For your TANDY Color Computer

Registered by Australian Post — Publication No. QBG 4009

NZ $3.95 PNG K5.95

$4.95

AUSTRALIAN RAINBOW

January, 1987

No.67

Starting off a brand new, shining year with...

- ● CoCo and Multi-pac power supplies
- ● Disk Organization
- ● A CoCo Robot Operating System
- ● Games!! ● Hamming it up
- ● Using your CoCo as a research tool

BETTER FOR MICROS

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

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

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

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

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

* Coming

Telecom Australia
Better for Business

TM657

Telecom

VIATEL®

AUSTRALIA'S NATIONAL VIDEOTEX SERVICE

APPLICATION FORM

DATE OF APPLICATION / /

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

section 1

PLEASE TICK APPROPRIATE BOX TO INDICATE SERVICE REQUIRED

BUSINESS SERVICE ☐ NON-BUSINESS SERVICE ☐

(CHARGES INCURRED ON BUSINESS SERVICES ARE USUALLY TAX DEDUCTIBLE)

SURNAME (OR BUSINESS NAME IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS NUMBER/STREET

SUBURB/CITY STATE POSTCODE

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

section 2

CONTACT NAME (IF BUSINESS SERVICE) GIVEN NAMES

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

SUBURB/CITY STATE POSTCODE

CONTACT TELEPHONE NUMBER (INCLUDING STD CODE)

section 3

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

PLEASE INDICATE TYPE OF EQUIPMENT USED TO ACCESS VIATEL

special instructions

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

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

telecom use only

DTE PP VN

BG SC CI

REF

SPAN OVERLAY NO. 140

REGISTRATION AND SUBSCRIPTIONS

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

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

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

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

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

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

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

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

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

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

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

AUSTRALIAN RAINBOW

CONTENTS

January Special

DMP–105 Printer

$310

Sorry, offer closes 31/1/87

Bayne & Trembath

3 Boneo Rd., Rosebud, Victoria 3940
Ph: (059) 86 8288. A/H: (059) 85 4947

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.

PRINT #-2

Wow it's 1987!

I hope you are having/had a great holiday over the Christmas period!

As this magazine is being prepared and published well before the Christmas, I can't say what I did over Christmas can I!

However we do trust that you enjoy the holiday, and that you have/are using the time to come up with some interesting ideas to use with your CoCo in 1987!

This year I hope to be adding a whole lot more to Australian Rainbow Magazine, like hardware modifications, Australian-orientated reviews, more utilities, more OS-9, more projects, and more machine language!

And we'll continue to present articles which are on the leading edge of current thinking in the CoCo world.

We're still fairly busy here - we still have to catch up on doing next month's magazine. You see, it works this way: by the time you get say, January's magazine, we're about to finish February's magazine and we've usually already started on March's!

Unlike other publications, we don't work too far in advance - sure we have material 'up our sleeves', but by keeping leadtimes short we can usually present information & news which is very timely.

CoCo III

It's been a few months since the CoCo III was released, and I'm amazed at all the information that has come in for this machine!

Already there is a new version available - the Extended Basic v 2.1! This particular version fixes all the bugs the older model had (we're told), like the left-set screen on TV models, (with the exception of the National 34cm Colour TV which could handle the v2.0) - I had someone ring me and he said "Wow! I can read my screen perfectly - centered and all!"

WHAT they REALLY fixed I will be able to tell you about in next month's edition!

The CoCo III Disk/Tape is finally completed! If you want a copy, you can order it through us at the usual price of $10 (for a tape) or $16 (for a disk).

You'll find a whole lot of programs on this disk/tape, ranging from graphics pictures to games to a Biorhythm program - all designed for the CoCo III.

User Groups

As I mentioned in a previous issue, I was planning to drop in on the User Groups of the Brisbane area.

The last one I visited was the User Group in Ipswich. Led by Mick Murphy, the group (in my opinion) has great potential. The whole group is what one might call a REAL meeting. The things they are doing include monthly newsletters, speeches & tutorials - the works!

Now, if you want to give Mick a call about something (a computing problem, etc) and live in the general Ipswich area, give him a call on (07) 271 1777.

I had a break from visiting the groups for the rest of December, and am starting up in February again.

Goldlink.

We've had a few of the 'serious' users knocking Goldlink because they say it is too frivilous.

Clubroom is nothing if not time wasting and frivilous, but there are more boards on Goldlink than Clubroom.

In fact, for the 'serious' users there are the Tandy, OS-9 and IBM/Tandy 1000 boards, and these operate at the same speed and under the same system as Clubroom, except that the frivilous messages are to a large extent vetted off these boards.

Goldlink can be whatever you want it to be, but you have to be involved, you have to send the messages to get the replies you need!

CM 8's - A Sell Out!

As Graham said, way back in March when they had to be ordered, who would have liked the responsibility of ordering the CM 8 monitor for the new CoCo?

Someone had to do it, and he miscalculated - he, and we, just didn't imagine that so many of you would require CM 8's so early on!

So Tandy sold out - VERY quickly!

But they'll be back in stock soon, and we can assure you, they ARE worthwhile purchasing!

OS 9 Level 2

OS 9 Level 2 is due out at the end of this month.

Don't be surprised if its late, apparently there are problems.....

As OS 9 Level 2 is held up, this means that programs dependant on it, such as DeskMate 3 will also be held up.

Gives you a bit more time to get to know Basic and get a program in for the CoCo 3 programming competition! By the way - the competition closes soon, so hurry with those entries!

CoCo 2 Users

This year, our aim for Australian Rainbow Magazine is to use it to present state of the art, serious users' programs and articles for both the CoCo 2 and the CoCo 3.

Provided you share what you are doing, and what you are discovering with us all, we expect to be able to fulfill that aim admirably for you whether you own a CoCo 3 or a CoCo 2!

Conf '87

Conf '87 will be in Sydney this year - where we are just not sure at this stage - but we'll let you know soon!

And it looks like Graham MAY be in Melbourne for a one day conference/meet in March - more news on that next month!

Plan to be at both these events if you can.

Right now there is more happening in the CoCo world than ever before. Conf '87 and the big Melbourne Meet are the best ways we know of, to keep up with it all!

Alex

LETTERS

Dear Graham,

I am writing once again to seek assistance and advise you of several matters.

I indicated in my last letter that I would be preparing an index to Rainbow, from issue 1, using Pro-Color File. Well it's finished. I was waiting for the Heralded Geoff Tolpot index, but as yet progress has not reached the names of the Australian Rainbow or CoCo.

With 2200 records I have had a very good data base to compare the Australian Rainbow pre - August 1984. I offer these personal obseravtions on the relative performances.

Statistically, the older issues contain far more items/volume than the issues published since you filled the breach as editor. Naturally, I loved them and they stand as a fitting memorial to Greg Wilson. They are both invaluable and irreplaceable. They were a wonder of their times. Don't be alarmed as there were far more individual items and occasional lengthy pieces but your editions are vastly superior in the quality and detail of subject matter.

The set out and logicical presentation of articles, in the post-August 1984 Rainbow, is fár easier to document than previous editions. For what it's worth I loved the old magazines but putting their contents into an index was a very arduous task. I found your setup far easier.

Your average length is longer and the subject matter is more comprehensively approached. Certainly the range of subject matter has evolved with changes in today's technology. The latest editions have a much better index set out and this innovation is welcomed.

I use the old issues as much as the new - hence the need for an index. I did not include GoCo/MiCo etc and the entries are exclusively for Rainbow. Items such as "ScoreBoard" and "PRINT#-2" are not included. In the case of the editorial which contain some meat. Also, the preamble to the CoCoConference is not noted but the summary of the events at the conference, published post - conference are cited.

Needless to say, I did not reference the ads but they are a fundamental resource in all Rainbows. In association with this index I have two others which complete my references. One for CoCoOz and another for CoCo.

I would still like to see some hacking articles, for instance an on/off light for the Mulit-pak interface. Similarly, I am fond of the disk tutorials, and disk utility programs you publish. With CoCoOz I think it would be worthwhile including some of the more important utility programs published in Rainbow. These are on Rainbow on Tape but one program a month should not upset anyone. The programs I have in mind as suitable are:-

Edtasm to disk - a patch program

Micropainter to disk - a patch program

Datafile reader - displaying data from files to screen for identification - etc.

During my travels through the various articles I came across the following - "We Mean Business" - Page 13, August 1985 - Issue 50 which refers to a Telewriter and Pro-Color file user group. Information from Patrick Simonis led me to believe that a Mr. Geoff Tulpott ran one such group with an annual fee and news letters.

Unfortunately, he has proved very difficult to nail down. Brizbiz - Mr Brian Bere-Streeter is into more "serious applications" for Pro-Color file.

I use Pro-Color file for:-

1. All my personal and business accounting
2. A paphiopedium species (slipper orchids) database.
3. Time line exercises.
4. Rainbow index
5. CoCoOz index
6. Hexadecimal to decimal chart
7. ASCII chart

etc.

If there is a club which operates the more simple applications of these programs I would certainly be interested.

Mick Gooch
Lowood. Qld.

*

Dear Graham,

We tried out a program by Fred Scerbo from May 1986 issue of Rainbow, ie Title Maker. My nine year old (Richard) made a title page for one of his picture files. We ANALYSED it, to tape. It SKIPFED alright. It wouldn't CLOAD. We discovered a new type of ERROR - DS. On LISTing we got ...

```
10 FORI=1TO
```

... and that was all.

When I changed the value of DV in Line 10 to -2, ANALYSE worked fine to the printer. I tried sending it to the screen; there was nothing wrong with the commands ... but of course as it printed to the screen it began analysing what it was printing .. I guess that it works to disk.

It does not work with tape. However, I am pleased that with some changes (and a lot of time) I did get it to work!

I am sending a copy of the lines from 731 on. The changes are in them. I couldn't get it to 'lose' the extra spaces, or the zeros, in the DATA lines, but at least I got it to work!

The above is the edited version of a letter I've written to Mr Scerbo.

CHANGES NEEDED FOR TITLEMAKER

```
731 S=L-1024+32
732 AA$="10 CLS0:FORI=1TO"
733 S$=STR$(S)
734 AB$=":READ A:PRINTCHR$(A+1
28);:NEXT"
735 SS$=AA$+S$+AB$
740 OPEN"O",#DV,F$
745 PRINT#DV,SS$
765 LN=10:FORN=1024TO L STEP32
770 LN=LN+10
771 V$=STR$(LN):V=LEN(V$):V$=R
IGHT$(V$,QV-1)+"    DATA"
775 FOR M=N TO N+31
777 RS=PEEK(M)
785 RS=RS-128
787 RS$=RS$+ST$(RS)
792 IF M<>N+31THEN RS$=RS$+","
793 NEXT M
795 RS$=V$+RS$
796 PRINT#DV,RS$
800 RS$="":NEXTN
805 PRINT#DV,"1000 GOTO 1000
810 PRINT@480,STRING$(31,32);
815 CLOSE#DV
820 GOTO530
```

Johanna VAGG

*

Dear Graham,

Yep, still alive. Sorry I haven't written for such a while. Found theological College a big bite to chew.

Well enough of that, I hope all is well you and yours. The mags are getting better all the time. Much too distracting, though. My wife nearly has to hide them so I'll do some work.

I thought I might share a little discovery (even if I'm not the first to find it).

I was getting pretty frustrated that CoCoMax would only print in the small mode eight spaces from the left and I couldn't find a way of fixing it. By getting the old CoCo to do a bit of navel gazing I found the allusive printer commands:

	Memory Location	Data	Function
All	32483	27	Escape
decimal	32484	108	Sets left margin
values			
	32485	8	Margin width
	32332	8	Advances head 8x by sending SO code to printer, ie val. 20

By simply pressing "QUIT" in the program and poking a new value in location 32485 you can move your printed drawing from left to right on your page. Poking the same value in 32332 will ensure the correct left margin on the second pass when using the Double Print option. I use the Epson RX-80 so these values may change for different printers.

I developed the Hebrew alphabet on CoCoMax, if anyone wants a copy, and also have created a Greek tutorial program (requires disk). The latter mainly for memorizing paradigms.

Hope the sun keeps shining your way!

Tom Stuart
Kenthurst, NSW.

Tom,

Good to hear from you, I'm glad you've solved all those horrible problems you had earlier with your printers!

Graham.

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

STOP GLARE !

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

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

BLAXLAND COMPUTER SERVICES PTY. LTD.

(047) 39-3903

COCO & 1000 SPECIALISTS

Special prices on a range of different printer papers & mailing labels.

Special prices apply during December & January!!

Hurry!! Some prices rise in January!!

Send, phone or check Viatel *64263# for our FREE "Last Minute" Christmas Catalogue!!

76A MURPHY ST. BLAXLAND 2774

USA PRODUCT REVIEWS

Hardware Review

TELEMAT ANTI–GLARE SPRAY

Many apologies .....

... we thought we had put it all in. Last month we presented a review for "Telemat Anti-Glare Spray" and we never completed it.

Here is the rest of that review. Mind you, it is a totally Australian product, so don't be led off by the title.

Technical Information dept.

```
Classification: "Hardware Modification"
          Cost: Depends on the screen type
                $42 for computer and monitor screens.
                $35 for a standard & domesticated TV
      Supplier: Telemat Pty Ltd
                3 Fortrill Drive
                Springwood, Qld 4127
```

"What is it?" dept.

"Telemat Anti-Glare sprays have been developed in Australia to combat the reflection and glare problems associated with computer and television screens."

Quote from the booklet.

"Storytime" dept.

One day the representative for Telemat appeared at the door saying he had a spray that you applied to your monitor or TV set to stop the glare. We invited him in and asked him a few questions. They were:

1. Why do we need it? What's the use?

The major advantage in using this product are that it will "virtually eliminate reflection on the screen from windows, lights etc", reduce eyestrain and headaches caused by reflection, give better contrast in a bright room and generally give a clearer image."

2. Are you going to take the monitor away? If you are, for how long?

"No, it won't be taken away; it's done right where your monitor sits."

3. Alright, so how long are you going to be?

"About 10 minutes."

4. What about keeping it clean?

"No worries. Clean it with Anti-Static polish once a week. Each can of antistatic costs $12."

5. Ok, so we'll let you apply the spray to one of our monitors. What if we decide we don't like it?

"You can take it off quickly and easily, anytime."

"Problems" dept.

* You DO lose a bit of resolution both the monitor and TV. The text appears to look a little fuzzy.

* When they applied the spray to the monitor, we had to evacuate the room. The spray had a definite pungent odour about it and we couldn't return to the room for about half hour. If you decide to spray your TV/monitor, we recommend you do that outside.

* For the price, it is expensive. All it involves is some masking tape & paper to cover the outside of the TV/Monitor and some transparency spray.

"General Comments" dept.

Overall, (in my opinion) it is a luxury. For the average user, it is not a necessitity.

The spray is really only designed for those who have a serious reflection problem. But then that can be remedied by moving the position of your TV/Monitor.

If, on the other hand, you are absolutely drenched in light and have moved the location of your TV/Monitor many times and still can see the reflection of the back window, then this spray is worth it. The reflection becomes a fuzz and you can actually see your screen instead of something else.

"Rating" dept.

If you REALLY could use it, then I would give it a mark somewhere between 6 and 10, depending on how bad your area is and how effective the spray is.

by Alex

Software Review

Darkmoor Hold: A Valuable Software Library Addition

Wandering through the dimly-lit corridors of a mysterious castle in search of adventure and riches is a daydream many of us romantics share. Until the CoCo came along, dreaming and reading were the extent of my fantasies. Now with software such as *Darkmoor Hold*, by Prickly-Pear, I can finally wander to my heart's content.

Darkmoor Hold is a graphics Adventure/Simulation that puts you, a human mercenary, in charge of a trio of adventurers hired by the ruler of a troubled kingdom. It seems an evil wizard is killing the king's beloved taxpayers and he would like you to remedy the situation. The wizard's castle, Darkmoor Hold, is comprised of 10 levels, each consisting of 20-plus rooms. You must survive all 10 levels while gathering better armament and treasure on the way. Since I was only able to get to level six, I can't testify as to what happens when you finally reach the wizard on the 10th level.

Darkmoor Hold has a different format than most Adventures on the market. In fact, it may be closer to a fantasy Simulation than an Adventure. The screen has the appearance of an ancient parchment scroll and is split into several parts. The top one-third shows a 3-D graphics representation of the rooms and corridors through which you wander. The bottom two-thirds is divided into three columns, allowing you to enter commands for each of the three adventurers. You, the human, a small but powerful Dwarf and a magical Elf make up the trio. The commands you can enter are predefined and consist of just under 20 choices. Examples are directions, search to find objects, fight to defend yourself and inventory. As my combat experience proved, 80 percent of the time you will find yourself entering fight. To call this game a slugfest would be putting it mildly. While I'm sure there are strategies to be developed, most of the time it's hard enough just staying alive.

Traveling through the various levels you face creatures of increasing power. On each level all the creatures look the same and only the Elf can correctly identify them. The graphics for the rooms and creatures are all very well-done and add considerably to the enjoyment of the game. In addition, for those of you who find spare time hard to come by, the game has a SAVE and LOAD feature allowing you to explore a little at a time.

The program is not copy protected, has a guaranteed free replacement for as long as you own it and is supported by some of the most considerate people I have ever done business with. There are many good companies selling CoCo software today and I can testify, without hesitation, that Prickly-Pear is one of the leaders.

I liked the program and would recommend it to any fantasy buff. I do feel, however, that the more experienced adventurers and role players out there might find the challenge a little too limited for their tastes. On the other hand, inexperienced adventurers who want a sample of Dungeon exploring, would do well to consider this program for their library.

(Prickly-Pear Software, 2640 N. Conestoga Ave., Tucson, AZ 85749, 64K ECB and one disk drive required, $29.95)

— Ken Boyle

Software Review

Develop Concentration With *The Memory Game*

As you probably know, it is hard to find a good software package these days that has some educational value in it. Well, I believe that the people at Mikaron Software have finally bridged the gap between fun and education.

The Memory Game is a 64K, Extended Color BASIC program requiring disk drive. The program comes with *The Memory Game*, Puzzle Disk One (stored on the same disk as *The Memory Game*) and a small instruction card. Although the card is small, it contains some very good information and an example of game play. Loading instructions for the game are on the disk.

The Memory Game is played like a game of Concentration. For those of you who are not familiar with it, I will explain. The screen is divided into 30 boxes numbered one through 30. Behind each box is a point value, which remains constant throughout the game. Each turn, you pick two boxes, revealing their point values. If they match, you gain the points, and the boxes they occupied are filled with parts of the main puzzle. If they don't match, the boxes are replaced and the next turn begins.

The main puzzle is a collection of symbols, objects and letters that represent a phrase or saying when put together. The disk has 10 puzzles on it, so you should have fun for awhile. Mikaron Software says they will have more puzzle disks soon.

I found *The Memory Game* to be very enjoyable and fun to play, but I do have one complaint: The speed of the game is extremely slow. Even though I don't look at this as a plus for the game, some people might. The longer there is between turns, the longer you have to remember the positions of the point values, thereby increasing the effectiveness of this program.

The game keeps a running high score, so you can see how well you have been playing. If you would like to play against a friend, the game has a two-player mode for added fun and competition.

Overall, *The Memory Game* is a great value for its price. If you want a game that is fun as well as educational, then *The Memory Game* is right up your alley.

(Mikaron Software Company, P.O. Box 1064, Chester, CA 96020-1064, $9.95)

— Sean McDonough

Software Review

Pinball Factory Rings Up Points

I've never been a big fan of computer pinball — until now. I've tried practically every CoCo pinball game on the market, and while most of them have been good, after a few games I would invariably become bored. After all, these games only offer the choice of a few playing fields, and even the most diehard pinball fan is sure to soon tire of playing the same game over and over and over.

Enter *Pinball Factory* from MichTron, a terrific new game which not only lets you play pinball on your CoCo, but lets you create your very own pinball game. *Pinball Factory* is similar to *Pinball Construction Set* on the Apple, and it puts new life into a very old game.

Pinball Factory comes on disk and is accompanied by a set of easy-to-read, complete instructions. I was glad to see that the disk was not copy protected and was accompanied by instructions on making a backup disk for your own use. Loading the game is as simple as putting the disk in the drive and typing RUN "PINBALL". After the title page appears, you have the option of playing the current game, loading a new game from disk (eight sample games come on the disk), modifying the current game or creating a new one, saving your creation to disk, or taking a directory of the pinball files on the disk. The program requires only one drive, but can use a second drive if you have one.

Load a game by moving the on-screen pointer with keyboard, mouse or joystick to the section of the screen marked LOAD and press ENTER (or the joystick button) then point to the file you want to load and press ENTER again. There is no need to type in the filename from the keyboard. Because the pinball data files are stored in a special format, they do not appear on the disk directory, and the manual warns not to save any other files on your pinball diskette.

Once loaded, selecting Play from the screen lets you play the current pinball game. You then have control of the two flippers and can "tilt" the board in one of three directions (left, right and up) to make the ball go where you want it. The graphics are in black and white only, but are quite good nonetheless. The animation is good and, except for a few rare moments, flicker-free. Sound effects are adequate, but not diverse. Pause and Quit options are provided.

The area in which *Pinball Factory* shines is in the creation and modification of your own pinball games. You can control the placement of everything except the flippers; you can even design your own high-resolution graphics logo for your pinball game. You can place bumpers of varying types around the board. When the ball hits one of these, you score a certain number of points — it is up to you how many points each type of bumper is worth, as well as how many points it takes to earn extra balls — and your ball is deflected back at a fast speed. There are even "multirail" bumpers which provide bonus points if all rails are hit. *Pinball Factory* also allows you to place polygons anywhere on the board. These polygons act only as physical obstructions, yielding no points and hindering the balls. They may be used to change the shape of the playing board or increase the challenge. Once placed, any object on the board can be removed or changed, and the program provides a test option for trying out your creation. Up to 90 objects may be put on the playing field.

Once the playing field is set up, you can change the rules of the game. This includes setting the speed, the pull of gravity on the ball, the number of balls, and the elasticity (which controls how fast the ball will bounce off an obstruction). As I mentioned before, you can also change the scoring rules, thus making the games harder or easier.

Overall, this is an excellent game. Unlike past pinball games, *Pinball Factory* puts you in control of the entire pinball game, letting you tailor games to your particular liking. You can even design pinball games with friends and have a pinball marathon or competition. Because *Pinball Factory* does not lock you into a particular game, it kept my interest, as I'm sure it will yours. The only minor bug I found was that the selection (menu) pointer scrolls off the screen in one direction but not the other, which was sometimes annoying, but doesn't affect game design or play.

Suggestions? MichTron could have included more features to build a pinball game from, such as ramps, optional flippers, moving targets, etc., or a feature to let the user design his own bumpers or polygons. Letting the user create his own sound effects or background music would have put the icing on the cake, but *Pinball Factory* is a most enjoyable game as it now stands.

On a scale of one to five, I would rate *Pinball Factory* as follows: playability, 5; sound effects, 3; keeps interest, 4; price vs. value, 3; graphics, 4; speed/animation, 4; documentation, 4; and overall rating, 4.

