAN	086 82 2385

VIC:

MELBOURNE	TONY LLOYD	03 882 4664

FORTH:

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

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
HUB	002 49 4405

1200/75 BAUD TANDY INFORMATION

GOLDLINK	VIATEL *642#
MOUSETEX	WATCH 642!
PRESTEL (ENGLAND)	0011-44-681-25411

(Visitor ID = 4444444444 PW = 4444)

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
STUART HALL	939765790
BOB KENNY	665122050
JEFF LARSEN	705471270
IAIN MACLEOD	944821360
CHRIS NAGLE	689523360
RICHARD PANKHURST	280717870
ROSS PRATT	648230650
JUDY RUTLEDGE	285350000
RICCAY SCHMAHL	298151500
ARTHUR SLADE	262289400
DARCY O'TOOLE	755105770
RON WRIGHT	352924510

ROBOTICS:

BOWEN	TONY EVANS	077 86 2220
GOLD COAST	GRAHAM MORPHETT	075 51 0015
SYDNEY	GEOFF FIALA	02 84 3172
WAGGA WAGGA	CES JENKINSON	069 25 2263

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

GOLDLINK

COM. STATION 692

ON

Telecom VIATEL®
AUSTRALIA'S NATIONAL VIDEOTEX SERVICE

BULLETIN BOARDS
TANDY COMPUTERS
ATARI COMPUTERS
COMMODORE COMPUTERS
FAST FLORIST
REVIEWS
HARDWARE & SOFTWARE
SHOP

YES WE EVEN TALK BACK !!