As a final note, I was impressed by the trust MichTron puts in their customers. Not only was the game able to be backed up, but MichTron offers a guarantee of satisfaction and a 30-day warranty. If you even remotely like pinball, I think you can trust that you'll enjoy *Pinball Factory.*

(MichTron, 576 S. Telegraph, Pontiac, MI 48053, 313-344-5700, 64K disk required, $34.95)

— Eric Tilenius

Software Review

Wall Street Keeps the Interest Flowing

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

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

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

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

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

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

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

— Patricia Arrington

Software Review

Become a Hi-Res Hero With *Dragon Blade*

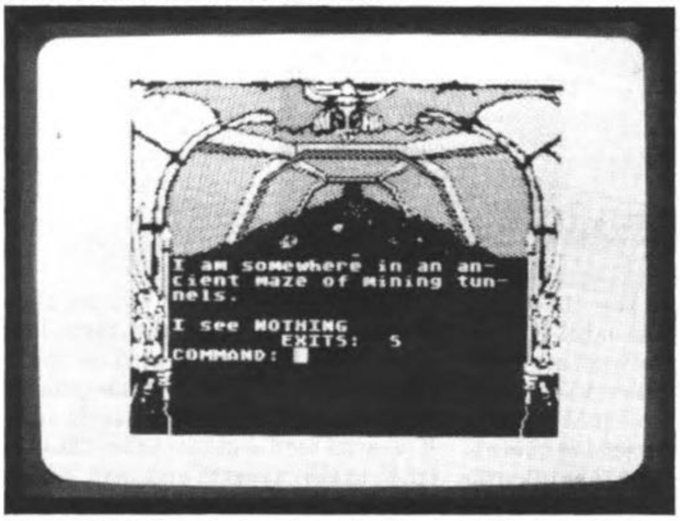
Those who like playing Adventure games are in for a treat. Prickly-Pear Software has created an animated, Hi-Res Adventure that sends you into the Middle Ages with style.

The object of the game is to find the Dragon Blade, which is the only thing that will kill the ancient dragon that has been terrorizing your small medieval village.

In its last attack, the dragon killed your father, the chief of the village. Now you must succeed him as chief, and somehow save the village from further attack.

You begin your quest in the Forest of Lore, soon discovering that the quiet countryside holds mystery and danger. As you search for the Blade, you encounter gargoyles, witches and a menacing guardian. You may fall into an abandoned mine shaft and find yourself wandering in a dark maze, or facing a monster who can turn you into stone with a single look. Despite these risks you must forge ahead, gathering the tools and knowledge that will allow you to slay the dragon and save your village.

Some Hi-Res Adventures are so slow you have to wait more than you play, but *Dragon Blade* is quick enough to keep the action rolling. The graphics are first-rate. (See Figure 1.) They are well-designed and do a good job setting the stage for a medieval quest. Several screens bring the Adventure to life with animation.



```
I am somewhere in an an-
cient maze of mining tun-
nels.
I see NOTHING
       EXITS:  S
COMMAND: █
```

Figure 1: Although the photo is shown in black and white, the actual Hi-Res game screen is in color.

Dragon Blade is definitely worth buying if you enjoy Adventure games — and it would make a great Christmas gift for your favorite CoCo nut.

(Prickly-Pear Software, 213 La Mirada, El Paso, TX 79932, 64K disk required, $29.95)

— Andy Dater

Software Review

CyberTank Won't Let You Go

To tell the truth, when I first realized I was to review a game, I was somewhat disappointed. It seems that most games I've run across become uninteresting and even annoying after several plays. If you are one of the many people who feel the same way, I think you will be in for a pleasant surprise with *CyberTank* from Mark Data Products.

CyberTank is a Simulation in which you find yourself in a futuristic tank besieged by Autonomous Land Vehicles (ALVs), which are actually four different types of enemy tanks with varied armaments and capabilities.

You, of course, are not left defenseless. Along with a non-renewable shield, you have many types of weapons at your disposal, including B-1 stealth plating, radar, cannon and rockets. However, you only have a limited supply and must use them judiciously lest you become destroyed before you can re-supply.

One of the things that makes the game interesting is that you must be, in effect, three people: the gunner, the loader and the commander. You have a separate screen for each one; when and how well you change screens can affect how long you stay alive.

The gunner's screen is where you will spend most of your time while in combat. (See Figure 1.) From here you aim and fire on the enemy. You have a small radar indicator, so you know from which direction you are being attacked. You also have fuel and shield gauges.

The loader's screen is where you decide what type of weapon and munitions to use and make them ready to use. You also access the supply depot from here.

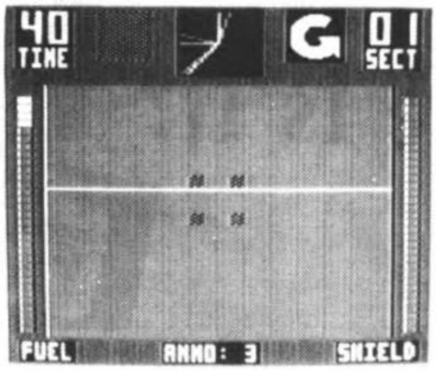


Figure 1: Although the photo is shown in black and white, the actual Hi-Res game screen is in color.

The last screen, the commander's screen, is a cumulative radar map of the sector you are in currently. (See Figure 2.) The effectiveness of your radar screen is determined by equipment used by the loader.
a small number of weapons to call upon. However, as you progress to different sectors via the teleport device you pick up at the munitions depot, you accumulate more and deadlier weapons.

The documentation mentions no ultimate goal other than to kill the enemy and stay alive, and I didn't find a way to finally win. But to be fair, I only got as far as Sector 7, so I don't know what happens in later sectors.

The program comes in a nice, black vinyl book, is on a copy-protected disk and has six pages of documentation. To use *CyberTank* you need 64K and one disk drive. Joysticks are not necessary.

The main complaint I have with *CyberTank* is the lack of understandable documents. The reference manual makes sense *after* you've figured out how to play the game, but doesn't help much the first few times you play.

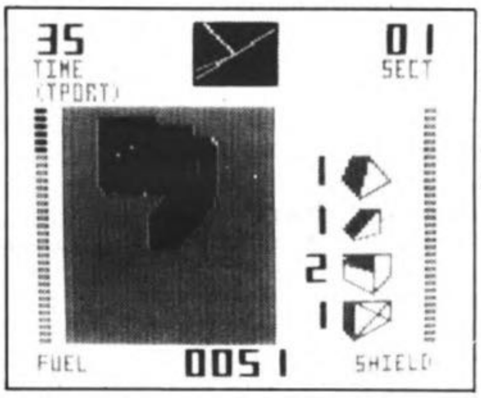


Figure 2: Although the photo is shown in black and white, the actual Hi-Res game screen is in color.

To sum up, *CyberTank* takes good strategy and quick hands to survive, and I've had many enjoyable late night marathons with it. Hopefully, the documentation will be re-written; in spite of that, I highly recommend *CyberTank*, especially to all of you who have become bored with games.

(Mark Data Products, 24001 Alicia Parkway, No. 207, Mission Viejo, CA 92691, disk only, $27.95.)

— Bill Tottingham

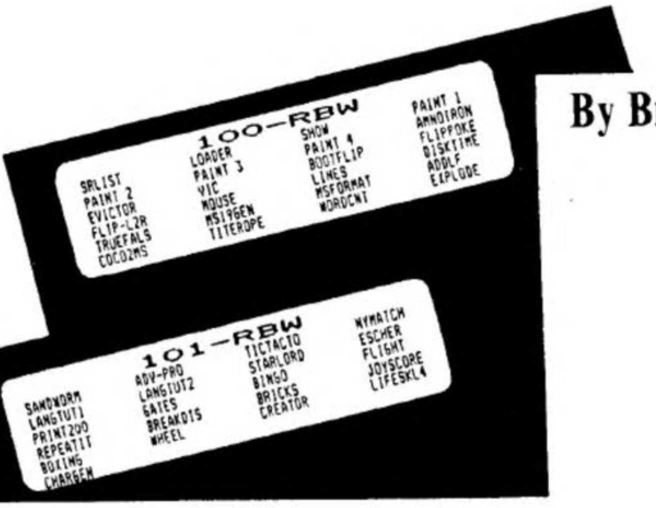
ORGANIZATION

16K
Disk

Display up to 24 program names on a neat disk label

By Brian Biggs

Where Is It?



Do you list the directory of disk after disk to find the program you're looking for? Do your disks sit in a box, out of order, or even worse, on the floor in a pile? If so, *Disk Labeler* can help.

Disk Labeler numbers your disks and prints up to 24 program names on 3½-by-$^{15}/_{16}$-inch, one-across mailing labels.

To get started, type in the listing, save a copy, and run the program. The title screen appears and you are reminded to set the printer to 9600 baud. To change the baud rate, simply poke a different value into Location 150 in Line 170 using the table in Figure 1.

Figure 1

600 baud	———	`POKE 150,87`
1200 baud	———	`POKE 150,41`
2400 baud	———	`POKE 150,18`
4800 baud	———	`POKE 150,7`

If the printer is not on, the program pauses until it is turned on, otherwise you see the main menu. The status section shows the current setup for printing. You can change disk number and disk code by pressing 1 and 2, respectively. The disk code can be any three-letter code that suits your needs.

Both the disk number and disk code must be three characters long. If they are less than or greater than three characters, *Disk Labeler* will automatically go back to the main menu. If you have already selected 1 or 2 from the menu and want to leave the contents the same, just press ENTER and you will go back to the main menu and the status line will not change.

Menu Option 3 lets you change the drive of the disk whose label you are printing. All drives (0 to 3) are available.

Menu Option 4 takes you to the program entry mode of *Disk Labeler*. You are prompted to enter the name of each program on the disk, one after the other. If you want to change menu options or abort the program entry mode, type R and press ENTER. You will be returned to the main menu.

Typing D and pressing ENTER gives you a directory of the disk. This function is used to get program names from the disk to enter them into *Disk Labeler*. Pressing any key after the directory is displayed returns you to program entry mode.

After all the programs on one disk have been entered, press P and ENTER. The screen will display the label with the program names printed in three columns. This three-column format must be used because of the size of the CoCo's screen, but the names will be printed in four columns on the label. If all the names are correctly spelled, press Y, and printing begins. Otherwise, press N and you are returned to the program entry mode.

After printing is over you are asked if you want to print the same label again. This function is used to line up the labels in the printer without having to type in all the program names again. If you do not want to print the same label again, type N and you will be returned to the main menu. To leave *Disk Labeler*, select main menu Option 5 and you can exit.

Disk Labeler is quite useful as written but, as is the case with most programs, it was written to be modified. One thing that could be added is a subroutine to alphabetize all programs entered, or a subroutine to read the programs directly from disk. I wrote *Disk Labeler* for a Star SG-10 printer. If you want to change the codes, an explanation of the printer codes I used, in the order they appear in the program, can be found in Figure 2.

Any modifications you have made or any questions you have can be sent to me at 3555 Rolling Hills Lane, Grove City, OH 43123. Or call (between 7 p.m. and 10 p.m.) (614) 878-1081. Please include an SASE if you want a response. □

Figure 2

`CHR$(27) "E"`	———	Emphasized print
`CHR$(27) "G"`	———	Double-strike print
`CHR$(27) "C" CHR$(6)`	———	Set page length to 6
`CHR$(14)`	———	One line expanded print
`CHR$(15)`	———	Condensed print
`CHR$(27) "A" CHR$(9)`	———	Set line feed to $^{9}/_{72}$ inch
`CHR$(13)`	———	Carriage return
`CHR$(12)`	———	Form feed
`CHR$(27) "@"`	———	Reset printer
`CHR$(7)`	———	Sound printer bell

continued to page 15

16K ECB

Mission: Hold the Bridge

By Charles Farris

During World War II, your paratrooper battalion is dropped behind enemy lines to capture several bridges along the Rhine River. Unfortunately, your battalion runs into a Panzer division at one bridge. After a short battle, your battalion retreats. In the confusion, you become separated from the battalion, finding yourself at the far end of the bridge. Looking back across the bridge, you see the German tanks start up to pursue your fleeing battalion. After finding an abandoned mortar hidden in the bushes, you prepare to meet the enemy. As the first tank rolls onto the bridge, you fire.

Mortar Command is a 16K ECB program requiring one joystick plugged into the right port. Control the angle of the mortar by moving the joystick horizontally. Moving the joystick to the right decreases the angle of the mortar while moving it to the left increases the angle. The maximum angle is 85 degrees and the minimum angle is 45 degrees. Pushing the button fires the mortar.

The tanks that roll across the bridge come in waves of six. The speed of the tank is controlled by its placement in the wave. The first tank moves very slowly, the second a little bit faster and so on until the last tank in the wave crosses, moving very fast. Once the sixth tank is destroyed, the next wave begins.

There are two ways to destroy the tanks. One is to hit them with your mortar shells. A hit anywhere on the tank destroys it. The second way is to let the tank roll over your ammo, which is in front of your mortar, and blow up. The only problem with this method is that you lose a life. Every time you destroy a tank with your mortar, a small swastika is drawn in the score box at the top of the screen. You only get a swastika if you hit the tank with a shell. Allowing the tank to run over your ammo does not count as a "kill" and you will not get a swastika. After destroying 36 tanks, a glider will fly down and land on the bridge, signaling that you have won the game. Also, for every 10 tanks you destroy, you get an extra man.

Upon running the game, you are asked if you want to use the high speed poke (POKE 65495,0). If your computer cannot take this poke, delete lines 6 through 9. After you have selected YES or NO, the title screen is drawn. Pushing the firebutton starts the game. After you lose all your men, the words GAME OVER are displayed and the triumphant German tanks roll across the bridge at high speed. Pushing the firebutton restarts the game. Pressing the space bar returns the computer to normal speed if you selected the high speed poke and ends the program. Be sure when you type in the program that the poke in Line 365 is POKE 65494,0 and not POKE 65495,0.

Line changes for playing the game on the keyboard are listed at the end of the article. The use of the high speed poke can be chosen at the beginning of the game. The game is best played on a monitor, since the artifact colors in PMODE 4,1:SCREEN 1,1 make the ground look striped. To fix this, change the command SCREEN 1,1 in Line 175 to SCREEN 1,0. The ground will still look striped, but it will look better than before. On a monitor, the ground is not striped, but black speckled with white dots.

The program uses a sine wave to plot the course of the shell. The routine for that is located in lines 115 through 160 and accessed from Line 60. The angle of the mortar is controlled from lines 55 through 75. The tanks are controlled from lines 90 through 110. The tanks' coordinates are defined in lines 190 through 215. The background is drawn in lines 285 through 340. The score box and scoring is done in lines 385 through 430. The routine for determining whether or not a tank is hit by a shell is found in lines 250 through 255, and the tank is destroyed in lines 260 through 280. The other parts of the program not mentioned here are labeled with remark statements for easy access.

Most of the graphics are done with DRAW statements, such as the tanks, glider, score box, swastikas, and the lettering in the title screen, game-over sign, and end-of-the-game graphics. The bridge was done using the LINE command, the river with CIRCLE commands, the banks with the PRESET command.

The hardest part of the program to develop was explosions. I put the tank array that I defined with GET in Line 185 onto a smaller area. The computer, confronted with this problem, mashes the picture in order to fit it into the smaller area. By using a FOR-NEXT loop to slowly decrease the size of the PUT area, the tank seems to blow apart while in reality, it is being squashed together. An interesting explosion happens when the computer only decreases the PUT area by one pixel. The tank appears to be blown off its treads.

When I wrote the bridge graphics, I did not know that you can use the bridge pylons as sights. The mortar shells will either land on the pylons or in the blank spaces on the bridge between them. The last two tanks in every wave move extremely fast and you will be able to get only one shot in, so wait till they get on the bridge, then shoot straight up. The shot should catch them just before they roll over you. Experiment to find the right setting and time to fire.

Change the M=M-1 in Line 95 to M=M-0 for more lives.

For those without joysticks, I have included a list of line changes for converting the game to keyboard control. They are:

```
55  A$=INKEY$
60  IF A$=CHR$(32) THEN MF=1
65  IF A$=CHR$(8) THEN LINE(0,170)
    -(C,D),PSET:C=C-1:IF C<1
    THEN C=1
70  IF A$=CHR$(9) THEN LINE(0,170)
    -(C,D),PSET:C=C+1:IF C>10
    THEN C=10
245 IF INKEY$=CHR$(32) THEN
    PCLS5:GOTO 285 ELSE GOTO 240
360 IF INKEY$=CHR$(32) THEN
    RE=0:K=0:GOTO 20
365 IF INKEY$="E" THEN POKE
    65494,0:STOP
550 IF INKEY$=CHR$(32) THEN 9
    ELSE 550
```

These changes will enable you to control the angle of the mortar with the right- and left-arrow keys. The space bar becomes the firebutton. When the GAME OVER sign appears, the E key stops the game and returns the speed to normal. The space performs as the firebutton for all other functions.

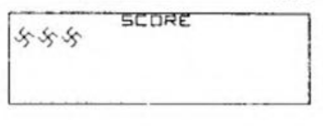
If you have any questions concerning the game, send an SASE with your questions to me at P.O. Box 582, 1141 USAFSAS Det 2, APO, NY 09011. If you live in Europe, send the SASE to Eindstraat 15, 6451 AA Schinveld, the Netherlands.

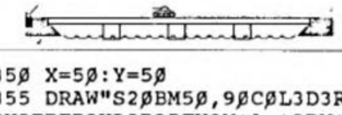
Have fun and good luck! □

65	126
160	251
235	81
295	91
370	143
440	211
480	164
END	0

The listing: MORTAR

```
Ø REM ******************
1 REM * MORTAR COMMAND *
2 REM *       BY       *
3 REM * CHARLES FARRIS *
4 REM ******************
5 ' HIGH SPEED POKE (Y/N)
6 CLSØ:PRINT@33*8,"high"+CHR$(12
8)+"speed"+CHR$(128)+"poke";
7 PRINT@3Ø2,"y"+CHR$(128)+"n";
8 IF INKEY$="Y" THEN POKE 65495,
Ø:GOTO 9 ELSE IF INKEY$="N" THEN
 POKE 65494,Ø:GOTO 9 ELSE GOTO 8
9 REM
1Ø SS=1
15 DIM T(43),TX(1Ø),TR(78)
2Ø GOTO 17Ø
25 RE=Ø:C=1Ø:D=16Ø:TY=157:WT=1:M
=3:TS=1:K=Ø:L=Ø
3Ø ' DRAW MEN LEFT
35 IF M=Ø THEN C=Ø:GOTO 35Ø ELSE
 FOR G=1 TO M:L=L+1Ø:DRAW"S4BM"+
STR$(L)+",191CØURE2G2RDL2":NEXT
G:L=Ø
4Ø DRAW"C5"
45 ' MAIN PROGRAM
5Ø A=JOYSTK(Ø):B=JOYSTK(1)
55 IF PEEK(6528Ø)=126 OR PEEK(65
28Ø)=254 THEN MF=1
6Ø IF A=Ø THEN  LINE(Ø,17Ø)-(C,D
),PSET:C=C-1:IF C<1 THEN C=1
65 IF A=63 THEN LINE(Ø,17Ø)-(C,D
),PSET:C=C+1:IF C>1Ø THEN C=1Ø
7Ø MA=C*5Ø
75 COLOR 5,Ø:LINE(Ø,17Ø)-(C,D),P
RESET
8Ø CIRCLE(Ø,17Ø),5,Ø
85 ' TANK MOVEMENT
9Ø IF WT=6 THEN RE=1:WT=1:GOSUB1
9Ø
95 PUT(TX(WT),TY)-(TX(WT)+26,TY+
12),T,PSET
1ØØ TX(WT)=TX(WT)-TS:IF TX(WT)<=
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
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,5ØCØS4R7ER4FDGL4HU
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"CØS24BM3Ø,5ØU3R3D3L3U3BFDRU
LRBEBRD2FREU2LD2LU2LBR4R3DL2RDLR
2DL3U3R3BRR2FGFLHDLU3RBDRBR2BUR3
DL2R2D2L3UR2L2U2BR4R3DL2RDLR2DL3
U3R3BRBDD2RURDRU2HLGBRRBEBRR3DL2
R2D2L3UR2L2U2"
23Ø DRAW"BM3Ø,55R3DL2R2D2L3UR2L2
U2R3BRR3D3L3U3BFDRULBUBR3R3DL2RD
LDLU3R3BRR3DLD2LU2LUR3BRD3ERFU3L
DLULBR4BDD2RURDRU2HLGBRRBR2UR2FG
FLHDLU2BRRBUBR2R3DL2RDLR2DL3U3"
235 DRAW"S16BM8Ø,14ØR2DL2DU2BR3R
2DLFHLDU2BR3R2L2DRLDR2BRR2UL2UR2
 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
275 GOTO 35
28Ø ' DRAW RIVER,BANKS, & BRIDGE
285 Y=17Ø:L=15
29Ø L1=255-L
295 FOR X=1 TO L STEP 2:PSET(X,Y
,Ø):NEXT X:FOR X=L1 TO 255 STEP
2:L1=255-L:PSET(X,Y,Ø):NEXT X:Y=
Y+1:L=L+1:IF Y=186 THEN 3Ø5 ELSE
 FOR X=Ø TO L STEP 2:PSET(X,Y,Ø)
:NEXT X:FOR X=L1 TO 255 STEP 2:L
1=255-L:PSET(X,Y,Ø):NEXT X:Y=Y+1
:L=L+1
3ØØ IF Y=186 THEN 3Ø5 ELSE 295
3Ø5 LINE(Ø,17Ø)-(5,177),PRESET,B
F
31Ø FORB=38 TO 224 STEP 1Ø:CIRCL
E(B,185),6,Ø,.41,1,.5
315 NEXT B:O=Ø
32Ø LINE(19,17Ø)-(236,175),PRESE
T,B:LINE(19,17Ø)-(14,17Ø),PRESET
:LINE(236,17Ø)-(243,17Ø),PRESET
325 FOR X=45 TO 224 STEP 27:O=O+
1:IF O=2 OR O=4 OR O=6 THEN LINE
(X-7,175)-(X+7,188),PRESET,B:NEX
T X
33Ø NEXT X
335 GOSUB 385
34Ø X=Ø:GOTO 25
345 ' GAME OVER
```





```
35Ø X=5Ø:Y=5Ø
355 DRAW"S2ØBM5Ø,9ØCØL3D3R3ULBR2
DU2ERFD2UL3R3BFU3M+1,+2RM+1,-2D3
BRR3L3URLU2R3BR6R3D3L3U3R3BRM+1,
+3RM+1,-3BRR3L3D2RLDR3BRU3R3D2L2
FHL"
36Ø IF PEEK(6528Ø)=126 OR PEEK(6
528Ø)=254 THEN RE=Ø:K=Ø:GOTO 2Ø
365 IF INKEY$=CHR$(32) THEN POKE
 65494,Ø:STOP
37Ø IF SS=Ø THEN 46Ø
```

```
375 GOTO 36Ø
38Ø ' SCORE BOX GRAPHICS
385 IF SS=1 THEN 43Ø ELSE IF K=1
Ø OR K=2Ø OR K=3Ø THEN M=M+1:GOT
O 39Ø ELSE IF K=36 THEN 475
39Ø LINE(Ø,Ø)-(25Ø,75),PRESET,B:
DRAW"CØBM95,1Ø;S16R2UL2UR2BRR2L2
D2R2BRR2U2L2D2R2BRU2R2DL2RFBRR2L
2URLUR2"
395 F=5:LV=2Ø
4ØØ FOR KI=1 TO K:IF F>24Ø THEN
F=5:LV=LV+2Ø
4Ø5 IF K=Ø THEN RETURN
41Ø DRAW"S8CØBM"+STR$(F)+","+STR
$(LV)+"F2E4F2BL4F2G2BU8G2F2"
415 F=F+2Ø
42Ø NEXT KI
425 RETURN
43Ø ' TITLE SCREEN
435 DRAW"S24CØBM22,5ØU5RFRERD5LU
4GLHD4LBR7R3EU3HL3GD3FBU2UERFDGL
HBM+5,+2U5R4FDGLFRDLH2LD2LBU3BRR
3UL3DBF3BR4RLU4L2UR5DL2D4BR3U3E2
RF2D3LUL3DLRBU2UERFDL3R3BRBD2BRU
5R4FDGLFRDLH2LD2LBU3BRR3UL3D"
44Ø DRAW"BM4,9ØBRR3L3HU3ER3FDLUL
2GDFR2URDGBR3R3EU3HL3GD3FBEREUHL
GDFBR4BDU5RFRERD5LU4GLHD4LBR6U5R
FRERD5LU4GLHD4LBR6U3E2RF2D3LUL3D
LRBU2UERFDL3BR5BD2U5RF3U3RD5LUH3
D4LBR6U5R3F2DG2L3RBUU3R2FDGL2"
445 DRAW"BM8Ø,12ØS16U2FED2BRR2LU
2LR2BRR2L2DR2DL2BR3R2UL2UR2BRR2L
D2LR2BRR2U2L2D2R2BRU2F2U2BRBDR2
45Ø DRAW"BM33,14ØU2DR2UD2BRR2U2L
2D2R2BRU2D2R2BRREHLD2BR6U2LR2BRD
2UR2UD2BRR2L2URLUR2BR4BD2R2U2L2D
2R2ULRBFU2R2DL2RFBRR2LU2LR2BRRFG
LU2BR3R2L2D2R2ULRBDBRR2L2URLUR2C
5"
455 SS=Ø:TX(1)=255:TY=157
46Ø DRAW"C5":TX(1)=TX(1)-6:IF TX
(1)<=Ø THEN LINE(Ø,TY)-(TX(1)+26
,TY+12),PSET,BF:GOTO 455
465 PUT(TX(1),TY)-(TX(1)+26,TY+1
2),T,PSET
47Ø GOTO 36Ø
475 ' GLIDER DRAWN AND DEFINED
48Ø DRAW"S12CØBM128,96E2R3M+4,+1
M+1,-2RD2GM-5,+1L3M-2,-1R2U2DLDB
R2ERM+2,+1BR3BUR2L2BDBL3L4C5"
485 GET(128,85)-(17Ø,99),TR,G
49Ø LINE(128,85)-(17Ø,99),PSET,B
F
495 X=23Ø:Y=96
5ØØ ' LAND GLIDER ON BRIDGE
5Ø5 IF Y=155 THEN 525 ELSE 51Ø
51Ø X=X-3:Y=Y+1
515 PUT(X,Y)-(X+42,Y+14),TR,PSET
52Ø GOTO 5ØØ
525 X=X-1:PUT(X,Y)-(X+42,Y+14),T
R,PSET:IF X=1Ø THEN 54Ø
53Ø GOTO 525
535 ' END OF GAME GRAPHICS
54Ø DRAW"BM6Ø,12ØS2ØCØU2R2L2D2R2
ULRDBRU2R2D2L2R2BRU2R2D2L2R2BRU2
RFGLBR5U2D2EFU2BRR2D2L2U2R2BRD2U
2R2DL2RFBRU2DM+2,-1BD2M-2,-1"
545 DRAW"BM25,14ØU2FED2BRR2U2L2D
2R2BRU2R2DL2RFBR2U2LR2BRBDDUEFDU
L2R2BRBDU2R2DL2RFBR4R2L2U2R2BRR2
D2L2U2R2BRD2U2FED2BRU2FED2BRUEFD
UL2R2BDBRU2F2U2BRRFGLU2"
55Ø IF PEEK(6528Ø)=126 OR PEEK(6
528Ø)=254 THEN 9 ELSE 55Ø
```

Where Is It?

continued from page 12

```
200 ......195
320 ......196
500 ......245
690 .......43
END .....162
```

The listing: LABELER

```
1Ø CLEAR1ØØØ
2Ø CLS:FOR P=1157 TO 1179:READ A
:POKE P,A:NEXT
3Ø FOR P=1189 TO 12Ø5:READ A:POK
E P,A:NEXT
4Ø FOR P=1289 TO 13Ø1:READ A:POK
E P,A:NEXT
5Ø FOR P=1327 TO 133Ø:READ A:POK
E P,A:NEXT
6Ø FOR P=1413 TO 1435:READ A:POK
E P,A:NEXT
7Ø FOR P=1455 TO 1458:READ A:POK
E P,A:NEXT
8Ø DIM G$(24)
9Ø FOR X=1 TO 1ØØØ:NEXT
1ØØ CLS:FOR X=1 TO 1Ø
11Ø PRINT@225,"SET 96ØØ BAUD RAT
E ON PRINTER"
12Ø FOR P=1 TO 5Ø:NEXT
13Ø PRINT@225,"set 96ØØ baud rat
e on printer"
14Ø FOR P=1 TO 5Ø:NEXT
15Ø NEXT X
16Ø GOSUB 8ØØ
17Ø POKE 15Ø,1:'SET BAUD RATE TO
 96ØØ
18Ø N1$="123":N2$="ABC"
19Ø CLS:PRINT STRING$(32,"=");
2ØØ PRINT"status: DISK NUMBER
  ";N1$:PRINT"          DISK CODE
     ";N2$:PRINT"          CURREN
T DRIVE  ";D:PRINT STRING$(32,"=
");
21Ø PRINT@224,"1)CHANGE DISK NUM
BER":PRINT"2)CHANGE DISK CODE":P
RINT"3)CHANGE DRIVE":PRINT"4)ENT
ER PROGRAMS":PRINT"5)QUIT"
22Ø A$=INKEY$:IF A$="" THEN 22Ø
23Ø IF A$="1" THEN GOSUB 83Ø:PRI
NT@423,;:INPUT"NEW DISK NUMBER";
B$:IF LEN(B$)<>3 OR B$="" THEN 1
9Ø ELSE N1$=B$:GOTO 19Ø
24Ø IF A$="2" THEN GOSUB 83Ø:PRI
NT@423,;:INPUT"NEW DISK CODE";B$
:IF LEN(B$)<>3 OR B$="" THEN 19Ø
 ELSE N2$=B$:GOTO 19Ø
25Ø IF A$="3" THEN PRINT@423,;:I
NPUT"NEW DRIVE";D:IF D<Ø OR D>3
THEN 25Ø ELSE 19Ø
26Ø IF A$="4" THEN GOTO 33Ø
27Ø IF A$="5" THEN PRINT@423,"AR
E YOU SURE (Y/N)?":GOTO 29Ø
28Ø GOTO 22Ø
29Ø A$=INKEY$:IF A$="" THEN 29Ø
3ØØ IF A$="N" THEN 19Ø
31Ø IF A$="Y" THEN CLS:PRINT"THA
NK YOU FOR USING THIS PROGRAM":E
ND
32Ø GOTO 29Ø
33Ø FOR N=1 TO 24
34Ø CLS:PRINT@12,N1$;"-";N2$
35Ø PRINT@37,"ENTER THE PROGRAM
NAMES":PRINT
36Ø PRINT@97,"<P>RINT <R>ESTART
<D>IRECTORY":PRINT
37Ø PRINT" PROGRAM #";N;:INPUT G
$(N)
38Ø IF G$(N)="P" THEN 43Ø
39Ø IF G$(N)="R" THEN GOSUB 84Ø:
GOTO 19Ø
4ØØ IF G$(N)="D" THEN 73Ø
41Ø IF LEN(G$(N))>8 THEN PRINT"T
OO LONG (LIMIT:8 CHARACTERS)":GO
TO 37Ø
42Ø NEXT N
43Ø CLS:PRINT@12,N1$;"-";N2$:PRI
NT:PRINT
44Ø T=65
45Ø FOR Q=1 TO N-1
46Ø PRINT@T,G$(Q):T=T+11
47Ø IF Q=3 OR Q=6 OR Q=9 OR Q=12
 OR Q=15 OR Q=18 OR Q=21 OR Q=24
 THEN T=T-1
48Ø NEXT Q
49Ø PRINT:PRINT
5ØØ PRINTTAB(5)"IS THIS OK (Y/N)
?";
51Ø A$=INKEY$:IF A$="" THEN 51Ø
52Ø IF A$="Y" THEN 56Ø
53Ø IF A$="N" THEN 33Ø
54Ø GOTO 51Ø
55Ø 'START PRINTING
56Ø GOSUB 8ØØ
57Ø CLS:PRINT@236,"PRINTING"
58Ø PRINT#-2,CHR$(27)"E"CHR$(27)
"G"CHR$(27)"C"CHR$(6);
59Ø PRINT#-2,CHR$(14);"       "N1$
;"-";N2$;
6ØØ PRINT#-2,CHR$(15);CHR$(27)"A
"CHR$(9)
61Ø FOR Q=1 TO N-1
62Ø PRINT#-2,G$(Q),
63Ø IF Q/4=INT(Q/4) THEN PRINT#-
2,CHR$(13);
64Ø NEXT Q
65Ø PRINT#-2,CHR$(12);
66Ø PRINT#-2,CHR$(27)"@";
67Ø PRINT#-2,CHR$(7);
68Ø PRINT@225,"PRINT SAME LABEL
AGAIN (Y/N)?"
69Ø A$=INKEY$:IF A$="" THEN 69Ø
7ØØ IF A$="Y" THEN 56Ø
71Ø IF A$="N" THEN GOSUB 84Ø:GOT
O 19Ø
72Ø GOTO 69Ø
73Ø CLS:DIRD:PRINTTAB(9)FREE(D);
"GRANS FREE";:EXEC44539:GOTO 34Ø
74Ø DATA68,73,83,75,96,76,65,66,
69,76,69,82,96,1Ø4,8Ø,82,79,71,8
2,65,77,83,1Ø5
75Ø DATA7Ø,79,82,96,83,71,1Ø9,11
3,112,96,8Ø,82,73,78,84,69,82
76Ø DATA66,89,96,66,73,76,76,96,
83,69,77,8Ø,7Ø
77Ø DATA113,121,12Ø,117
78Ø DATA13,15,4,9,6,9,5,4,32,2,2
5,32,2,18,9,1,14,32,2,9,7,7,19
79Ø DATA49,57,56,54
8ØØ PE=PEEK(65314)AND1
81Ø IF PE=Ø THEN RETURN ELSE PRI
NT@486,"PRINTER NOT ON LINE!";
82Ø GOTO 8ØØ
83Ø PRINT@483,"MUST BE 3 CHARACT
ERS LONG";:RETURN
84Ø FOR X=1 TO 24:G$(X)="":NEXTX
:RETURN
```

One Character Space at a Time

By Joseph Kolar
Rainbow Contributing Editor

As the proud possessor of a working, original model, 4K expanded to 16K ECB, cassette-based CoCo, I am constantly struck with the simplicity, power and versatility of Mr. CoCo.

No matter how intriguing monster-sized keyboards are, how wide the color selections, how many function keys and other goodies to press may beckon, I have a tendency to return to the amazing CoCo. The keyboard is so clutter-free and businesslike that whatever perceived or imagined short-comings there may be, it is a joy for the recruit as well as the veteran CoCo nut to sit down and tickle the keys.

Just as a fledgling pilot learns to fly in a Piper Cub rather than a Lear Jet, so, too, will a newcomer learn best from a simple-to-manage computer, rather than an intelligence-insulting, icon-loaded moron machine with a zillion K memory.

In the last tutorial, we worked with LEFT$, RIGHT$ and MID$, displaying complete words or lines of print on the screen. The theme for today's lesson is one character/space at a time. Practically every bit of text will be displayed on the screen one character/space at a time.

This means we shall use a variable in every LEFT$, RIGHT$ and MID$ statement instead of a constant numeric value (the last value within the parentheses).

You may notice that I am very unimaginative and use the string variable, A$, repeatedly. CoCo has a tendency to search out and select the last instance of a variable in a listing at the point where it is operating, so this presents no problem. You may prefer to use a different variable in each instance.

Look at Listing 1, which is broken down into segments. Lines 0, 160, 340, 540 and 650 begin the five segments. The GOSUB routine at Line 1000 is used to flesh out the first segment. Since it isn't essential, it was relegated to the end of the listing.

Key in Listing 1 and run it. Not being too clever, I decided to print a heading with my daughter's name and address. But being egocentric, guess who's name I printed? Quick as a flash, it struck me that it was wrong. CoCo got cranky and showed me up with the flashing GOSUB 1000 routine. So, chastised, my name was peeled off, one letter at a time. However, I would not give up center stage. I insinuated my name at the center just like a ham and, having upstaged one and all, strode off the screen.

My daughter's name was printed at the heading, one letter at a time, but rolled up and off the screen. At this point, her husband, Jimy (sic), flashed his name on one letter at a time. However, she was indignant and push came to shove and his name was removed to be triumphantly replaced by her own, which was my original idea. Whew!

Now that you know the story, let's get a more detailed explanation. LEFT$, RIGHT$, MID$ and LEN were utilized to create all these interesting effects. List Line 100. It was decided to print the lines simultaneously, one letter at a time. The three lines were put into respective strings and assigned variables. To accomplish the goal, a loop, T, was created to loop enough times to place all three lines, including the longest, on the screen. LEN(A$) was used as a counter. However, C$, was the longest line. The number of characters/spaces in A$, since it was chosen as the counter, had to be greater than or equal to those in C$. This was done simply, but effectively, by adding enough blank spaces into the A$ string, so that it would be a bit longer than the length of C$.

PRINT@ locations were guesstimated on the first three screen rows, to be adjusted as required. LEFT$(A$,T) instructed CoCo to start from the left side of the string, A$, and print a letter, designated by T, at location 10. Then LEFT$(B$,T) told CoCo to put the first letter, '0', at 40, and LEFT$(C$,T) to put the 0th letter, 'S', at 70. A tiny pause followed and then CoCo put on the next trio of letters, where T equals one.

Rather than give you a possibly confusing burst of verbiage, edit Line 90 so 51 becomes 510 and run. Study this slowed down version carefully and rerun it until you can see what is what. Notice how efficiently CoCo tacked on each batch of characters.

In LEFT$(A$,T), T is an ever-increasing single digit number augmented by 1. Try adding STEP2 to Line 5, and run. A jerky presentation results when T equals two characters.

This is an unusual way to produce a three-line heading that you may want to save. A little pizazz goes a long way!

Note that Line 100 could have been zapped and Line 90 could have ended in Z,T. It is not as easy to see the outer loop, T. There is no law saying that you must compress or multiple-statement-line your program to death. Remember, when you run your program, you don't see the listing. You'll never see the debris, such as Line 105, if you forget to kill it.

For the purpose of instruction, the rule for this tutorial is, "one statement — one program line," except for the FOR/NEXT pause loops.

You will notice a lot of lines like Line 101, which are repeated in this listing and the next one. The programs really cry for GOSUB routines and, when you finalize your programs, you may prefer to make appropriate changes.

Restore Line 90 to 51 and drop the STEP2 from Line 5. List Line 105. We pause, have a short trip to the "wrong" GOSUB routine, and then return for another pause. Now delete Line 105. Run it to make sure it was a fossil.

Now, look at the routine in lines 1000 on. What we want to do is flash on "wrong" and blank it out 10 times with a small pause in each loop and then return to continue the tutorial.

My favorite name was replaced by wrong!. Notice that a few spaces were prefixed to wrong!. It was not necessary to suffix any because without an ending semicolon, the rest of the line would be blanked out by CoCo. The blanking line could just as easily begin at Location 10, rather than stingy 13, (Line 1020) to match Line 1009.

Make it a point to try all the minor alternates to see for yourself. Suppose you deleted up to NEXT in lines 1010 and 1030? What would Line 1040 look like? Not very stimulating. Better replace the removed segments of the two lines.

Coming back from the subroutine, we bump into another pause. The name reappears only to be picked off quickly, one letter at a time, beginning from the left side.

List lines 110 to 150. LEN(A$) was assigned a variable. To see what the value is, run it and press BREAK when the name is in process of being deleted from the screen. Type PRINT L and press ENTER. Now you know! The reason you must press BREAK while you are in the target area where L is being processed, is that the variable L is used later for other strings and you might easily pass over into one of those areas and pick up the value of the wrong L.

Itching to try out RIGHT$? Note lines 120 and 130, where everything appears to be backward. RIGHT$(A$,A) can be defined similarly to LEFT$(A$,A), where RIGHT$(A$,A) signifies that in the string A$, beginning from the right end, count A characters.

We expect to use RIGHT$ to put on the characters, A, one at a time (STEP-1). Note carefully, if we are using RIGHT$ in Line 120, 0 is the rightmost character, proceeding letter after letter STEP-1, until the leftmost character L is reached.

In effect, we are using RIGHT$ from left to right exactly opposite from the way it is ordinarily used to mimic LEFT$'s action.

In order to find our PRINT@ location for the first letter, 30-A (the last character is the first to be removed), the character is removed by the blank space, " ", and RIGHT$(A$,A) tells CoCo which A value is to be blanked out.

It might be wise to change 51 to 510 in Line 140 to see this operation proceed slowly. To visualize it more readily, temporarily add the line 132 PRINT@0,A;. Since the A$ string is in Line 20, it contains the extra blanks which are harmless and go unnoticed. Now run your work.

To make RIGHT$ pull it from the rightmost position to the leftmost, mask Line 130 with a REM. Add the line 131 PRINT@9," "+RIGHT$(A$,A). Run it and delete Line 132.

Did you note that Line 130 removed the name one letter at a time going from right to left, and Line 131 pulled the letters away through a single location, 9, one at a time?

Now, we can get silly. Mask lines 120 and 131. Unmask Line 130. Add 121 FOR A = TO L and run. This displays the line one letter at a time, last letter first, working leftward.

Finally, mask Line 130 and unmask Line 131. Can you guess what will happen? Run it and see.

The line was pushed backward, out of the hole at location 9. Change 510 back to 51 in Line 140. Forget about these last two cockeyed presentations — unmask lines 120 and 130 and either mask or delete lines 121 and 131.

List lines 210 to 330. In Line 210, we changed string A$ by looping off all the trailing blank spaces (Line 20). We assigned a variable, L, to the length of A$. In a loop, using LEFT$, we pulled the name out of a hole, Location 214, and dragged it leftward until it was completely displayed, pausing for a reasonable time lapse to evoke a smooth, banner-like motion.

Immediately, through the hole at Location 202, we stuffed it down by using RIGHT$ to maintain our leftward direction, letter by letter, until the entire name vanished down the rat hole.

You may want to return to this part of the program and pull it back out of the hole at Location 202, drag it rightward and bury it at Location 214. You can do it! An answer is given at the end of the tutorial.

List lines 400 to 535. A$ is a new string, which is placed on the screen using LEFT$, beginning at Location 8. T is incremented by +1, from 0 up to the value of the last letter of the string. After a pause, it places each succeeding letter in the next available space heading rightward. A long pause sets the completed name in place. Then, assigning L as the length of the string, from the last letter back to the first, the name is rolled up and off the screen using LEFT$ (Line 511).

If Line 510 is unmasked and Line 511 is masked, using RIGHT$, the name is removed by being pulled through the hole at Location 8.

List lines 600 to 640. The spot-usurping spouse, using the same string variable, A$, quickly slaps his name in the slot. It doesn't take long to kick him out of the area using LEFT$ by pecking away at his name going from right to left. Note Line 610. It is equivalent in form to lines 490 and 500 combined into one program line. If it were in two lines, it would read: 610 L=LEN(A$) and 611 FOR T=L TO 0 STEP-1.

Finally, triumphant, she put her name firmly in the header slot in the simplest manner by using the old faithful A$ as the string variable containing her name, counting the characters/spaces in her name, putting them into the FOR loop as T and using LEFT$, placing it on the screen, beginning at Location 7.

Here is a solution to the problem mentioned before. Add lines:

```
331 FOR A=0 TO L:PRINT@202,RIGHT$
(A$,A): FOR B=1 TO 200: NEXT B,A

332 FOR A=L TO 0 STEP-1:PRINT@
214-A," "LEFT$(A$,A): FOR B=1 TO
200: NEXTB,A
```

"Push your name in one direction and when it vanishes, pull out your mate's name from the rat hole and shoot it back across the screen."

If you have a problem, check lines 510 and 511 to see if the proper one is unmasked. To produce a faster, repeated push-pull banner, add 200 FOR C=1 TO 10 and add ,C to the end of Line 332.

In lines 250, 320, 331 and 332, change the pause value from 200 to 50.

Here's an idea — push your name in one direction and when it vanishes, pull out your mate's name from the rat hole and shoot it back across the screen. You get a nice domestic quarrel effect.

Note that in some instances, as in lines 130 and 332, a blanking space was required. Remove each and check out the sorry state of affairs.

Listing 2 is a homework assignment. It is one half of a demo program. Make sure you save it on tape. The other half will be a continuation of this listing. It is similar to Listing 1 but not quite. It is hoped that you crack open your notebook and analyze the listing. If you find any of the routines useful, you may want to put them into your Reference Notebook. □

Listing 1: HEADING

```
Ø 'LISTING1"
1Ø CLS
2Ø A$="JOSEPH KOLAR          "
3Ø B$="824 NE 56TH ST.
4Ø C$="SEATTLE, WASHINGTON"
5Ø FOR T= Ø TO LEN(A$)
6Ø PRINT@1Ø, LEFT$(A$,T)
7Ø PRINT@4Ø,LEFT$(B$,T)
8Ø PRINT@7Ø, LEFT$(C$,T)
9Ø FOR Z= 1 TO 51:NEXT
1ØØ NEXT
1Ø1 FOR Z= 1 TO 5ØØ:NEXT
1Ø2 GOSUB 1ØØØ
1Ø4 FOR X= 1 TO 5ØØ:NEXT
1Ø5 Z$=A$
11Ø L=LEN(A$)
12Ø FOR A=L TO Ø STEP-1
13Ø PRINT@3Ø-A," "RIGHT$(A$,A)
14Ø FOR B=1 TO 51:NEXT
15Ø NEXTA
16Ø '***
21Ø A$="JOSEPH KOLAR"
22Ø L=LEN(A$)
23Ø FOR A=Ø TO L
24Ø PRINT@214-A,LEFT$(A$,A)
25Ø FOR B=1 TO 2ØØ:NEXT
26Ø NEXT A
3ØØ FOR A=L TO Ø STEP-1
31Ø PRINT@2Ø2,RIGHT$(A$,A)
32Ø FOR B=1 TO 2ØØ:NEXT
33Ø NEXT A
34Ø '***
4ØØ A$="BETTY ANN WHITE"
41Ø FOR T= Ø TO LEN(A$)
42Ø PRINT@8,LEFT$(A$,T)
43Ø FOR Z= 1 TO 5Ø:NEXT
44Ø NEXT
45Ø FOR Z= 1 TO 5ØØ:NEXT
49Ø L=LEN(A$)
5ØØ FOR A=L TO Ø STEP-1
51Ø 'PRINT@8,RIGHT$(A$,A)
511 PRINT@8,LEFT$(A$,A)
52Ø FOR B=1 TO 2ØØ:NEXT
53Ø NEXT A
535 FOR Z= 1 TO 5ØØ:NEXT
54Ø '***
6ØØ A$="JIMY OWEN WHITE"
6Ø5 PRINT@8,A$
6Ø6 FOR Z= 1 TO 2ØØ:NEXT
61Ø FOR T= LEN(A$) TO Ø STEP-1
62Ø PRINT@8,LEFT$(A$,T)
63Ø FOR X= 1 TO 2ØØ:NEXT
64Ø NEXT T
65Ø '***
7ØØ A$="BETTY ANN WHITE"
71Ø FOR T=Ø TO 15
72Ø PRINT@8,LEFT$(A$,T)
73Ø FOR X=1 TO 2ØØ:NEXT
74Ø NEXT T
75Ø GOTO 75Ø
1ØØØ '
1ØØ5 FOR X= 1 TO 1Ø
1ØØ9 PRINT@1Ø,"   WRONG!"
1Ø1Ø FOR Z= 1 TO 2Ø:NEXT
1Ø2Ø PRINT@13,"         "
1Ø3Ø FOR Z= 1 TO 2Ø:NEXT
1Ø4Ø NEXTX
1Ø5Ø RETURN
```

Listing 2: HOMEWORK

```
Ø '<LISTING2>
5 CLEAR 5ØØ
1Ø CLS
2Ø A$="BETTY ANN WHITE"
3Ø B$="824 NE 56 ST.
4Ø C$="SEATTLE, WASHINGTON"
41 PRINT@8,LEFT$(A$,5);:FORZ=1TO
2ØØ:NEXTZ
42 PRINT MID$(A$,6,5);:FORZ=1 TO
2ØØ:NEXTZ
43 PRINTRIGHT$(A$,5)
44 FOR Z= 1 TO 5ØØ:NEXT
45 CLS
46 '***
47 A$="JOSEPH KOLAR          "
5Ø FOR T= Ø TO LEN(A$)
6Ø PRINT@9, LEFT$(A$,T)
7Ø PRINT@41,LEFT$(B$,T)
8Ø PRINT@7Ø, LEFT$(C$,T)
9Ø FOR Z= 1 TO 51:NEXT
1ØØ NEXT
1Ø1 FOR Z= 1 TO 5ØØ:NEXT
1Ø2 GOSUB 2ØØØ
1Ø3 PRINT@9,A$
1Ø4 FOR Z= 1 TO 5ØØ:NEXT
11Ø L=LEN(A$)
12Ø FOR A=L TO Ø STEP-1
13Ø PRINT@3Ø-A," "RIGHT$(A$,A)
14Ø FOR B=1 TO 51:NEXT
15Ø NEXTA
16Ø '***
21Ø A$="JOSEPH KOLAR"
22Ø L=LEN(A$)
23Ø FOR A=Ø TO L
24Ø PRINT@213-A," "+LEFT$(A$,A)
25Ø FOR B=1 TO 2ØØ:NEXT
26Ø NEXT A
27Ø FOR Z=1TO5ØØ:NEXT
3ØØ FOR A=L TO Ø STEP-1
31Ø PRINT@2Ø2,RIGHT$(A$,A)
32Ø FOR B=1 TO 2ØØ:NEXT
33Ø NEXT A
34Ø '***
4ØØ A$="BETTY ANN WHITE"
41Ø FOR T= Ø TO LEN(A$)
42Ø PRINT@8,LEFT$(A$,T)
43Ø FOR Z= 1 TO 5Ø:NEXT
44Ø NEXT
45Ø FOR Z= 1 TO 1ØØØ:NEXT
46Ø '***
49Ø L=LEN(A$)
5ØØ FOR A=L TO Ø STEP-1
51Ø 'PRINT@8,RIGHT$(A$,A)
511 PRINT@8,LEFT$(A$,A)
52Ø FOR Z=1 TO 2ØØ:NEXT
53Ø NEXT A
535 FOR Z= 1 TO 5ØØ:NEXT
54Ø '***
6ØØ A$="JIMY OWEN WHITE"
6Ø5 PRINT@8,A$
6Ø6 FOR Z= 1 TO 5ØØ:NEXT
61Ø 'FOR T= LEN(A$) TO Ø STEP-1
611 FOR T=Ø TO LEN(A$)
62Ø PRINT@8,LEFT$(A$,T)
63Ø FOR X= 1 TO 2ØØ:NEXT
64Ø NEXT T
65Ø FOR Z= 1 TO 5ØØ:NEXT
66Ø '***
7ØØ A$="BETTY ANN WHITE"
71Ø FOR T=Ø TO 15
72Ø PRINT@8,RIGHT$(A$,T)
73Ø FOR X=1 TO 2ØØ:NEXT
74Ø NEXT T
75Ø FOR Z= 1 TO 5ØØ:NEXT
8ØØ '***
81Ø B$="JIMY OWEN WHITE"
811 L=LEN(B$)
82Ø FOR A= Ø TO L
83Ø PRINT@22-A," "+LEFT$(B$,A)
84Ø FOR X= 1 TO 2ØØ:NEXT
85Ø NEXT A
86Ø FOR Z=1TO5ØØ:NEXT
2ØØØ '
2Ø1Ø FOR X= 1 TO 1Ø
2Ø2Ø PRINT@9,"    WRONG!"
2Ø3Ø FOR Z= 1 TO 2Ø:NEXT
2Ø4Ø PRINT@12,"          "
2Ø5Ø FOR Z= 1 TO 4Ø:NEXT
2Ø6Ø NEXTX
2Ø7Ø RETURN
```

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: $90.00

(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

PRINTER

by Jim Jacobs

THIS IS A PROGRAM to improve the printer control of Colour Basic so that programs can be listed satisfactorily. It provides for variable line and paragraph spacing, line width and margins. It is meant to be CLOADed and RUN before CLOADing the program to be printed. It is for the Dick Smith GP-80 printer but it is described in detail so that ir can be modified to suit other printers. It should run on any CoCo.

INTRODUCTION

Color Basic assumes the following about the printer:

1. 600 Baud.
2. Width 132.
3. Busy signal when it is not ready.
4. Automatic carriage return at the end of a line.
5. Serial data format, one start bit (zero), eight data bits (lsb first), two stop bits (ones) and no parity per character.

These did not all suit my GP-80 printer and also I wanted to improve the format of the printout. I imagine that others have similar problems and that an examination of my solution may help.

A serial to parallel converter from Geoff Fiala solved my first problem, interconnection.

Then by much checking the manual I worked out that the Busy signal was OK, a carriage return ($0Dh or 13dec.) was required at the end of a line and a position flag ($10h or 16) followed by the position in ascii decimal characters was required

By reading old RAINBOW articles and disassembling the ROM code I found that output from the COCO jumps to a vector in RAM at $0167h for each character then goes to the serial out routine at $A2BFh, if it is to be printed. The line character count ***

The description of the CLEAR command is wrong in the Colour Basic manual. The address at the end is the first protected address not the highest Basic can use, this is stored at $0027-8h.

PROGRAM

Must:

a. Control printed output only.
b. Be invisible to other programs.
c. Input line and paragraph spacing, line length and margin width.
d. Start each line at the margin.
e. Start a new line with the right spacing when the line character count exceeds the length.
f. Check each character to be printed for a carriage return and if so start a new line with the paragraph spacing.

"PRINTER" is a combination of Basic and machine language, see LISTING2. This is how it runs:

```
0 GOTO10
1 '*******  PRINTER ************
2 '***** JIM JACOBS ************
3 SAVE"93:3":END
10 GOSUB40:CLEAR99,A-81:GOSUB40:
FORA=A TOA+81:READB:POKEA,B:NEXT
:EXECA-82:CLEAR200,A-59
20 GOSUB40:PRINT@480,"PRINTER >>
":INPUT" PARA SPACE";B:POKEA+12,
B:INPUT" LINE SPACE";B:POKEA+53,
B:INPUT" LINE WIDTH";B:POKEA+25,
B:INPUT" MARGIN";B
30 POKEA+42,INT(B/10)+48:POKEA+4
6,B-INT(B/10)*10+48:EXECA+51:END
40 A=PEEK(39)*256+PEEK(40)+1:RET
URN
50 DATA142,1,103,166,132,167,140
,34,134,126,167,128,236,132,237,
140,26,51,140,3,239,132,57,52,6,
214,111,92,42,8,129,13,38,11,198
,0,141,17,167,228,53,6
60 DATA0,0,0,214,156,193,0,38,24
5,141,21,32,241,141,22,90,38,251
,134,16,141,15,134,0,141,11,134,
0,90,215,156,57,204,13,0,141,232
,126,162,191
```

. First high memory is protected so the machine language routines which are held as data can be poked out of harm.

. Then the output vector is altered so that each character can be examined as it is sent to the printer.

. Then the setup routine is dropped and the memory protection reset, releasing a few more bytes of memory.

. Next screen prompts appear for the parameters to be entered and poked into the right locations in the program.

. Finally the line routine is EXECuted, (the printer must be operating at this stage), to clear the printer buffer, set the first margin and see that all is OK.

Now "PRINTER" can be deleted or overwritten by another program which can be printed using the LLIST command. The machine language routines remain until the machine is switched off.

The parameters can be changed by ENTERing GOTO1 if the program still exists, otherwise lines which are copies of 1,2 and 3 will be required.

When printing the action is:

. The ROM print routine increments the counter for each character sent to the printer, this is not OK for the non printing margin codes so before the last one is sent the counter is set to -1 giving zero at the start of the line.

. If the counter is at maximum the character is held until the line feed routine is done.

. When a carriage return is found it is taken to be a paragraph marker and the required number of carriage returns are sent, this time the last margin code is sent in place of the original carriage return.

The program could be elaborated to form a print routine for a text processor by adding better line ending, page control and counting, etc., or the basis of a screen dump routine.

LISTING1 Printer Routines

The addresses shown are for a 16K machine. The program is automatically located by the Basic program.

continued to page 22

Formatting Text Presentations to Suit Yourself

By Joseph Kolar
Rainbow Contributing Editor

Ready for more punishment? In this tutorial, we shall finish the demo program we began the last time around. You were asked to do some homework. If you did it, give yourself an A.

Load in your homework assignment. Refer to Listing 1 and key in lines 900 to 1999. If you just arrived on the scene, or neglected to do your homework, key in the entire listing. Run it.

The result is quite attractive. It is an interesting, alternate way to place lots of text on a screen. It takes some effort but once you understand the system, you will find it easy to use this technique. The ultimate viewer will find it relaxing to read along without being asked for any hands-on input, such as press this or do that. In fact, I call it the read along gambit.

List lines 910 and 920. Variables A$ and B$ contain strings of text that fill the balance of the text panel. Creating the actual sentences is the most difficult task. You must compose sentences that say what you want said and yet fit into the constraints of the text page or panel.

Here are some factors to take into consideration. There are 219 characters/spaces in A$. The maximum that I could have produced in this particular instance is 245. You can check it out if you are in the right part of the program, without using any program line number, by asking CoCo to print L=LEN(A$).

Sometimes it is necessary to reword a sentence/paragraph to shorten it. At other times, it may be convenient to break it up into shorter bites. There are no hard and fast rules except the dictates of the situation. You must use the invisible vertical line gambit to format the text.

You must have a very good idea where you plan to locate the strings. You can do this by trial and error. If you do not feel comfortable with this method, feel free to try this next technique. Add the following lines:

```
930 PRINT@128,A$
931 PRINT@352,B$
932 GOTO932
```

Press CLEAR and type RUN900

Adjust the PRINT@ locations as necessary. Alter the contents/length of any sentence to give it a better fit. This is called your layout. After finalizing your layout, make a note of the A$ and B$ locations. Did you notice how abruptly the text was presented? Obviously, it will require a lot of time to read the text. (More on this phase later.)

Delete lines 930 to 932. Other problems and considerations may crop up and will be confronted as we encounter them. Now that we have a good idea of the contents and appearance of the text panel, we can use the read along gambit, using LEFT$ to create the routine that will work our will on CoCo.

List lines 940 to 1000. By trial and error, we determined that a time lapse of 50 units between the placement of each succeeding letter would afford us sufficient time to read the text as it is thrown onto the screen. (Feel free to use some other value, but be consistent throughout the program.)

Note in Line 940, T=O, etc., my typo error. Good-natured CoCo saved the day by reading it as T=0, etc.

Here is a fun thing! Temporarily insert 941 PRINTT; to prove that CoCo sees T=0, and type RUN900. Question: Why did CoCo print T up in the corner and then print T of the balance of A$, underneath?

Study the placement of this line and the solution will come to you. TRON is helpful. Delete Line 941 and type TROFF.

List lines 940 to 1000. A$ was put on first, using the location that you selected from your layout. It was done as explained in the last tutorial. B$ was put on next, also in a nested loop, where Line 1000 is equivalent to lines 960 and 970.

Refer to last month's tutorial if you are encountering difficulty recalling what the two similar routines do.

Could we mask Line 970 and add :NEXT at the end of Line 960? Take time out to see what CoCo does.

List lines 940 to 1005. Add :GOTO900 at the end of Line 100, and add 1001 GOTO 1001 and 1010 GOTO1010 in order to bypass part of the program and still retain the heading in the first panel. Now run it.

The last string, B$, was too long for the screen and CoCo was forced to scroll up in order to get to the next blank space. Scrolling upward pushed the name up and off the screen. The address remained. This did not look very professional.

One solution was to remove the remaining heading from the text panel. Line 1005 was one quick way to do it. Nothing was printed in the leftmost locations in the two top rows concerned. Delete Line 1001 and run.

It could have been centered vertically a little better. Care to have a go? I will later on because it irritates me.

Delete Line 1010. Insert 1125 GOTO 1125. Watch the last part of the action. Now run. List lines 1005 to 1125.

I placed the program section separators, lines 1100 and 1300, in what seems to be the wrong places. They really should be just before the CLS lines. This is a real-live, hand-crafted program, with insertions and deletions, and to give it a feeling of spontaneity I decided to show warts and all. Remember, you are expected to rearrange the program lines to suit yourself.

Line 1110 kept the display panel on the screen for 1500 units (alter this value to suit), and then the screen was blanked out to prepare for the next panel. Run it. Delete Line 1125 and list lines 1130 to 1140.

The second panel had two batches of

text. Using the same line formulations as in lines 980 to 1000, this text is centered for better appearance, then programmed in two segments.

Rekey 100 NEXT. Add 6 GOTO 1120 and 1120 GOTO 1220.

Suppose you wanted a breather after C$ was placed on the screen to absorb what was written? Instead of a pause loop, you could add a bunch of spaces at the end to enlarge C$.

List Line 1130. Edit this line by moving the closing quote mark until it is under the 's' in "screen." Run it.

CoCo will have to cycle through all these additional blanks, because T is now increased to include all these spaces. They take time to loop through and, in effect, you get an extra pause without the addition of a conventional pause line. Run again.

You can adjust it by adding or removing these trailing spaces. Press BREAK. Now, edit Line 1140, and add some trailing blanks. Mask Line 1210 and delete Line 1220. Now run it.

You can see that the extra time CoCo needs to cycle through T multiplied by the number of trailing spaces, replaces the run-of-the-mill pause routine in Line 1210.

You were issued a challenge in D$. It wasn't demonstrated, but I am sure it can be done. Work on it later!

Rekey 6 GOTO 1130. Remove any trailing blanks in lines 1130 and 1140. Unmask Line 1210. Insert 1135 GOTO1300 and 1350 GOTO 1350. Now run.

Did you see what we did? We restored some lines to their original state and we bypassed the heading and the rest of the first panel. We picked up C$ and bypassed the second panel. List lines 1300 to 1340.

If you mask Line 1305 and pull out +X$ in lines 1320 and 1330, you may understand this better. Make these suggested changes and run.

The text runs nicely across the screen, albeit backward, and it is kind of confusing until it is completely displayed. I thought it might look more dignified if it was led by an arrow, since there were a few unused spaces on the last row where it settled into its final home.

List lines 1300 to 1340. Unmask Line 1305. A small string, X$, was created four units long. Restore +X$ in Line 1320. The length was increased by four, in the new string, C$+X$. Restore +X$ in Line 1330. This meant that we have an enlarged string to display and, as we are starting from the right end of the string, our guiding arrow would lead the sentences as they flowed onto the screen. Run to make sure.

We don't need the arrow any more. In fact, it is an affront to our artistic sensibilities. We don't waste any time, and remove it forthwith. Line 1405 does the job. The location was chosen by trial and error. Delete Line 1350.

Trial and error does not mean that you throw up your hands and take a wild guess. No! You take a reasonable guess. What follows is how my mind ticks, if ever so slowly.

The target location was at the end of the eighth row. The ninth row begins with Location 256. I backed up a few digits to 250 and tried it out. When it was tested, I saw I knocked off the period. An adjustment was made.

As an aside, many moons ago, you were asked to make a training aid for yourself, on part of which, you were advised to make a PRINT@ list of the starting locations of each row. I made mine and refer to it constantly. It is no big effort to look on top of my TV for the guide. You don't think for a minute that I remember all the nuts and bolts in CoCo's vast storehouse of knowledge?

Delete Line 6 and add 11 GOTO 1410. We move our starting offset to begin after CLS, so we could start in the middle of the third panel. Delete Line 1135 and run.

Line 1415 gave us a pause. Not too long. The last string, C$, had been redisplayed correctly, if not orthodoxly, so there wasn't much incentive to study it further. We had just located the arrow on the line above, so it wasn't mind-bending to pick up the correct location of the next program line, A$. The last panel, being well-formatted and well-centered, called for an abrupt end at Line 1999.

"These last cosmetic improvements separate the men from the boys."

Delete Line 11 and run to make sure it is working properly. The last paragraph, A$ (Line 1410), was located just below the unorthodoxly presented top paragraph, C$, to bind it in firmly and present the whole panel as a preconceived unit. To wit: It gives it a natural, flowing appearance and neat change of pace, first proceeding in one direction and then augmented, from the opposite direction.

Finishing a program always causes a quandary for the author. When is it really finished? A program is never finished. There is always something that can be polished to make it better.

It reminds me of an artist completing an oil painting in three days. Then he dabs a little here and a bit there for over three months, never quite satisfied with the result. When it is hanging in an art gallery, he can still spot areas where he could touch it up. It is never finished!

So, too, with this demo, I could make changes. However, at some point, unperfected though it may be, I have to say that this is it.

That is not to say that you can't continue to modify it. For instance, two changes could be made in the last panel. In Line 1330, change 64 to 32 and in Line 1405, change 251 to 251-32 or 219.

The first panel bugs me. With the period in Line 920, CoCo is ornery and scrolls up, destroying the heading. Two solutions come to mind — omit the punctuation mark or revise B$. However, the text is exactly what I want to say and the only change I would tolerate is to change THE to OUR. This doesn't affect the length of B$ — and I want my period! Neither, do I want to give up the blank row under the header.

GOES TO PROVE could be changed to PROVES, but the former has a sense of a continuing, repeated action. Thus, for me, there is no choice. Being stuck with panel one, I don't like the unbalanced, vertical centering. Despite the short holding pause at Line 1110, it could be scrolled up one row for a neater effect. Adding 1001 PRINT solves that problem.

These last cosmetic improvements separate the men from the boys. As you run through this program, you will find something that I overlooked that offends your sensibilities. By all means, feel free to incorporate your improvements so it gets your stamp of approval.

Since the backward presentation is unreadable, why not speed it up? Change Line 1305 by inserting a space in front of the closing quote mark. To Line 1320, add STEP2. Give it a long pause at Line 1415, from 500 to 1500.

We might just as well double-time it off the screen. Add the following lines:

```
1406 Y$="<--- ":FOR Z= 1 TO
  1500:NEXT
1407 FOR A= LEN(C$+Y$) TO 0
  STEP-2
1408 PRINT@64,RIGHT$(C$+Y$,A)
1409 FOR Z= 1 TO 50:NEXTZ,A
```

This spoils our final display. Help me to recenter the last A$ display.

It is so easy to go off on tangents. Did you change Line 1430 from 256 to 128?

One more tangent, and that's it! Add 1412 CLS8 and run. Add ; at end of Line 1430 and run. Shades of tutorials past.

□

210	216
600	149
920	87
1150	86
END	13

The listing: DEMO

```
Ø '<LISTING1> DEMO
5 CLEAR 5ØØ
1Ø CLS
2Ø A$="BETTY ANN WHITE"
3Ø B$="824 NE 56 ST.
4Ø C$="SEATTLE, WASHINGTON"
41 PRINT@8,LEFT$(A$,5);:FORZ=1TO
2ØØ:NEXTZ
42 PRINT MID$(A$,6,5);:FORZ=1 TO
 2ØØ:NEXTZ
43 PRINTRIGHT$(A$,5)
44 FOR Z= 1 TO 5ØØ:NEXT
45 CLS
46 '***
47 A$="JOSEPH KOLAR           "
5Ø FOR T= Ø TO LEN(A$)
6Ø PRINT@9, LEFT$(A$,T)
7Ø PRINT@41,LEFT$(B$,T)
8Ø PRINT@7Ø, LEFT$(C$,T)
9Ø FOR Z= ] TO 51:NEXT
1ØØ NEXT
1Ø1 FOR Z= 1 TO 5ØØ:NEXT
1Ø2 GOSUB 2ØØØ
1Ø3 PRINT@9,A$
1Ø4 FOR Z= 1 TO 5ØØ:NEXT
11Ø L=LEN(A$)
12Ø FOR A=L TO Ø STEP-1
13Ø PRINT@3Ø-A," "RIGHT$(A$,A)
14Ø FOR B=1 TO 51:NEXT
15Ø NEXTA
16Ø '***
21Ø A$="JOSEPH KOLAR"
22Ø L=LEN(A$)
23Ø FOR A=Ø TO L
24Ø PRINT@213-A," "+LEFT$(A$,A)
25Ø FOR B=1 TO 2ØØ:NEXT
26Ø NEXT A
27Ø FOR Z=1TO5ØØ:NEXT
3ØØ FOR A=L TO Ø STEP-1
31Ø PRINT@2Ø2,RIGHT$(A$,A)
32Ø FOR B=1 TO 2ØØ:NEXT
33Ø NEXT A
34Ø '***
4ØØ A$="BETTY ANN WHITE"
41Ø FOR T= Ø TO LEN(A$)
42Ø PRINT@8,LEFT$(A$,T)
43Ø FOR Z= 1 TO 5Ø:NEXT
44Ø NEXT
45Ø FOR Z= 1 TO 1ØØØ:NEXT
46Ø '***
49Ø L=LEN(A$)
5ØØ FOR A=L TO Ø STEP-1
51Ø 'PRINT@8,RIGHT$(A$,A)
511 PRINT@8,LEFT$(A$,A)
52Ø FOR Z=1 TO 2ØØ:NEXT
53Ø NEXT A
535 FOR Z= 1 TO 5ØØ:NEXT
54Ø '***
6ØØ A$="JIMY OWEN WHITE"
6Ø5 PRINT@8,A$
6Ø6 FOR Z= 1 TO 5ØØ:NEXT
61Ø 'FOR T= LEN(A$) TO Ø STEP-1
611 FOR T=Ø TO LEN(A$)
62Ø PRINT@8,LEFT$(A$,T)
63Ø FOR X= 1 TO 2ØØ:NEXT
64Ø NEXT T
65Ø FOR Z= 1 TO 5ØØ:NEXT
66Ø '***
7ØØ A$="BETTY ANN WHITE"
71Ø FOR T=Ø TO 15
72Ø PRINT@8,RIGHT$(A$,T)
73Ø FOR X=1 TO 2ØØ:NEXT
74Ø NEXT T
75Ø FOR Z= 1 TO 5ØØ:NEXT
8ØØ '***
81Ø B$="JIMY OWEN WHITE"
811 L=LEN(B$)
82Ø FOR A= Ø TO L
83Ø PRINT@22-A," "+LEFT$(B$,A)
84Ø FOR X= 1 TO 2ØØ:NEXT
85Ø NEXT A
86Ø FOR Z=1TO5ØØ:NEXT
9ØØ '*****
91Ø A$="  THIS PROGRAM WAS CREAT
ED TO    FOOL AROUND WITH SOME OF
 THE     CAPABILITIES OF THE <LEF
T$> AND <RIGHT$> FUNCTIONS OF TH
E COCO.   AS A CHANGE OF PACE, T
HIS TEXTYOU ARE READING, WAS PLA
CED ON  THIS DISPLAY USING <LEFT
$>."
92Ø B$="  IT JUST GOES TO PROVE
THAT     THERE IS MORE THAN ONE W
AY TO   DO ALMOST ANYTHING WHEN
IT COMESTO PROGRAMMING THE BELOV
ED COCO."
94Ø FOR T= O TO LEN(A$)
95Ø PRINT@128, LEFT$(A$,T)
96Ø FOR Z=1 TO 5Ø:NEXT
97Ø NEXT
98Ø FOR T=Ø TO LEN(B$)
99Ø PRINT@352, LEFT$(B$,T)
1ØØØ FOR Z= 1 TO 5Ø:NEXTZ,T
1ØØ5 PRINT@Ø,"":PRINT@32,""
11ØØ '***
111Ø FOR Z= 1 TO 15ØØ:NEXT
112Ø CLS
113Ø C$="  A SECOND DISPLAY PANE
L WAS    ADDED TO SHOW ONE WAY T
O AVOID  <INPUT PRESS 'ENTER'> R
OUTINES.   THIS WAS DONE, AGAIN,
 TO SHOW HOW TO USE AN ALTERNATE
 WAY TO  GET THE TEXT TO THE SCR
EEN."
114Ø D$="  CAN YOU CHANGE THE RO
UTINE TO UTILIZE <RIGHT$> INSTEA
D OF     <LEFTS> TO PUT THIS TEX
T ONTO   THE SCREEN?  TRY IT AND
 SEE IF  YOU CAN COME UP WITH A
WORKABLE ROUTINE!"
115Ø FOR T=Ø TO LEN(C$)
116Ø PRINT@64, LEFT$(C$,T)
117Ø FOR Z= 1 TO 5Ø:NEXTZ,T
118Ø FOR T=Ø TO LEN(D$)
119Ø PRINT@256, LEFT$(D$,T)
12ØØ FOR Z=1 TO 5Ø:NEXTZ,T
121Ø FOR Z=1 TO 5ØØ:NEXT
13ØØ '***
13Ø5 X$="--->"
131Ø CLS
132Ø FOR A=Ø TO LEN(C$+X$)
133Ø PRINT@64,RIGHT$(C$+X$,A)
134Ø FOR Z=1 TO 5Ø:NEXTZ,A
14ØØ '***
14Ø5 PRINT@251,""
141Ø A$="  NOT VERY PRACTICAL!
BUT THAT IS ONE OF THE PERILS OF
 CREATIVEEXPERIMENTATION.  THE B
EGINNER  GETS TO DO LOTS OF USEL
ESS      THINGS, BUT IT IS STILL
 LOTS OF FUN!"
1415 FOR Z= 1 TO 5ØØ:NEXT
142Ø FOR T=O TO LEN(A$)
143Ø PRINT@256,LEFT$(A$,T)
144Ø FOR Z= Ø TO 5Ø:NEXTZ,T
1999 GOTO 1999
2ØØØ '
2Ø1Ø FOR X= 1 TO 1Ø
2Ø2Ø PRINT@9,"    WRONG!"
2Ø3Ø FOR Z= 1 TO 2Ø:NEXT
2Ø4Ø PRINT@12,"          "
2Ø5Ø FOR Z= 1 TO 4Ø:NEXT
2Ø6Ø NEXTX
2Ø7Ø RETURN
```

PRINTER

continued from 19

. The assembly language is simply to explain the listing and would need to be altered to suit an assembler if used.

```
 3FAE
   .8E0167: SETUP  LD X 'hook' change
   B1 A684:        LD A ,X     output
  .A78C22:         ST A GOON,P vector
   6 867E:         LD A 'jmp'
   8 A780:         ST A ,X+   new
   A EC84:         LD D ,X
  .ED8C1A:         ST D GOON+1,P
  .338C03:        LEA U CHRIN,P
  C2 EF84:         ST U ,X    new
   4   39:        RTS

3FC5 3406: CHRIN PSH S D
   7 D66F:        LD B DEVTYPE
   9   5C:        INC B    printer?
   A 2A0A:        BPL CONT      not
   C 810D:        CMP A 'CR'
   E 260B:        BNE CHRTR
  D0 C600:        LD B 'Para'
   2 8D11:        BSR LNFD
   4 A7E4:        ST A ,S replace CR

   6 3506:  CONT PUL S D
  .000000:  GOON  - - - - old vector

   B D69C: CHRTR  LD B CNT  check
   D C100:        CMP B 'End' line
   F 26F5:        BNE CONT
  E1 8D15:        BSR LINE
   3 20F1:        BRA CONT

   5 8D16:  LNFD BSR CHROUT
   7   5A:       DEC B
   8 26FB:       BNE LNFD
   A 8610:        LD A 'Flg'  set
   C 8D0F:       BSR CHROUT  margin
   E 8600:        LD A 'Hipos'
  F0 8D0B:       BSR CHROUT
   2 8600:        LD A 'Lopos'
   4   5A:       DEC B  initial
   5 D79C:        ST B CNT  -1
   7   39:       RTS

  .CC0D00:  LINE  LD D 'CR,Lnsp'
   B 8DE8:        BSR LNFD
  .7EA2BF:CHROUT JMP PRINT
```

Top of Memory - $4000h

More on the New Video Display Generator

By Tony DiStefano
Rainbow Contributing Editor

Last month I described the new VDG (Video Display Generator) MC6847T1 and the modes that are possible. I also showed you how to hook up a few switches in order to access these modes. The only problem with this is the new VDG is only available in the CoCo 2 'B' model. At home, I have the regular white CoCo. They call it the 'F' board. I wanted the new T1 chip in my CoCo, too. So, with the help of Bill Warnica, I modified my 'F' board CoCo to work with this new chip.

The new VDG and the old VDG are very similar but not pin-for-pin compatible, so you can't just pull the old one out and plug the new one in. It is, however, not too difficult to modify the computer board to make it fit. The new VDG also has built-in hardware that saves two chips on the computer board. The chips that are saved are no longer on the 'B' board. That is why the new board is smaller than the older boards. The two chip numbers saved are the 74LS244 and 74LS273. These chips are TTL logic gates used to isolate the CPU data bus from the video data bus.

Without getting into too much detail, these two chips are now part of the VDG and are no longer needed on the main board. At first, it was thought that both of these chips had to be removed from the old PC board and the new VDG completely rewired to fit in. Luckily, it turns out that only one of these chips has to be removed. This saves a lot of wiring.

Like most of my projects, this one requires you to open the computer and dig inside with a soldering iron and some tools. A good hardware hacker with experience is needed to do this one. To do this project, you will need a soldering iron, tools, wire, solder and, of course, a new VDG. More on the parts later.

The upgrade I did was on an 'F' board CoCo. As far as I know, these instructions work for just about every CoCo and CoCo 2, but on certain models, the VDG and other parts involved are soldered directly onto the PC Board. That means you have to unsolder the chips and insert a socket. This can be done, and I have done it many times, but it requires a solder sucker or chip remover. Soldering experience is necessary. Also, before you start, be forewarned! The jumpers I will tell you to install in the 'F' board may be different on different boards. But, not all is gloom and doom. A little trial and error and you should find the right pin numbers.

There are two parts you need. The first is the VDG, Motorola part number MC6847T1. If you cannot get this part at your local electronics store, try Radio Shack. The part number is MX-6551. The next part is just a plain and simple resistor. The resistor value is 1K or 1000 ohms quarter watt or half watt. That's it; the rest is a little bit of work.

Unplug the computer, undo the case, remove the keyboard, etc. You know, all those boring things.

Now comes the fun part. The first thing you must do is remove the VDG. That's simple. It's the chip marked MC6847, or U9 on the 'F' board. On other boards, the U number might be different but it will always be the MC6847. On some boards the VDG is soldered in. In that case, you must unsolder the VDG and insert a 40-pin socket. Prepare the new VDG (T1) in the following manner. Cut the resistor leads so that it will just fit between pins 25 and 11. Put the resistor across the top of the VDG and solder one end of the resistor to the top part of Pin 25. Make sure the solder doesn't leak down the pin. Next, solder the other end to Pin 11 (same precaution). Now pry out Pin 31 vertically, so it does not insert into the socket when you plug the new VDG in.

Insert the new VDG into the socket. Make sure Pin 1 is in the right place. Now solder a short piece of wire-wrap wire to Pin 1 of the VDG. Don't solder the pin to the socket. You won't be able to get the chip out if you do. (If you prefer, solder all connections to these pins before inserting the chip into the socket.) Solder the other end of this wire to Pin 31, the one that you bent up before. Solder a second wire to Pin 12 of the new VDG. Run this wire to Pin 10 of the SAM (Synchronous Address

"Never connect two outputs together, and never connect two inputs together."

Multiplexer). You remember ol' SAM, she's the one that does all the timing in the CoCo. I did an article on her not long ago in this magazine. Her name is MC6883 or SN74LS783N.

It was said, by whom I don't know, that you needed the new MC6885 or SN74LS785N SAM in order to make this new VDG work, but this rumor turns out to be false. The old one works just fine. As a matter of fact, I have the old SAM in my CoCo and it just purrs along. Anyway, back to work. Solder a third wire from Pin 13 of the VDG to Pin 12 of the SAM. That's about it for the VDG. But there is a little more work to do.

The next stage of this project deals with the buffer chip I mentioned earlier. Start off by removing the chip, number 74LS273, from its socket. You no longer need this chip, but keep it in your parts bin for a rainy day or in case you decide to remove the modification and replace the old VDG chip. The modification I did is on the so called 'F' board and the 74LS273 chip labeled U13 on the PC board. It also was not soldered in. It had a socket; all I had to do was pull it. If you are doing this on another board and the IC is not socketed, you must do a little more work. First, remove the old chip. Then solder in a 20-pin socket. You need the socket for this next step.

Prepare eight (about 1.5 inches) short pieces of wire by stripping 3/16 inch of insulation off each end. Use a number 22 or 24 gauge solid wire. Old Bell wire is best. Now insert each wire into the pins of the 20-pin socket as follows.

One End	**Other End**
3	12
4	15
7	9
8	6
13	2
14	5
17	16
18	19

Pins 1, 10, 11 and 20 are left empty. Do not connect anything to these pins. (Pin 10 is ground and Pin 20 is the 5-volt supply. You may use them if you need these power connections in other projects.)

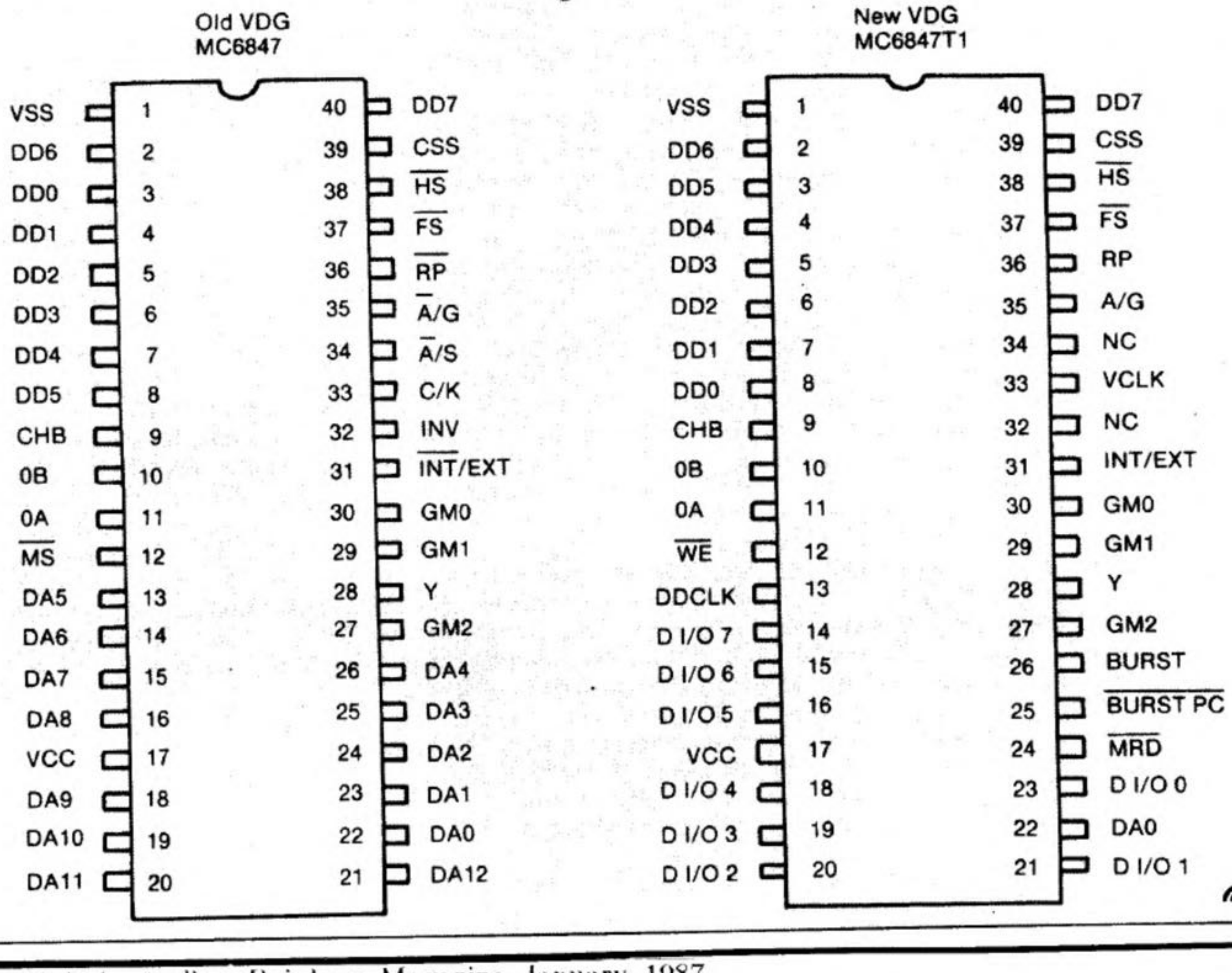
Now, this chip is called an Octal D-type flip-flop. If you recall, many moons (monthly articles) ago, I described flip-flops; they are no more than a sort of latch. This particular chip has eight latches. One for each of the eight data bits of the CPU. Each of these bits has an input and an output. I have arranged the pin numbers in such a way that the One End column pin numbers are all inputs and all the pin numbers in the Other End column are outputs. This is important to know. Notice that one jumper exists for every in/out pair. If you are trying to modify a board other than the 'F' board, the pin numbers may not match. Not having tried all the CoCos and CoCo 2s, I cannot print every pin diagram. Try to wire the connections as they stand above, but if the screen looks confused and you do not get the same letters on the screen you type on the keyboard, it's because the pinout is different.

In that case you will have to do a trial and error method to get the right combination. There are two rules to follow: Never connect two outputs together, and never connect two inputs together. The first may cause permanent damage to your computer. Jumper all eight wires and try it. If it is not right, make note of the combination you did and try another. If you do combinations in order, you will eventually get the right combination. When you do, if you send me the pinout combination and which computer board you did it on, I will print them in the next article I write and give you credit for it.

That's all there is to it! Plug everything back in and turn it on. You now have the new VDG in your CoCo. If you want to access the new modes of the new VDG, you will have to do a little more work. Last month, I wrote on how to access the new modes using switches or software. It works for this modification perfectly. All you have to do is follow the instructions and use the method that suits you best. Next month, I'll show you how to use the new modes without switches. All you will need are a few electronic parts. When you change modes from text to graphics, you won't have to throw all your switches — the electronics will do it for you.

For those who are interested, Figure 1 shows the pinouts of the old and the new VDGs side by side so you can compare the differences between them. □

Figure 1



Alien Raiders

by Darrel Behrmann

RAIDERS is a game requiring skill and concentration. Aliens are stealing supplies from the planet and you must stop them by docking with their ships. To dock with an alien ship, you must be directly above it and not moving at the time of contact. The lower on the screen you are (closer to the bottom) at the time of contact, the more points you score. You must also avoid the bombs traveling up the screen.

The game ends when time runs out or your ship is destroyed by bombs. To begin the game, press the fire button while the theme song is playing on the title screen.

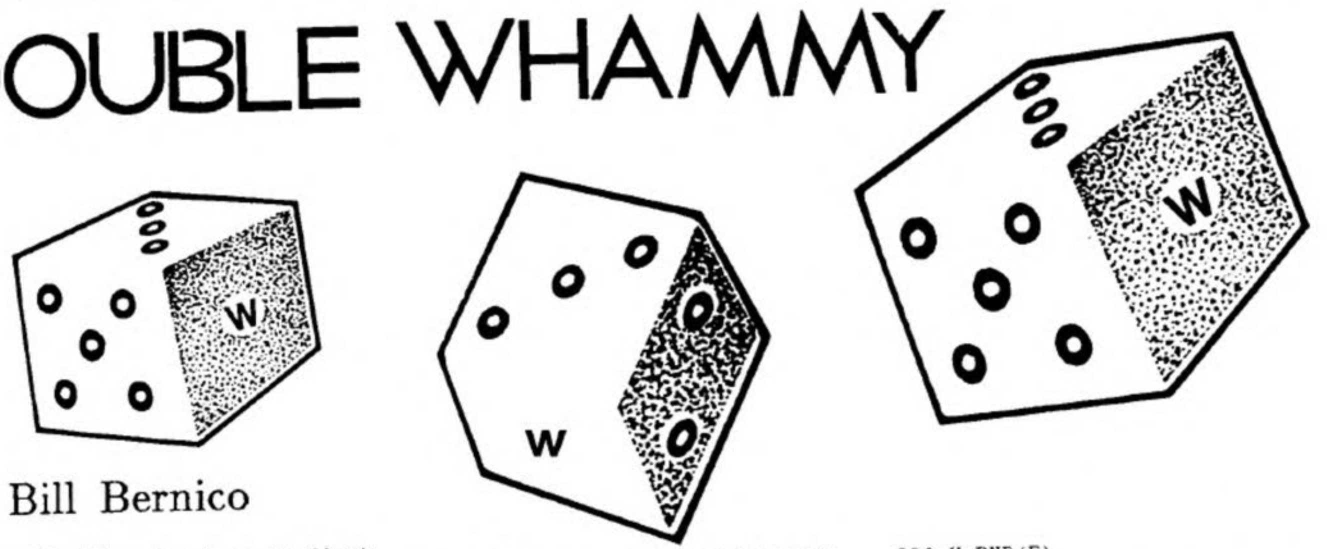
The Listing: RAIDERS

```
1 '****************************
2 '*          RAIDERS         *
3 '*             BY           *
4 '*      DARREL BEHRMANN     *
5 '*     U-251 RD. 16 RT. 1   *
6 '*     NAPOLEON, OH 43545   *
7 '*        JANUARY, 1986     *
8 '****************************
9 POKE65495,0
10 CLEAR 500
11 DIM EM(8,12),CS(8,12),SS(8,12
),AN$(15)
12 GOSUB 95'GET LETTERS
13 PMODE4,1:PCLS0:SCREEN1,1:PMOD
E3,1
14 COLOR3,4
15 LINE(0,165)-(255,192),PSET,BF
16 LINE(10,164)-(10,104),PRESET
17 LINE-(40,94),PRESET
18 LINE-(70,104),PRESET
19 LINE-(70,164),PRESET
20 LINE-(10,164),PRESET
21 PAINT(40,104),4,4
22 LINE(30,84)-(50,104),PRESET,B
F
23 LINE(10,74)-(70,84),PRESET,BF
24 CIRCLE(110,154),10,4
25 CIRCLE(150,154),10,4:CIRCLE(1
75,154),10,4
26 PAINT(110,154),2,4:PAINT(150,
154),2,4:PAINT(175,154),2,4
27 DRAW"BM80,155;C3U10R10E15R90D
25L115"
28 PAINT(90,150),3,3
29 LINE(105,135)-(125,145),PRESE
T,B
30 PAINT(120,140),2,4
31 DRAW"BM16,50;S16C2"+AN$(14)+A
N$(10)+AN$(13)+AN$(11)+AN$(12)+A
N$(14)+AN$(15)
32 DRAW"C4S4"
33 IFPEEK(65280)<>126 AND PEEK(6
5280)<>254THENPLAY"T12L102ABD":I
FPEEK(65280)<>126 AND PEEK(65280
)<>254THENPLAY"L2CFO1ABDL2CF":GO
TO3334 PLAY"V10T255L255"
35 PMODE4,1:PCLS0
36 CIRCLE(3,8),3,5:LINE(0,0)-(3,
5),PSET:LINE(3,0)-(3,5),PSET:LIN
E(7,0)-(3,5),PSET:PAINT(3,8),5,5
37 GET (0,0)-(7,11),EM
38 PCLS0
39 LINE(0,0)-(7,1),PSET,BF:LINE(
3,2)-(4,3),PSET,BF:LINE(3,3)-(0,
6),PSET:LINE-(0,11),PSET:LINE-(7
,11),PSET:LINE-(7,6),PSET:LINE-(
4,3),PSET:PAINT(3,9),5,5
40 GET(0,0)-(7,11),CS
41 PCLS0
42 LINE(0,0)-(3,11),PSET:LINE-(4
,11),PSET:LINE-(7,0),PSET:LINE-(
0,0),PSET:PAINT(3,3),5,5
43 GET(0,0)-(7,11),SS
44 PCLS0
45 POKE65495,0:PH=16:PV=1:E1=0:E
2=0:G1=0:G2=0:SC=0
46 SCREEN1,1:PCLS0:TIMER=0
47 LINE(PH*8,PV*12)-(PH*8+7,PV*1
2+11),PRESET,BF
48 J=JOYSTK(0):J1=JOYSTK(1):GOSU
B87
49 IFJ<15THENPH=PH-1ELSEIFJ>45TH
ENPH=PH+1
50 IFJ1<15THENPV=PV-1ELSEIFJ1>45
THENPV=PV+1
51 IFPH>31THENPH=31ELSEIFPH<0THE
NPH=0
52 IFPV>15THENPV=15ELSEIFPV<1THE
NPV=1
53 PUT(PH*8,PV*12)-(PH*8+7,PV*12
+11),SS
54 IFE1=0ANDRND(5)=5THENE1=1:H1=
RND(32)-1:V1=15
55 IFE2=0ANDRND(5)=5THENE2=1:H2=
RND(32)-1:V2=15
56 IFE1=1THENLINE(H1*8,V1*12)-(H
1*8+7,V1*12+11),PRESET,BF
57 IFE2=1THENLINE(H2*8,V2*12)-(H
2*8+7,V2*12+11),PRESET,BF
58 V1=V1-1:IFV1<1THENE1=0
59 V2=V2-1:IFV2<1THENE2=0
60 IFE1=1THENPUT(H1*8,V1*12)-(H1
*8+7,V1*12+11),EM
61 IFE2=1THENPUT(H2*8,V2*12)-(H2
*8+7,V2*12+11),EM
62 IF (V1=PV OR V1-1=PV) AND H1=
PH THEN GOSUB 76
63 IF (V2=PV OR V2-1=PV) AND H2=
PH THEN GOSUB 76
64 IFG1=0ANDRND(10)=10THENG1=1:H
3=RND(32)-1:V3=15
65 IFG2=0ANDRND(10)=10THENG2=1:H
4=RND(32)-1:V4=15
66 IFG1=1THENLINE(H3*8,V3*12)-(H
3*8+7,V3*12+11),PRESET,BF
67 IFG2=1THENLINE(H4*8,V4*12)-(H
4*8+7,V4*12+11),PRESET,BF
68 V3=V3-1:IFV3<1THENG1=0
69 V4=V4-1:IFV4<1THENG2=0
70 IFG1=1THENPUT(H3*8,V3*12)-(H3
*8+7,V3*12+11),CS
71 IFG2=1THENPUT(H4*8,V4*12)-(H4
*8+7,V4*12+11),CS
72 IF V3-1=PV AND H3=PH THEN GOS
UB79
73 IF V4-1=PV AND H4=PH THEN GOS
UB79
74 GOSUB 81
75 GOTO 47
```

continued to page 29

16K
ECB

DOUBLE WHAMMY



by Bill Bernico

I would like to share my first game with the rest of the readers. It's called WHAMMY, and is based on the old dice game called Skunk.

The object of Whammy is to acquire a preset number of points before your opponet does. Each player takes turns rolling the dice. You may roll as many times as you like, provided you don't roll a one, which in the game is represented by a 'W' (for Whammy).

With each roll, you accumulate the number of points made on that roll. If you choose not to roll again, you keep the points accumulated during that turn. Roll a one (W) and you lose all the points from that turn and the dice are passed to the next player. If you roll two ones (Ws), it's a double whammy and you lose all the points you have earned throughout the game. That's especially aggravating near the end when you have more to lose.

There's some skill required to know when to stop and pass the turn to someone else, but this game has the one element that is essential to a good game - greed. The temptation to roll just one more time when you're behind makes for some excitement during the game.

One last note on the screen presentation used. It contains the statement POKE 359,60:SCREEN 0,1 to turn the screen a shade of orange when a whammy is hit. Simply follow the next direction on the screen and press ENTER to continue. The screen will return to normal. If you should press the BREAK key while the screen is orange, the program will hang up. Just press the RESET button, type POKE 359,126 and press ENTER. It won't show up on the screen, but the fix will be made and you'll see the green screen with the OK prompt once again.

The Listing: WHAMMY

```
10 ' THE GAME OF WHAMMY
20 ' BY BILL BERNICO
30 ' 708 MICHIGAN AVE.
40 ' SHEBOYGAN, WI 53081
50 ' (414) 459-7350
60 '
70 D=0:E=0:F=0:G=0:H=0:I=0:J=0:K
=0:L=0:Z=43345
80 C$=CHR$(170):D$=CHR$(165):E$=
CHR$(172):F$=CHR$(163):Y$=CHR$(1
59):R$=CHR$(191):L$=CHR$(175)
90 CLS:PRINT@136,"THE GAME OF wh
ammy
100 PRINT@201,"BY BILL BERNICO
110 PRINT@0,STRING$(32,223);:PRI
NT@480,STRING$(31,223);:POKE1535
,223:FORX=32TO448STEP32:PRINT@X,
CHR$(223);:NEXT
120 FOR X=31 TO479STEP32:PRINT@X
,CHR$(223);:NEXT:PRINT@33,STRING
$(30,207);:PRINT@449,STRING$(30,
207);
130 FORX=65TO417STEP32:PRINT@X,C
HR$(207);:NEXT:FORX=94 TO 478 ST
EP32:PRINT@X,CHR$(207);:NEXT
140 PRINT@66,STRING$(28,239);:PR
INT@418,STRING$(28,239);:FORX=98
TO386STEP32:PRINT@X,CHR$(239);:N
EXT
150 FORX=125TO413STEP32:PRINT@X,
CHR$(239);:NEXTX:SOUND89,3:SOUND
109,3:SOUND125,3:SOUND109,3:FORX
=1TO120:NEXT:SOUND125,3:SOUND89,
3:FORX=1TO200:NEXT:SOUND175,2
160 PRINT@358,"HIT ANY KEY TO BE
GIN";:EXEC44539
170 CLS:PRINT"NUMBER OF PLAYERS
(1-4)";:FOR X=1024 TO 1046:POKE
X,PEEK(X)-64:PLAY"O5T60B":EXEC Z
:NEXT X:INPUT F
180 IF F<1 OR F>4 THEN 170
190 PRINT@32,STRING$(32,150);
200 FOR G=1 TO F
210 PRINT:PLAY"O5T60B":EXEC Z:PL
AY"O4B":EXEC Z:PLAY"O5B":EXEC Z:
PLAY"O4B
220 PRINT"PLAYER #";G;:INPUT A$(
G):NEXT G
230 H=RND(F)
240 PRINT@384,"POINTS NEEDED TO
WIN";:FOR X=1408 TO 1427:POKEX,P
EEK(X)-64:PLAY"O5T60B":EXEC Z:NE
XT:INPUT L
250 CLS:GOSUB 790
260 IF F>2 THEN PRINT@64,STRING$
(32,191);
270 IF F<3 THEN PRINT@32,STRING$
(64,191);
280 PRINT@96,R$;R$;A$(H)"'S TURN
 ";STRING$(28,191);
290 PRINT@118,"goal"L;
300 PRINT@128,STRING$(32,191);
310 I=RND(6):J=RND(6):K=I+J
320 PRINT@160,STRING$(96,175);
330 PRINT@194,"your roll";:PRINT
@205,I;:PRINT@211,J;:PRINT@219,K
;:POKE1222,32:POKE1233,43:POKE12
39,61
340 PRINT@172,C$;F$;F$;F$;D$;
350 PRINT@204,C$;L$;:PRINT@207,L
$;D$;
360 PRINT@236,C$;E$;E$;E$;D$;
370 PRINT@178,C$;F$;F$;F$;D$;
380 PRINT@210,C$;L$;:PRINT@213,L
$;D$;
390 PRINT@242,C$;E$;E$;E$;D$;
400 PRINT@185,STRING$(7,175);
410 PRINT@249,STRING$(7,175);
420 PRINT@256,STRING$(32,255);
430 PRINT@288,STRING$(32,159);
440 PRINT@352,STRING$(32,159);
450 PRINT@416,STRING$(32,159);ST
RING$(32,255);
460 IF PEEK(1230)=113 THEN POKE
1230,23
470 IF PEEK(1236)=113 THEN POKE
1236,23
480 IF I=1 AND J=1 THEN PRINT@28
8,STRING$(160,159);:SOUND 1,14:S
OUND 34,4:SOUND 44,2:FOR X=1 TO
340:NEXT X:SOUND 1,9:GOTO600
490 IF I=1 OR J=1 THEN PRINT@288
,STRING$(160,159);:FOR X=1 TO 50
:PLAY"O1T60":EXEC Z:NEXT X:GOTO6
10
500 E=E+K:D(H)=D(H)+K
510 IF D(H)=>L THEN 710
```

continued to page 29

16K
ECB

Treasure Quest: The Golden Adventure

By Eric Tucker

You've made it to the second room. All is quiet, and right across from you is a gold bar! As you look around the room, you count a total of five bars of gold. You check the walls, but see nothing. Looking at the floor, there's nothing but gold. You step across the floor, and fall into a pit.

Treasure Quest is an arcade Adventure game that requires a 16K Extended Color BASIC CoCo with one joystick. It uses the speedup POKE (POKE 65495,0). If your machine can't handle it, edit lines 160 and 1130 to delete it.

The title screen has two parts. The first uses CoCo's eight colors and CHR$ blocks to write the title. The second is the graphics screen which shows you entering the castle.

The castle has three rooms: the Vacuum Room, the Magic Pit Room and the Wind Tunnel. In all three rooms your man is the blue square.

In the Vacuum Room, there is a large bar of gold on the other side of the room. You must cross the room, get the gold and return to the blue door. On either side of the room are two large ducts. Avoid both of them, or you'll get pulled in by the suction from the duct. If you get close to the gold, you may be repelled by an invisible force field. It comes and goes at random, so keep trying. Once you get the gold, you must go back across the room to the blue door to get out.

The second room is the Magic Pit Room. There are five bars of gold situated throughout the room. Surrounding them are large, square, disappearing pits. You must avoid getting caught when one appears or down you go! After picking up all the gold, a small blue key may appear on the screen. Pick up this key and you are awarded 100 points and an extra man. However, you must then go back and go through the room again.

The third room, the Wind Tunnel, is the easiest of the three. To your right is a large fan and three gold keys. To the left is a blue spiked wall. You must fight against the fan to get the three keys. During your struggle, the spiked wall moves slowly but surely toward you. Once you reach all three keys, you are congratulated, then sent back to the Vacuum Room.

To begin, you are given three men. The only way to gain an extra one is to get the small blue key in the Magic Pit Room. All treasures except the blue key are worth 250 points. You automatically get five points each time you move. In the Vacuum Room, you must pass between the ducts to get the points. After passing all three boards, you are given another 100 points.

At the end of the game, your score and the high score are displayed. A spring-centered joystick should be used to best feel the effects of the first and third rooms. I hope to see some of your scores on the Scoreboard.

If your man doesn't look like a square, press BREAK and run the program again.

140	121	760	120
230	72	900	17
360	89	1010	104
470	209	1130	67
580	147	END	181
640	171		

The listing: TREASURE

```
20 CLEAR200:DIM MAN(9,11)
30 PCLS:DRAW"C3":LINE(100,92)-(1
04,96),PSET,BF:GET(98,88)-(108,9
8),MAN,G:SC=0
40 CLS0:C=0
60 FORT=1TO8:PRINT@34,STRING$(3,
143+(C*16));:PRINT@67,CHR$(143+(
C*16));:PRINT@99,CHR$(143+(C*16)
);:PRINT@131,CHR$(143+(C*16));
70 B$=CHR$(143+(C*16)):B2$=CHR$(
140+(C*16)):B3$=CHR$(131+(C*16))
:G$=CHR$(128)
80 PRINT@70,B$+B$+G$+B$+CHR$(140
+(C*16))+B$+G$+B2$+B2$+B$+G$+B$+
B2$+B2$+G$+B$+G$+B$+G$+B$+B$+G$+
B$+B2$+B$;
90 PRINT@102,B$+G$+G$+B$+B3$+B$+
G$+B$+B2$+B$+G$+B2$+B2$+B$+G$+B$
+G$+B$+G$+B$+G$+G$+B$+B3$+B$;
100 PRINT@134,B$+G$+G$+B$+B3$+B3
$+G$+B$+B3$+B$+G$+B3$+B3$+B$+G$+
B$+B$+B$+G$+B$+G$+G$+B$+B3$+B3$;
110 PRINT@200,B$+B$+B$+B$;:PRINT
@218,B$;
120 PRINT@232,B$+G$+G$+B$+G$+B$+
G$+B$+G$+B$+B2$+B$+G$+B$+B2$+B2$
+G$+B2$+B$+B2$;
130 PRINT@264,B$+G$+CHR$(137+(C*
16))+B$+G$+B$+G$+B$+G$+B$+B3$+B$
+G$+B2$+B2$+B$+G$+G$+B$;
140 PRINT@296,B$+B$+B$+B$+G$+B$+
B$+B$+G$+B$+B3$+B3$+G$+B3$+B3$+B
```

```
$+G$+G$+B$;:PRINT@332,CHR$(136+(
C*16));
150 C=C+1:NEXT:PRINT@395,"BY ERI
C TUCKER";:SCREEN0,1:FORT=1TO200
0:NEXT
160 SOUND200,4:POKE65495,0:PMODE
3:PCLS2:MD=0
170 DRAW"BM0,50;C4;F10D15R15U30E
10C3U10R5D1R3D1L8D8C4F10D30R15U1
5E10F10D65L80U75":PAINT(6,60),4,
4
180 DRAW"C1":LINE(0,126)-(86,191
),PSET,BF:LINE(144,126)-(255,191
),PSET,BF:DRAW"C3":LINE(88,130)-
(142,191),PSET,BF:LINE(0,160)-(2
55,191),PSET,BF
190 DRAW"C4":LINE(192,125)-(194,
120),PSET,BF
200 LINE(82,126)-(82,66),PSET
210 SCREEN1,0:FORT=1TO2000:NEXT:
X=82:Y=66
220 FORT=0TO14:DRAW"C4":LINE(82,
66)-(X,Y),PSET:LINE(82,126)-(X,Y
),PSET:SOUND245,1:FORZ=1TO200:NE
XT:DRAW"C2":LINE-(82,126),PSET:X
=X+4:Y=Y+4:NEXTT
230 DRAW"C3":LINE(82,126)-(X,Y),
PSET:LINE-(82,66),PSET:PLAY"T6L1
6V30O1CCCV25CCCV20CCCV15CCCV10CC
CV5CCC"
240 FORT=1TO1000:NEXT
250 X=192:Y=125:FORT=1TO56:DRAW"
C3":LINE(82,66)-(142,126),PSET:L
INE(X,Y)-(X+2,Y-5),PSET,BF:FORQ=
1TO30:NEXT:PLAY"L32T20V30O1DBV20
DBV10DB":DRAW"C2":LINE(X,Y)-(X+2
,Y-5),PSET,BF:X=X-2:NEXT
260 FORT=1TO1500:NEXT
270 SOUND100,1
280 CLS:PRINT@107,"VACUUM ROOM":
PRINT@198,"PICK UP THE GOLD BAR"
:PRINT@330,"YOUR SCORE:";SC:PRIN
T@362,"MEN LEFT:";3-MD
290 FORT=1TO3500:NEXT
300 REM vacuum room
310 PCLS:DRAW"C4BM100,0;G40R152H
40;BM100,191;H40R152G40":PAINT(1
10,5),4,4:PAINT(100,185),4,4:DRA
W"C2BM10,100E8R15F8L31"
320 PAINT(15,98),2,2
330 DRAW"C4":LINE(0,40)-(255,40)
,PSET:LINE(0,151)-(255,151),PSET
340 DRAW"C3":LINE(252,86)-(255,1
06),PSET,BF
350 PX=200:PY=96:PUT(PX,PY)-(PX+
10,PY+12),MAN,PSET
360 SCREEN1,0:POKE65314,248:FORT
=1TO500:NEXT
370 PLAY"T3O2L32V5GBGBGV10BGBGBV
15GBGBGV20BGBGBV25GBGBGV30BGBGB"
:CT=0:GD=0
380 GOSUB 1140
385 IF PX<=42THENPX=42
390 PLAY"T200V10L64O2CD"
400 IF PX>56 AND PX<202 THEN SC=
SC+5
430 QW=RND(10):IF QW<8 AND PX=60
 THEN PX=60:IFPY<96THENDRAW"BM58
,45;C4;D51;L2U51C1;D51R2U51;" EL
SE DRAW"BM58,96;C4;D51L2U51;C1;D
51R2U51"
431 IF QW<8 AND PX=60 THEN PX=86
:PY=PY+1:PUT(PX,PY)-(PX+10,PY+12
),MAN,PSET:LINE(60,PY)-(70,PY+12
),PRESET,BF:PLAY"T50L32V31O4CDV2
1CDV11CD"
440 PUT(PX,PY)-(PX+10,PY+12),MAN
,PSET:IF PY+4>96 THEN 460
450 IF PX<202 AND PX>56 AND PY+4
<96 THEN PY=PY-(RND(4)-1)
460 IF PX<202 AND PX>56 AND PY+4
>96 THEN PY=PY+(RND(4)-1)
470 IF PY<=42 OR PY>=138 THEN GO
SUB1080:IFMD<3THEN280
480 IF PX>41 AND PX<47 AND PY>=8
5 AND PY<=100 AND GD=0 THEN PAIN
T(15,98),1,1:PLAY"V31T5L32CDEFGA
B":GD=1:SC=SC+250
490 IF GD=1 AND PX>=240 THEN PLA
Y"T6L32V30O4BBBV25BBBV20BBBV15BB
BV10BBBV5BBB":GOTO510
500 GOTO 380
510 REM open pit room
520 FORT=1TO1500:NEXT:SOUND100,1
530 CLS:PRINT@104,"MAGIC PIT ROO
M":PRINT@197,"PICK UP THE GOLD B
ARS":PRINT@330,"YOUR SCORE:";SC:
PRINT@362,"MEN LEFT:";3-MD:FORT=
1TO3500:NEXT:FORT=1TO5:GD(T)=0:N
EXT:XY=0:BZ=0
540 GX(1)=80:GY(1)=40:GX(2)=80:G
Y(2)=150:GX(3)=140:GY(3)=96:GX(4
)=212:GY(4)=60:GX(5)=212:GY(5)=1
30
550 PCLS:FORT=1TO5:DRAW"BM"+STR$
(GX(T))+","+STR$(GY(T))+";C2E4R6
F4L14":PAINT(GX(T)+2,GY(T)-1),2,
2:NEXT
560 DRAW"C3":LINE(0,86)-(4,100),
PSET,BF
570 HX(1)=60:HY(1)=84:HX(2)=132:
HY(2)=10:HX(3)=132:HY(3)=150:HX(
4)=192:HY(4)=80:SCREEN1,0:POKE65
314,248:PX=12:PY=92:HX(5)=40:HY(
5)=15:HX(6)=40:HY(6)=146:HX(7)=1
32:HY(7)=42:HX(8)=132:HY(8)=118:
HX(9)=192:HY(9)=20:HX(10)=192:HY
(10)=140
580 R=RND(10):DRAW"C4":LINE(HX(R
),HY(R))-(HX(R)+30,HY(R)+30),PSE
T,BF
590 GOSUB 1140:LINE(PX,PY)-(PX+1
0,PY+12),PRESET,BF
591 IF H<5 THEN PX=PX-2
592 IF H>59 THEN PX=PX+2
593 IF J<5 THEN PY=PY-1
594 IF J>59 THEN PY=PY+1
600 SC=SC+5
605 XY=0:FORT=1TO5:IFGD(T)=1THEN
XY=XY+1:NEXT:IF BZ=0 AND XY=5 AN
D PX<=64 AND PX>35 AND RND(90)<=
2 THEN DRAW"S2C3BM140,96L4D8R4U8
D16G4F4E4H4S4":KX=140:KY=96:BZ=1
:SOUND1,1
606 IF BZ=1 AND PPOINT(KX,KY)<>3
 THEN GOSUB 5000:GOTO550
610 PUT(PX,PY)-(PX+10,PY+12),MAN
,PSET
620 Z=RND(10):IF Z<6 THEN LINE(H
X(R),HY(R))-(HX(R)+30,HY(R)+30),
PRESET,BF:SOUND1;1:GOTO580
630 XY=0:FORT=1TO5:IFGD(T)=1THEN
XY=XY+1:NEXTELSENEXT
640 IF XY=5 AND PX<=12 AND PY>85
 AND PY<100 THEN PLAY"T6L32V30O4
BBBV25BBBV20BBBV15BBBV10BBBV5BBB
":GOTO710
650 FORT=1TO5:IF PPOINT(GX(T)+12
,GY(T))<>2 AND GD(T)=0 OR PPOINT
(GX(T),GY(T))<>2 AND GD(T)=0 THE
N SOUND250,8:SC=SC+250:LINE(GX(T
),GY(T))-(GX(T)+14,GY(T)-5),PRES
ET,BF:GD(T)=1:NEXT ELSE NEXT
660 IF PPOINT(PX+12,PY+13)=4 THE
NGOSUB1020:IFMD<3THEN530
670 IF PPOINT(PX+12,PY-1)=4 THEN
 GOSUB 1020:IFMD<3THEN530
680 IF PPOINT(PX-2,PY-1)=4 THEN
GOSUB1020:IFMD<3THEN530
690 IF PPOINT(PX-2,PY+13)=4 THEN
 GOSUB1020:IFMD<3THEN530
700 GOTO 590
710 REM wind tunnel
720 FORT=1TO1500:NEXT:SOUND100,1
:CLS:PRINT@106,"WIND TUNNEL":PRI
NT@197,"TRY TO REACH THE KEYS":P
RINT@330,"YOUR SCORE:";SC:PRINT@
362,"MEN LEFT:";3-MD:FORT=1TO350
0:NEXT
721 K1=0:K2=0:K3=0:CT=0
730 CT=0
740 PCLS:D$="F6G6F6G6F6G6F6G6F6G
6F6G6F6G6F6G6F6G6F6G6F6G6F6G6F6G
6F6G6F6G6F6G6F6G6":DRAW"C3BM12,0
"+D$:DX=12
750 PAINT(1,5),3,3:DRAW"C4":LINE
(255,76)-(240,116),PSET,BF:LINE(
255,94)-(225,98),PSET,BF:DRAW"U2
U8D16H8E8":PAINT(223,96),4,4
760 LINE(228,10)-(230,181),PSET,
BF
770 KY$="L4D8R4U8D16G4F4E4H4"
780 DRAW"C2BM208,30"+KY$+"BM208,
140"+KY$+"BM208,85"+KY$
790 SCREEN1,0:POKE65314,248
800 PX=128:PY=92:FORT=1TO1000:NE
XT
810 GOSUB1140:IFPX>=212THENPX=21
2
815 IF J<5 THEN PY=PY-1
816 IF J>59 THEN PY=PY+1
820 IF PX<60 THEN PX=PX+1
830 IF PX>190 THEN PX=PX-1
840 SC=SC+5
850 CT=CT+1:IF CT/5=INT(CT/5) TH
EN DRAW"C3BM"+STR$(DX)+",0"+D$:D
X=DX+2
860 PX=PX-(RND(3)-1):PLAY"T200V2
0L64O1CCC"
870 PUT(PX,PY)-(PX+10,PY+12),MAN
,PSET
880 IF PPOINT(PX-2,PY)=3 OR PPOI
NT(PX-2,PY+12) 3 THEN GOSUB 1080
:IFMD<3THEN710
890 DRAW"BM228,10C3D171R1U171C4L
1D171R1U171L1"
900 IF PPOINT(PX+12,PY+13)=2 AND
 PY<45 AND PY>30 AND K1=0 OR PPO
INT(PX+12,PY)=2 AND PY<45 AND PY
>30 AND K1=0 THEN DRAW"C1BM208,3
0"+KY$:SOUND240,4:K1=1:SC=SC+250
910 IF PPOINT(PX+12,PY+13)=2 AND
 PY<155 AND PY>140 AND K3=0 OR P
POINT(PX+12,PY)=2 AND PY<155 AND
 PY>140 AND K3=0 THEN DRAW"C1BM2
08,140"+KY$:SOUND240,4:K3=1:SC=S
C+250
920 IF PPOINT(PX+12,PY+13)=2 AND
 PY<100 AND PY>85 AND K2=0 OR PP
OINT(PX+12,PY)=2 AND PY<100 AND
PY>85 AND K2=0 THEN DRAW"C1BM208
,85"+KY$:SOUND240,4:K2=1:SC=SC+2
50
930 IF K1=1 AND K2=1 AND K3=1 TH
EN 950
```

```
940 GOTO 810
950 REM cleared all boards
960 SOUND200,5:SOUND225,10
970 FORT=1TO2000:NEXT
980 CLS:PRINT@128,"  CONGRATULAT
IONS! YOU HAVE":PRINT"  COMPLETE
D THE THREE ROUNDS!"
990 SC=SC+1000:PRINT@320,"     YO
UR SCORE:";SC:SCREEN0,1
1000 SOUND200,5:FORT=1TO2000:NEX
T
1010 GOTO 260
1020 REM pitfall
1040 CLS0:PRINT@480,STRING$(31,2
07);:PRINT@15,CHR$(175);:Z=0
1060 FORT=15TO1STEP-1:PRINT@Z+15
,CHR$(175);:SOUNDT,1:PRINT@Z+15,
CHR$(128);:Z=Z+32:NEXT
1070 PRINT@396,CHR$(163);:PRINT@
403,CHR$(166);:PRINT@461,CHR$(16
8);:PRINT@463,CHR$(162);:PLAY"T6
L16V30O2CCCV25CCCV20CCCV15CCCV10
CCCV5CCC":FORT=1TO1500:NEXT:SCRE
EN1,0:POKE65314,248:GOSUB1080:RE
TURN
1080 FORT=1TO20:PUT(PX,PY)-(PX+1
0,PY+12),MAN,PRESET:SOUND250,1:P
UT(PX,PY)-(PX+10,PY+12),MAN,PSET
:NEXT
1090 MD=MD+1:IF MD<3 THEN RETURN
1100 PLAY"O1V25T3L2BL2.C"
1110 FORT=1TO2000:NEXT:CLS:PRINT
@64," T R E A S U R E   Q U E S
T":PRINT@192,"       YOUR SCORE:"
;SC:IF SC>HS THEN HS=SC:PRINT@22
4,"      HIGH SCORE:";HS ELSE PR
INT@224,"      HIGH SCORE:";HS
1115 PRINT@128,"       G A M E
O V E R"
1120 PRINT@288,"";:INPUT"ANOTHER
 GAME";I$
1130 IF LEFT$(I$,1)="Y"THENSC=0:
GOTO160 ELSE POKE65494,0:END
1140 H=JOYSTK(0):J=JOYSTK(1)
1150 IF H<5 THEN PX=PX-2
1160 IF PX<=8THENPX=8
1170 IF H>58 THEN PX=PX+2
1180 IF PX>=242THENPX=242
1190 IF J<5 THEN PY=PY-1
1200 IF PY<=8THENPY=8
1210 IF J>58 THEN PY=PY+1
1220 IF PY>=174THENPY=174
1230 RETURN
5000 PLAY"T3L32O4V5CDEFGABAGFEDC
V10CDEFGABAGFEDCV15CDEFGABAGFEDC
V20CDEFGABAGFEDCV25CDEFGABAGFEDC
V30CDEFGABAGFEDC"
5010 FORT=1TO1000:NEXT:CLS:PRINT
@128,"  CONGRATULATIONS! YOU PI
CKED":PRINT"        UP AN EXTRA M
AN!"
5020 PRINT@224,"MEN LEFT:";3-MD:
MD=MD-1:FORT=1TO2000:NEXT:PRINT@
224,"MEN LEFT:";3-MD:SOUND1,1:FO
RT=1TO1500:NEXT:SCREEN1,0:SC=SC+
100:BZ=0:FORT=1TO5:GD(T)=0:NEXT:
RETURN
```

WHAMMY

continued from page 26

```
520 PRINT@320,E;"points";Y$;"thi
s";Y$;"turn";STRING$(16,159);:PO
KE1344,159:IF PEEK(1346)=96THEN
POKE1346,159
530 GOSUB 790
540 PRINT@384,D(H);"total";Y$"po
ints";STRING$(20,159);:POKE1408,
159:IF PEEK(1410)=96THEN POKE 14
10,159
550 PRINT@487,"ROLL AGAIN (Y/N)?
";:FOR X=1510 TO 1528:POKE X,PEE
K(X)-64:PLAY"O3T60F":EXEC Z:NEXT
560 B$=INKEY$:IF B$=""THEN 560
570 IF B$="Y"THEN 250
580 IF B$="N"THEN 620
590 GOTO560
600 PRINT@320,STRING$(8,159);"do
uble";Y$;"whammy";STRING$(30,159
);:PRINT@384,STRING$(4,159);"you
";Y$;"lose";Y$;"all";Y$;"your";Y
$;"points";STRING$(20,159);:D(H)
=0:GOTO 620
610 PRINT@288,STRING$(44,159);:P
RINT@332,"whammy";STRING$(47,159
);:PRINT@385,"you";Y$;"lose";Y$;
"all";Y$;"points";Y$;"this";Y$"t
urn";STRING$(34,159);STRING$(32,
255);:D(H)=D(H)-E
620 H=H+1:E=0
630 IF H>F THEN H=1
640 GOSUB 650:GOTO 250
650 PRINT@485,"HIT <ENTER> TO CO
NTINUE";:FOR X=1504 TO 1535:POKE
 X,PEEK(X)-64:EXEC43345:NEXT X
660 IF I=1 AND J=1 THEN GOSUB 84
0
670 IF I=1 OR J=1 THEN GOSUB 850
680 IF INKEY$<>CHR$(13)THEN 680
690 POKE 359,126
700 RETURN
710 PRINT@320,STRING$(32,159);A$
(H);Y$;"wins";Y$;"with";Y$;D(H);
Y$;"points";STRING$(63,159);
720 PLAY"T60O1CDEFGABO2CDEFGABO3
CDEFGABO4CDEFGABO5CDEFGAB
730 GOSUB 790
740 PRINT@487,"PLAY AGAIN (Y/N)?
";:FOR X=1510 TO 1528:POKE X,PEE
K(X)-64:PLAY"O5T60B":EXEC Z:NEXT
750 C$=INKEY$:IF C$=""THEN 750
760 IF C$="N"THEN CLS:LIST-50:EN
D
770 IF C$="Y"THEN RUN
780 GOTO 750
790 PRINT@0,A$(1);D(1)
800 IF F=>2 THEN PRINT@16,A$(2);
D(2)
810 IF F=>3 THEN PRINT@32,A$(3);
D(3)
820 IF F=>4 THEN PRINT@48,A$(4);
D(4)
830 RETURN
840 FORX=1TO5:POKE359,60:SCREEN0
,1:PLAY"O4T60F":FORY=1TO100:NEXT
Y:POKE359,126:SCREEN0,0:PLAY"O3T
60F":FORY=1TO100:NEXTY:NEXTX:RET
URN
850 POKE 359,60:SCREEN0,1:RETURN
```

Raiders

continued from page 25

```
76 PLAY "V1":FORY=1TO30:PLAY"ABA
V+;":CIRCLE(PH*8+4,PV*12-4),RND(
15),RND(4):NEXTY:PLAY"V31":FORY=
1TO30:PLAY"CDCV-;":CIRCLE(PH*8+4
,PV*12-4),RND(15),RND(4):NEXTY
77 PLAY "V10"
78 GOTO 94
79 SC=SC+PV:SOUND PV*16,1
80 RETURN
81 'SCORING DISPLAY
82 PLAY "AABB"
83 LINE(0,0)-(32,12),PRESET,BF
84 TH=INT(SC/1000):HD=INT((SC-TH
*1000)/100):TN=INT((SC-TH*1000-H
D*100)/10):OE=SC-TH*1000-HD*100-
TN*10
85 DRAW "BM0,11;"+AN$(TH)+AN$(HD
)+AN$(TN)+AN$(OE)
86 RETURN
87 'TIMER DISPLAY
88 LINE(120,0)-(136,12),PRESET,B
F
89 TI=100-TIMER/60
90 TN=INT(TI/10):OE=TI-TN*10
91 IF TIMER/60 >100 THEN GOTO 94
92 DRAW"BM120,12;"+AN$(TN)+AN$(O
E)
93 RETURN
94 POKE65494,0:IF PEEK(65280)=12
6 OR PEEK(65280)=254 THEN 45 ELS
E 94
95 AN$(0)="BE1BU1U7E1R3F1D7G1L3U
1E1U2E1U2E1BD9BR3"'0
96 AN$(1)="BE1R5L2U9G3BD7BR6"'1
97 AN$(2)="BU8BR1E2R2F1D2G2L1G2D
2R6BF1"'2
98 AN$(3)="BE1BU1F1R2E2U2H1NL3E1
U2H1L3G2BD8BR8"'3
99 AN$(4)="BU9BR1D3R5L1U4D9BD1BR
3"'4
100 AN$(5)="BE1BU1F1R2E2U2H1L1H1
L1H1U2R6BD10BR1"'5
101 AN$(6)="BE2BU2E1R2F1D2G1L2H2
U5E1R1E1R1F1BD9BR2"'6
102 AN$(7)="BU8BE1E1R4D2G1D2G1D3
BD1BR4"'7
103 AN$(8)="BR3BU1H2U1E2H2E2R1F2
G2F2D1G2BD1BR4"'8
104 AN$(9)="BR5BU1U9L3G1D2F1R3BR
3BD6"'9
105 AN$(10)="BE1U4E1U3E1R1F1D3F1
NL4D4BF1BR1"'A
106 AN$(11)="BE1U9R2F1R1F1D5G1L1
G1L2BD1BR7"'D
107 AN$(12)="BE1U9R5L5D4R3L3D5R5
BD1BR2"'E
108 AN$(13)="BE1R5L3U9L2R5BD10BR
2"'I
109 AN$(14)="BE1U9R3F2D1G2L2F4BF
1BR1"'R
110 AN$(15)="BE1BU1F1R2E2U1H2L1H
2E2R2F1BD9BR2"'S
111 RETURN
```

Put worldwide communication right in your "packet"

Hamming It Up

By Len Popyack

When was the last time you lost sleep playing with your CoCo? For me it has happened only twice — once when I bought my computer back in 1981, and again after I bought my Packet Radio interface.

Have you ever thought about being able to call up anyone's computer system anywhere in the world, leave them a message, transfer a program, or simply chat with them — all for free? It is possible to communicate digitally with anyone via conventional toll telephone lines, but you always have to pay for it. It's also possible to communicate with any other licensed amateur radio operator (or his computer) using Packet Radio.

What Is Packet Radio?

Packet Radio is a way in which any Ham (slang for amateur radio operator) with a Terminal Node Controller (TNC) can send digital information to another Ham with a TNC. The data is transmitted at 1200 Baud and can be sent around the country. When the data arrives at its destination it is guaranteed to be error free. What more could one ask for!

The device that allows the radio transceiver to be connected to your CoCo is called a Terminal Node Controller. Most TNCs are actually self-contained microcomputers. The TNC allows you to connect to another TNC by commanding your TNC to make the connection, similar to the way you command the telephone company to connect you to another number by dialing the number.

Amateur radio packet communication actually got started in Canada about 1978. The Canadian government issued a special digital communications license, and packets of data were soon heard in Canada.

It wasn't until November of 1981 that the US amateurs got organized to develop a TNC that could be sold at a low cost. The organization formed was the Tucson Amateur Packet Radio Corporation (TAPR). Many dedicated engineers and programmers from around the country worked together in TAPR to design a low-cost TNC.

Other amateur groups, such as the Amateur Radio Research and Development Corporation (AMRAD), Amateur Radio Satellite Corporation (AMSAT) and the American Radio Relay League (ARRL) took part in packet radio's early development.

To get a feel for what packets are and how they are used, imagine a bus loaded with people. On the front of this bus is a sign displaying its destination. This bus is the packet and the people inside are data.

The bus originated at a bus station, say Buffalo, and its destination is Albany (see Figure 1). The bus only has a limited amount of fuel, so it must make stops at Rochester and Utica for

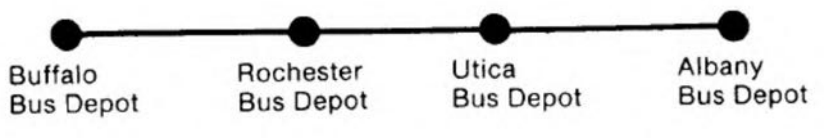


Figure 1: Packets are similar to buses on an interstate highway. In this example, a bus is traveling to Albany from Buffalo.

Len Popyack holds a bachelor's degree in electrical engineering and works as an engineer for General Electric in Syracuse, New York. Len enjoys developing assembly language programs for the CoCo. He is active in Amateur radio and his call is KA2NYJ.

refueling and to check that all the passengers arrived safely and comfortably.

The front of the bus is the packet header, the road is a communication channel and each bus station is a node (see Figure 2). Each packet originates at a node and may stop at a few nodes for error checking and re-transmission to the next node (similar to refueling the bus). The channel (road) is a radio link. This link connects each node to another. By transmitting packets from node to node, one can transfer data from Buffalo to Albany as easily as transferring data across town by conventional phone modems.

During the course of a digital communication, your TNC sends many packets of data along the path, from node to node, until the packets reach the destination. A packet contains the data you type on your CoCo, along with information that each TNC uses to decide where to send your data (see Figure 3). Additional information in each packet is used for error checking and other packet network information.

The nice thing is that your TNC takes care of all the "dirty work." You simply tell it who to establish a connection with and through which nodes.

A node is a TNC and a radio which is left on. Every packet station can act as a node, receiving and re-transmitting packets to other nodes. A single node can be used by many users at the same time. In other words, your node may be part of many different connections (see Figure 4). You may also be using your TNC to talk to another computer while someone uses your TNC as part of his connection path. The TNC takes care of it all. The connection placed through your node remains transparent to you.

Usually, amateurs tend to use a node with a wide coverage (i.e., capable of receiving and transmitting to a large area). These nodes generally consist of an amateur station with a TNC, a radio and sometimes a computer left on. These "super nodes" are referred to as digipeaters.

Because of the frequency of operation where packet communications takes place (145 MHz 2-meter band), the radio signals have a range of about "line of sight." This limitation dictates high-elevation digipeaters (for a greater line of sight). Most digipeaters are located on mountains or where the elevation of the terrain is the highest for a given area (see Figure 5). Digipeaters in my area (New York state) typically have ranges from about 50 to 100 miles.

One nice thing about Packet Radio is you can use up to eight nodes (digipeaters) to establish the communication path you want. These nodes don't have to be digipeaters per se, but may be local nodes.

Now that you know what a digipeater is (actually just another node), I'll throw a new twist into the digital communication network; gateways. A gateway is a means of access to another location other than the conventional node-to-node link. A gateway could be a high-speed link from the eastern U.S. to the west. It could also be a slow-speed link from east to west.

Think of a gateway as a node that looks to you as being in your area, but links you to a similar gateway at a location very far away (see Figure 6). The actual radio link between gateways can be a variety of communication types. The gateway could be a slow-speed (300 Baud), high-frequency link (Figure 7a), a high-speed, land-based

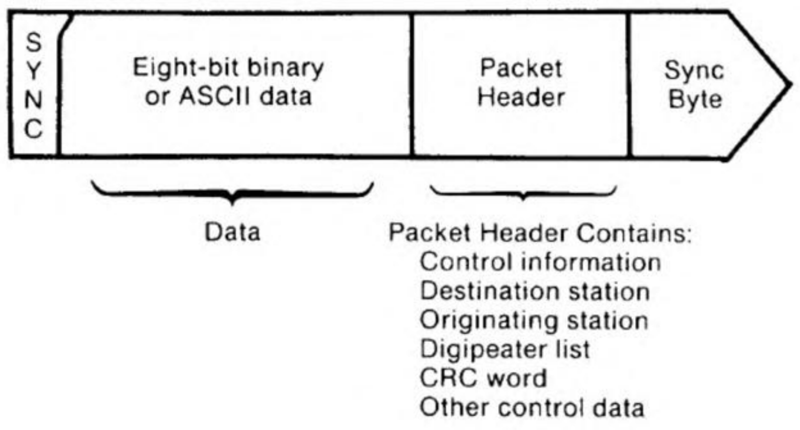


Figure 3: Expanded view of a packet

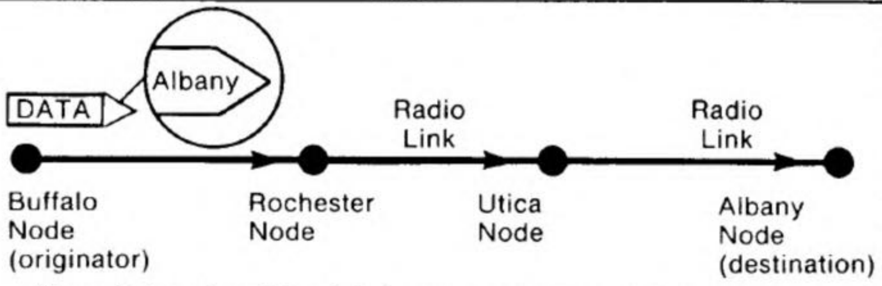


Figure 2: A packet of data is being transmitted from Buffalo to Albany via a node in Rochester and a node in Utica.

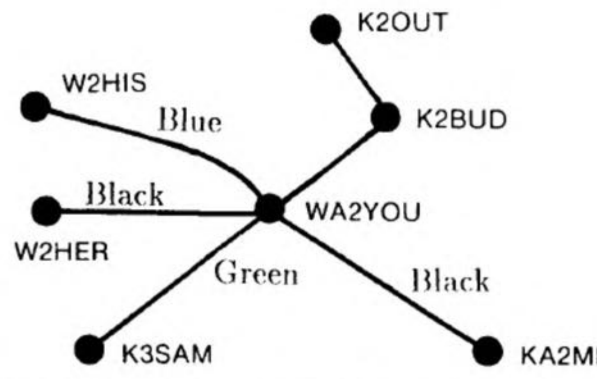


Figure 4: Your TNC may be used as part of other people's connection path. You may also use your TNC to communicate (blue) to another Ham while others use your TNC to rebroadcast their packets.

Tandy ELECTRONICS

Color Computer Fun for The Entire Family

Save $70

Reg **239.95**

229⁹⁵

TV NOT INCLUDED

● **Software Available for Games, Education & Home**

The ideal computer the whole family will enjoy using. Simply attach to your TV then plug in a Program Pak cartridge to entertain and educate your family for hours. Store a monthly budget or favourite recipes. Stylish, compact white case and low-profile electric typewriter quality keyboard. Add a cassette recorder, joysticks, printer, telephone modem, memory and more. With instruction manual. 26-3136

Value on Single-Sided Disks

Save $10

Reg **39.95**

29⁹⁵

Take advantage of this great saving! A set of ten single-sided 13.3cm double density, soft sector 40 track disks at less than $3 each! 26-406

Great Value!

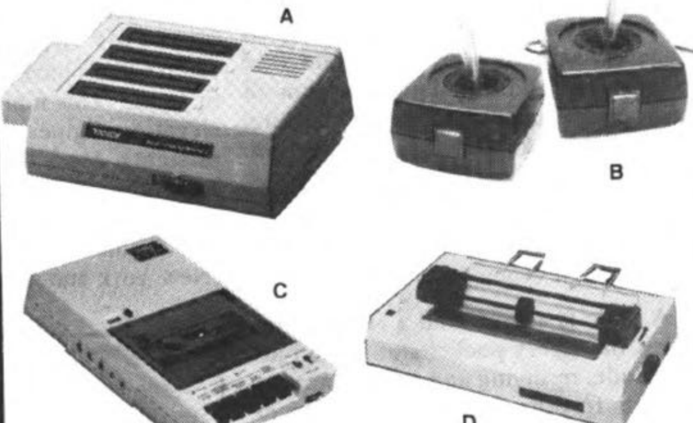
Save $50

Reg **129.95**

79⁹⁵

Tandy's touch pad makes controlling the cursor as simple as writing. Great for graphics. 26-1185

Great Value On Accessories



A. Multi-Pak Interface. Connects up to four program pak cartridges. Change between slots with selector switch or under program control. 26-3124 **179⁹⁵**

B. Joysticks. You can feel the difference! Fast 360° movement. Single shot button. 26-3008 **34⁹⁵**

C. Our most compact and lowest priced cassette recorder. Features volume control with preset marker. Includes cable (batteries not included). 26-1209 **79⁹⁵**

D. Save with our budget priced printer. Compatible with our color computer. Prints 80 upper and lower case characters per line at 43 lines per minute. 26-1276 **349⁹⁵**

Our Most Powerful Color Computer

INTRODUCTORY OFFER

Save $50

Reg **449.95** **399^{95}**

- **Connect to a High-Resolution Monitor or Your TV**

128K Extend BASIC Color Computer 3. Tandy's newest version of the famous color computer. Ideal for use in small business and home applications such as graphics, programming, budgets, word processing, data-base management, spreadsheet analysis and many more. Select up to 16 colors out of a pallet of 64. Uses 16 lines of 32 character text or 25 lines of 40 or 80 characters with a high resolution monitor. 26-3334

Tandy 1000 with Color Monitor

Reg **2598.00**

$1699

Save $899

- **Color Graphics Adapter Included**
- **The Hottest PC-Compatible Machine to Hit the Market**

Here's a new dimension for your imagination. The **Tandy 1000** linked with our new RGBI color monitor creates a level of color graphics that really needs to be seen to be fully appreciated. So now there's no reason to settle for a "games" computer because you can get a powerful, applications machine for a similar price. You can construct a full color graph of the months sales figures, draw up plans for that house extension, or use games software with graphics superior to any arcade machine. With Tandy 1000, the only limit is your imagination. 25-1000/25-1023

Tandy ELECTRONICS

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

Nearly 350 Stores Australia-Wide

WE SERVICE WHAT WE SELL!

Available From 350 Stores Australiawide Including Tandy Computer Centres

Order On VIATEL #642

Independent Tandy Dealers may not be participating in this ad or have every item advertised. Prices may also vary at individual Dealer Stores

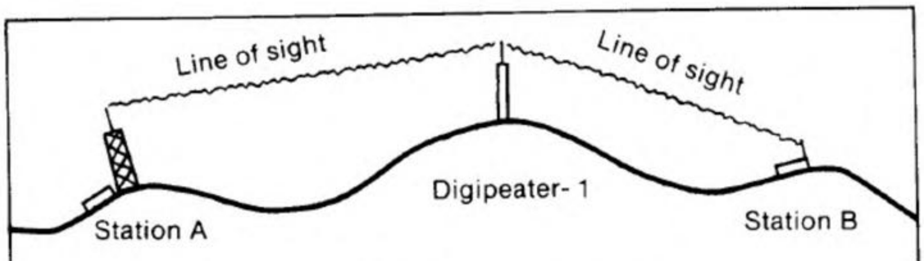


Figure 5: Because of the Earth's curvature, digipeaters are usually located at high locations. Station A can send packets to Station B using Digipeater-1.

communication channel (Figure 7b), or even a satellite link (Figure 7c).

As mentioned before, the Terminal Node Controller which you connect your CoCo to (via an RS-232 cable) is actually a small computer. The TNC has a microprocessor, RAM, ROM and I/O. The TNC consists of four functional parts: the TNC computer, the TNC 1200 Baud modem, the TNC-to-Color Computer interface and the TNC ROM software.

The TNC computer uses a small eight-bit microprocessor. The MJF-1200 uses a Z80. The Heathkit model HD-4040 uses a 6809. Whichever microprocessor is used, the TNC performs the same basic functions. The user interface is what really matters.

All TNCs have a 1200 Baud modem built into them. This modem produces an analog signal from your CoCo's digital signal and vice versa. The TNC-to-CoCo interface is usually made via an RS-232 link. Run your terminal program and the TNC looks just like your phone modem — almost.

When you connect to the TNC via the CoCo, you actually communicate with the TNC's computer. Most software built into the TNC computer follows the TARP standard. This standard is simply a set of commands used to tell the TNC to connect and disconnect to another node in the network. There are also commands to tell the TNC to perform a host of other tasks. Among them are telling the TNC to monitor all received packets, displaying and setting the time of day, sending beacon text, and far too many more to list here. For a good introduction to Packet Radio see Jim Grubbs' book, *Get *** CONNECTED to Packet Radio.*

There is as much available on Packet Radio as there are nodes to connect to. Packet Bulletin Board Services (PBBS) are popular. A typical PBBS covers a wide geographical area and allows message transfers from one PBBS to the next. If you could not connect to your buddy who is several states away, you might leave him an electronic mail message at his local PBBS. How does it get there? Simple. You connect to your local PBBS, leave him the message and direct the message to be sent to your friend's PBBS. During the early morning hours, your message will be transferred from PBBS to PBBS via the Packet Radio network!

I should mention that almost all the PBBSs run the same software (*WORLI*). This frees you from learning the ins and outs of a new PBBS everytime you connect to a different one. Some PBBSs have gateway functions that allow you to "hop" from one network to another (or, more appropriately, one frequency to another).

Where Is Packet Going From Here?

The new Japanese Amateur Satellite Number One (JAS-1), scheduled to be launched in August 1986, will allow packeters to transfer messages worldwide with only a 120 minute delay. JAS-1 will be a message store and forward system. JAS-1 will also allow packets to be sent in real time (ideal for chatting to friends around the globe).

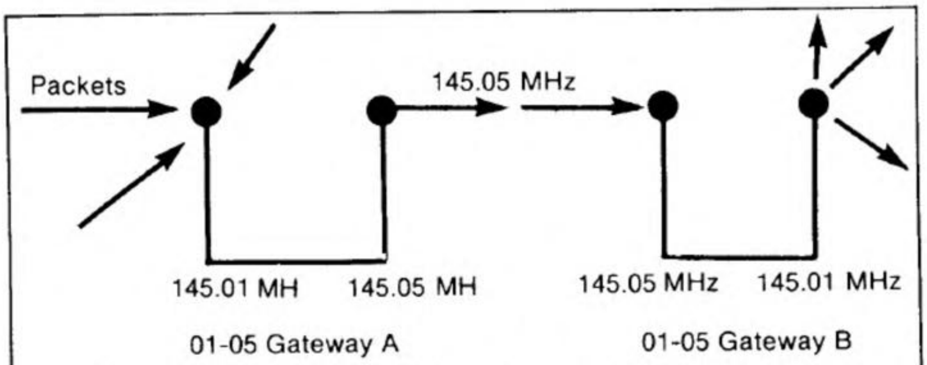


Figure 6: A gateway is two nodes at one location that pass packets from one frequency to another (or one network to another).

High-speed, land-based gateways will allow many local networks to communicate to distant networks easily. A public Packet Radio network has also been proposed (FCC rule making RM-5241). The future of packet radio is looking very active! If you are a licensed Ham (Technician class or higher), give it a try. If you are a would-be Ham, get out the code practice tapes and the CoCo Morse code simulator software and get going. You might try contacting a local Ham and ask him to give you a demonstration of Packet Radio. I'm sure that's all it will take to keep you awake nights . . . look at me!

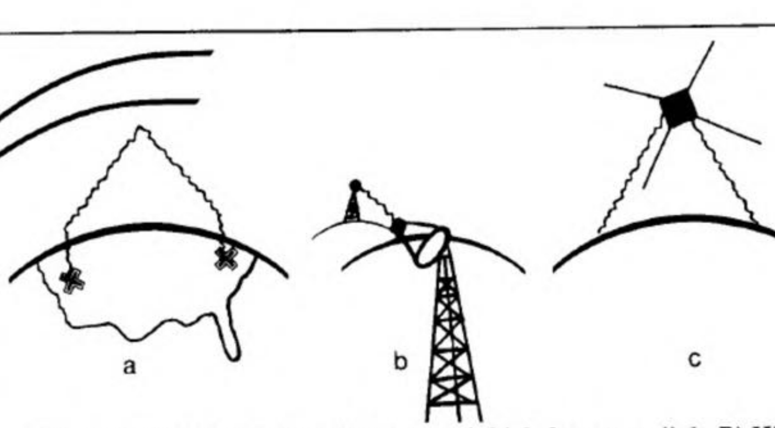


Figure 7: Gateway lines: A) Slow-speed, high-frequency link; B) High-speed microwave data link; C) Satellite data link.

DISK CRASHES Part 2

DISK RECOVERY—BLOOD SWEAT and TEARS SECTION B

by Ian G. Clarke

Here I am back again. With sweat and tears we have partially recovered our crashed disk (in theory only so far) by marking our sketched FAT boxes with "FF" for the granules which are unused or belong to killed files. We made sure (I hope) that the killed granules had not been overwritten by a live file. Now we come to the blood - recovery of live files .

We start with the files which the directory lists as using one granule only because they are easier. We know from the directory information which granule the file is on and how many bytes are in the last sector the file uses.

But which sector is the last used? You will recall from last time that the FAT stores the pointer to the last sector in the form "Cn" where n = the sector used. (Trap for young players - the DOS counts sectors starting with "1" not "0" as is the case with tracks and granules.) So we have a choice of nine sectors (1 granule=9 sectors... didn't you read the previous articles?)

We check sector by sector within the file's granule and look at the value of the last byte and the bytes directly before it. Confusion! How do we know whether the file ends in this sector? There was probably a previous file stored here and there is still data left from that file.

Easy! (My little attempt at humour; I found it far from easy.) We look for an EOF (End Of File) marker (probably the wrong technical term but you know what I mean). I am on shaky ground here as I am assuming EOF markers from looking at ends of files. I throw myself on the mercy of the more knowledgable members of the CoCo community.

The last byte of all ASCII files will be $0D whether they are data files or BASIC programs saved in ASCII format (e.g. SAVE progname A).

BASIC files saved normally are in binary or "crunched" format and have two bytes each of $00 at the end.

ML (Machine Language) files are also saved in binary format but finish with the two-byte EXEC address preceded by two bytes each of $00.

We check whether the bytes we are looking at fit any of the foregoing and if they do we jot down the sector number in our appropriate FAT box (remember to use the format "Cn").

Then on to the next one-granule file and so on until they have been completed. Our written FAT is filling fast and we are brimming with new-found confidence. Oh an optimist eh? We'll soon put a stop to that with the next operation.

Now for the hardest part. We look at the files which use more than one granule. The way I tackled it was to look at all granules which were shown in the directory as being the first file granules and jotted down key details as I looked at them e.g. recurring two-character variables.

It is also great if we find that a string is continued from the last sector of a granule into another granule. It provides a link.

We then look at all remaining granules with the same points in mind and with the list of "number of bytes in last sector" beside us we also check for possible final granules.

When we are sure we have all the granules for any given file, we jot down the number of the second granule in the FAT box belonging to the first granule.

Whew! Let me put that another way.

The computer reads the first granule number from the file directory information and then looks at that granule's value in the FAT. If the file uses more than one granule the value will be the number of the next granule in the file otherwise it will be the now familiar $Cn format. We jot these down as we go.

Still puzzled? Let's take a 3-granule file labelled TEST. The directory says the first granule is $0E. We look at that granule's value in the FAT and find it is $2D. We look at $2D in the FAT and find it is $35. We look at $35; it is $C6.

Therefore TEST uses granules $0E $2D and $35 in that order and the last sector in granule $35 is the 6th. OK?

We have finally filled in our boxes on our once-blank FAT. We use the zapper to change the FAT on the disk to these values. The disk has now been recovered.

Really? Are you sure? Let's make sure. DO A BACKUP NOW!! (And onto a different disk for those with double-sided drives.)

Test each file (should be simple enough as they will probably be mostly programs). Only after these tests do we rest on our laurels and breathe a sigh of relief.

Those masterpieces which we have toiled over for months have not been lost after all!

Dear CoCoNut, I have a confession and a dark secret. The confession - I was unable to recover a six-granule program. I obtained the first granule from the directory and found the EOF marker in another granule thus establishing the final granule. But I couldn't sort out the order of the middle four granules. I was about to try putting them in any order and then loading and listing the program to see what it looked like when a thought occurred to me. (Those who know me will realise the novelty of that experience.)

I would look at the FAT of the original disk (remember I had checked the FAT on the backup copy only). And now for the dark secret.

In the previous article I mentioned wasting six hours. The FAT I was now looking at only had about seven granule values corrupted; all the others were OK, including all of the program I couldn't sort

continued to page 38

32K
Disk

The Evolving REMOTE

A modification to use Remote2 with an RS-232 Pak

By Mark Crosby

I have changed Scott Taylor's *Remote2* improved remote terminal driver [November 1985, Page 106] to operate with a modified RS-232 Program Pak. By slightly modifying the RS-232 Pak, it can be adapted to work with a Y cable and disk drive.

Removing the ROM chip (U3) eliminates the memory conflict between Disk BASIC and the Pak ROM. Cutting the wire jumper connecting pads E1 and E2 separates the IRQ of the ACIA from positions 4 and 8 of the cartridge connector.

If the jumper is not cut, any data input causes the IRQ to become active, which triggers the non-maskable interrupt and the cartridge interrupt detection input of the computer. When the CART Pin is active, the computer resets. If you have the disk controller connected, the computer shows the opening credits.

After all the hardware modifications are done, it is just the exchange of routines to be able to communicate through the Pak.

The input routine could look like that in Figure 1.

The output routine could look like that in Figure 2.

These lines replace all programming required for the bit-banger method, and it is a lot faster too.

In addition to making the hardware change and using *Remot232* program or some other routine, you need to tell the 6551 ACIA what communication parameters you want to use. You can do this by poking certain values into the control register (location $FF6B) and the command register (location $FF6A). To configure the RS-232 Pak for 300 Baud, one stop bit, 7-bit word length, even parity and no echo, the following values need to be poked to the registers before loading *Remot232*:

```
POKE$HFF6B,54  (press ENTER)
POKE$HFF6A,107  (press ENTER)
```

For different settings, consult pages 15 and 16 of the manual that came with your RS-232 Pak.

One you have made all modifications and poked the control and command registers, LOADM and execute *Remot232*. Now you may control your CoCo via an external terminal, say a Model 100 or a full 80-column terminal.

I tested the modification by changing the RS-232 routines in the *Remote2* program. It worked well at 300 Baud and allowed the use of the printer. □

```
RCV     PSHS    B        save the contents of B
        LDB     $FF69    get status of ACIA
        ANDB    #$08     check for full receive register
        BEQ     NOIN     no input, jump out
        ANDB    #$07     check for receive errors
        BNE     ERR      jump to error section
        LDA     $FF68    load A with data input
        PULS    B        retrieve B
        RTS              go back to calling section
```

Figure 1

```
OUT     PSHS    B        save the contents of B
OUT1    LDB     $FF69    get status of ACIA
        ANDB    #$10     check for full transmit register
        BEQ     OUT1     if full, go back and check again
        STA     $FF68    transmit data
        PULS    B        retrieve B
        RTS              go back to calling section
```

Figure 2

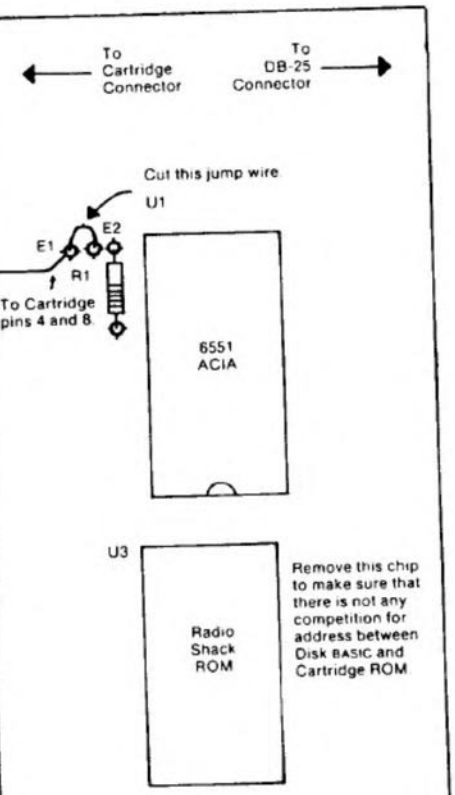

To Cartridge Connector
To DB-25 Connector
Cut this jump wire
U1
E1
E2
R1
To Cartridge pins 4 and 8
6551 ACIA
U3
Radio Shack ROM
Remove this chip to make sure that there is not any competition for address between Disk BASIC and Cartridge ROM

Figure 3

Mark Crosby, from Hymera, Ind., is a senior at Indiana State University majoring in applied computer technology. He has owned a CoCo for two years and is currently compiling information to construct an interactive space flight Simulation.

Editor's Note: *Remot232* will be included on RAINBOW ON TAPE as well as RAINBOW ON DISK. To transfer the file from tape to disk, type `CLOADM"REMOT232"` **and press ENTER. When the file has loaded from tape, enter** `SAVEM "REMOT232",&H7D00,&H7E E4,&H7D32`

The listing: REMOT232

```
                    ØØ1ØØ ***************************
                    ØØ11Ø *       REMOTE2           *
                    ØØ12Ø * AN IMPROVED REMOTE      *
                    ØØ13Ø *        TERMINAL         *
                    ØØ14Ø * DRIVER FOR THE COCO BY*
                    ØØ15Ø * SCOTT TAYLOR RAINBOW   *
                    ØØ16Ø *          11/85          *
                    ØØ17Ø * FROM THE ORIGINAL       *
                    ØØ18Ø * REMOTE PROGRAM BY       *
                    ØØ19Ø * DAN DOWNARD RAINDOW     *
                    ØØ2ØØ *          11/83          *
                    ØØ21Ø * SET TO OPERATE ON A     *
                    ØØ22Ø * MODIFIED RS232 PROGRAM*
                    ØØ23Ø * PAK BY MARK CROSBY      *
                    ØØ24Ø *           3/86          *
                    ØØ25Ø ***************************
7DØØ                ØØ26Ø           ORG     $7DØØ
                    ØØ27Ø *EQUATES FOR ROM & RAM
          Ø16A      ØØ28Ø IHOOK     EQU     $Ø16A
          Ø167      ØØ29Ø OHOOK     EQU     $Ø167
          AØØØ      ØØ3ØØ POLCAT    EQU     $AØØØ
          ØØ6F      ØØ31Ø DEV       EQU     $6F
          ØØ7Ø      ØØ32Ø FLAG      EQU     $7Ø
7DØØ      Ø1        ØØ33Ø BREAK     FCB     $Ø1
7DØ1      Ø1        ØØ34Ø CORNER    FCB     $Ø1
7DØ2      Ø1        ØØ35Ø LFFLG     FCB     $Ø1
7DØ3      ØØ        ØØ36Ø PRTFLG    FCB     $ØØ
7DØ4      ØØ        ØØ37Ø ICASE     FCB     $ØØ
7DØ5      ØØ        ØØ38Ø OCASE     FCB     $ØØ
7DØ6      9F        ØØ39Ø CURSOR    FCB     $9F
7DØ7      Ø4ØØ      ØØ4ØØ NOSCRL    FDB     $4ØØ
7DØ9      ØØ        ØØ41Ø COUNTR    FCB     $ØØ
7DØA                ØØ42Ø TABLE     RMB     $28
                    ØØ43Ø *INITIALIZE RAM HOOKS
7D32 BE   Ø168      ØØ44Ø START     LDX     1+OHOOK
7D35 AF   8D Ø12D   ØØ45Ø           STX     1+NTSCRN,PCR
7D39 BE   Ø16B      ØØ46Ø           LDX     1+IHOOK
7D3C AF   8D Ø193   ØØ47Ø           STX     1+IRET2,PCR
7D4Ø 86   7E        ØØ48Ø           LDA     #$7E
7D42 B7   Ø167      ØØ49Ø           STA     OHOOK
7D45 B7   Ø16A      ØØ5ØØ           STA     IHOOK
7D48 3Ø   8D ØØ48   ØØ51Ø           LEAX    OUT,PCR
7D4C BF   Ø168      ØØ52Ø           STX     1+OHOOK
7D4F 3Ø   8D Ø115   ØØ53Ø           LEAX    IN,PCR
7D53 BF   Ø16B      ØØ54Ø           STX     1+IHOOK
                    ØØ55Ø * MOVE TABLE OF ROM ADDRESS
                    ØØ56Ø *SET INKEY$ TO RAM BASED ROUTINE
7D56 8E   AA29      ØØ57Ø MOVTBL    LDX     #$AA29
7D59 31   8C AE     ØØ58Ø           LEAY    TABLE,PCR
7D5C 1ØBF Ø128      ØØ59Ø           STY     $128
7D6Ø EC   81        ØØ6ØØ GETADD    LDD     ,X++
7D62 ED   A1        ØØ61Ø           STD     ,Y++
7D64 8C   AA51      ØØ62Ø           CMPX    #$AA51
7D67 26   F7        ØØ63Ø           BNE     GETADD
7D69 3Ø   8D ØØØ5   ØØ64Ø           LEAX    INKEY,PCR
7D6D 31   3C        ØØ65Ø           LEAY    -4,Y
7D6F AF   A4        ØØ66Ø           STX     ,Y
7D71 39             ØØ67Ø RET       RTS
                    ØØ68Ø *CHECK KEYBOARD AND
                    ØØ69Ø *RS232 FOR INKEY$ 255
                    ØØ7ØØ *TIMES IF NOTHING IN $87
7D72 96   87        ØØ71Ø INKEY     LDA     <$87
7D74 26   15        ØØ72Ø           BNE     YES
7D76 86   FF        ØØ73Ø           LDA     #$FF
7D78 B7   7DØ9      ØØ74Ø           STA     COUNTR
7D7B AD   9F AØØØ   ØØ75Ø INKEY2    JSR     [POLCAT]
7D7F 26   ØA        ØØ76Ø           BNE     YES
7D81 BD   7E88      ØØ77Ø           JSR     REMIN
7D84 26   Ø5        ØØ78Ø           BNE     YES
7D86 7A   7DØ9      ØØ79Ø           DEC     COUNTR
7D89 26   FØ        ØØ8ØØ           BNE     INKEY2
                    ØØ81Ø *CHARACTER IN A REGISTER
                    ØØ82Ø *BRANCH IF BREAK (A=$Ø3)
                    ØØ83Ø *CONVERT TO STRING
7D8B 81   Ø3        ØØ84Ø YES       CMPA    #$Ø3
7D8D 1Ø27 3Ø78      ØØ85Ø           LBEQ    $AEØ9
7D91 7E   A56B      ØØ86Ø           JMP     $A56B
                    ØØ87Ø *OUTPUT CHARACTER IF DEV=Ø
                    ØØ88Ø *INSERT LINE FEEDS IF NEEDED
                    ØØ89Ø *USE NEW PRINT ROUTINE
7D94 34   Ø6        ØØ9ØØ OUT       PSHS    A,B
7D96 ØD   6F        ØØ91Ø           TST     <DEV
7D98 1Ø26 ØØC7      ØØ92Ø           LBNE    NTSCN1
7D9C 81   Ø8        ØØ93Ø           CMPA    #$Ø8
7D9E 27   1A        ØØ94Ø           BEQ     RMOUT2
7DAØ 81   ØD        ØØ95Ø           CMPA    #$ØD
7DA2 26   ØE        ØØ96Ø           BNE     REMOUT
                    ØØ97Ø *CHECK IF LINE FEEDS ARE
                    ØØ98Ø *TO BE SENT TO REMO-TERM
7DA4 F6   7DØ2      ØØ99Ø           LDB     LFFLG
7DA7 27   11        Ø1ØØØ           BEQ     RMOUT2
7DA9 86   ØA        Ø1Ø1Ø           LDA     #$ØA
7DAB BD   7ED5      Ø1Ø2Ø           JSR     RSOUT
7DAE 86   ØD        Ø1Ø3Ø           LDA     #$ØD
7DBØ 2Ø   Ø8        Ø1Ø4Ø           BRA     RMOUT2
                    Ø1Ø5Ø *ECHO CHARACTER IN PRTFLG
                    Ø1Ø6Ø *IF IT IS NOT - TO Ø
7DB2 F6   7DØ3      Ø1Ø7Ø REMOUT    LDB     PRTFLG
7DB5 27   Ø3        Ø1Ø8Ø           BEQ     RMOUT2
7DB7 B6   7DØ3      Ø1Ø9Ø           LDA     PRTFLG
                    Ø11ØØ *CHECK OUTPUT FOR UPPER-C
                    Ø111Ø *OR LOWER-C AND SEND IT
7DBA F6   7DØ5      Ø112Ø RMOUT2    LDB     OCASE
7DBD BD   7EAC      Ø113Ø           JSR     CKCASE
7DCØ BD   7ED5      Ø114Ø           JSR     RSOUT
7DC3 35   Ø6        Ø115Ø ORET1     PULS    A,B
7DC5 34   34        Ø116Ø ORET2     PSHS    B,X,Y
                    Ø117Ø *CHECK FOR BREAK KEY
7DC7 F6   Ø154      Ø118Ø           LDB     $154
7DCA C1   BF        Ø119Ø           CMPB    #$BF
7DCC 27   2C        Ø12ØØ           BEQ     PULL
7DCE 9E   88        Ø121Ø           LDX     $88
7DDØ 81   Ø8        Ø122Ø           CMPA    #$Ø8
7DD2 27   38        Ø123Ø           BEQ     BKSPC
7DD4 81   ØD        Ø124Ø           CMPA    #$ØD
7DD6 27   67        Ø125Ø           BEQ     ENTER
7DD8 81   2Ø        Ø126Ø           CMPA    #$2Ø
7DDA 25   25        Ø127Ø           BLO     PULL2
7DDC 81   1F        Ø128Ø           CMPA    #$1F
7DDE 22   Ø4        Ø129Ø           BHI     CMP2
7DEØ 86   6Ø        Ø13ØØ           LDA     #$6Ø
7DE2 2Ø   2Ø        Ø131Ø           BRA     PUTIT
                    Ø132Ø *TRANSLATE ASCII VALUE TO
                    Ø133Ø *SCREEN CHAR VALUE
7DE4 81   3F        Ø134Ø CMP2      CMPA    #$3F
7DE6 22   Ø4        Ø135Ø           BHI     CMP3
7DE8 8B   4Ø        Ø136Ø           ADDA    #$4Ø
7DEA 2Ø   18        Ø137Ø           BRA     PUTIT
7DEC 81   5F        Ø138Ø CMP3      CMPA    #$5F
7DEE 22   Ø2        Ø139Ø           BHI     CMP4
7DFØ 2Ø   12        Ø14ØØ           BRA     PUTIT
7DF2 81   7F        Ø141Ø CMP4      CMPA    #$7F
7DF4 22   ØE        Ø142Ø           BHI     PUTIT
7DF6 8Ø   6Ø        Ø143Ø           SUBA    #$6Ø
7DF8 2Ø   ØA        Ø144Ø           BRA     PUTIT
7DFA 8C   Ø6ØØ      Ø145Ø PULL      CMPX    #$6ØØ
7DFD 26   Ø2        Ø146Ø           BNE     PULL2
7DFF 8D   18        Ø147Ø           BSR     SCROLL
```

```
7EØ1 35    34         Ø148Ø PULL2   PULS   B,X,Y
7EØ3 39               Ø149Ø         RTS
7EØ4 A7    8Ø         Ø15ØØ PUTIT   STA    ,X+
7EØ6 9F    88         Ø151Ø FIXCUR  STX    $88
7EØ8 86    ØA         Ø152Ø         LDA    #$ØA
7EØA 2Ø    EE         Ø153Ø         BRA    PULL
7EØC 1Ø8E 6Ø6Ø        Ø154Ø BKSPC   LDY    #$6Ø6Ø
7E1Ø 1ØAF 82          Ø155Ø         STY    ,-X
7E13 9F    88         Ø156Ø         STX    $88
7E15 86    ØA         Ø157Ø         LDA    #$ØA
7E17 2Ø    E8         Ø158Ø         BRA    PULL2
                      Ø159Ø *CHECK IF SCROLL PROTECTED
                      Ø16ØØ *AREA IS IN RANGE OF $4ØØ
                      Ø161Ø *TO $5EØ, SCROLL SCREEN
7E19 BE    7DØ7       Ø162Ø SCROLL  LDX    NOSCRL
7E1C 8C    Ø4ØØ       Ø163Ø         CMPX   #$4ØØ
7E1F 25    Ø5         Ø164Ø         BLO    MAKEX
7E21 8C    Ø5EØ       Ø165Ø         CMPX   #$5EØ
7E24 25    Ø3         Ø166Ø         BLO    SCROL2
7E26 8E    Ø4ØØ       Ø167Ø MAKEX   LDX    #$4ØØ
7E29 E6    88 2Ø      Ø168Ø SCROL2  LDB    $2Ø,X
7E2C E7    8Ø         Ø169Ø         STB    ,X+
7E2E 8C    Ø5EØ       Ø17ØØ         CMPX   #$5EØ
7E31 26    F6         Ø171Ø         BNE    SCROL2
7E33 9F    88         Ø172Ø         STX    $88
7E35 C6    6Ø         Ø173Ø         LDB    #$6Ø
7E37 E7    8Ø         Ø174Ø LASTLN  STB    ,X+
7E39 8C    Ø6ØØ       Ø175Ø         CMPX   #$6ØØ
7E3C 26    F9         Ø176Ø         BNE    LASTLN
7E3E 39               Ø177Ø         RTS
                      Ø178Ø *CLEAR TO END OF LINE IF
                      Ø179Ø *'ENTER' KEY IS PRESSED
7E3F 96    89         Ø18ØØ ENTER   LDA    $89
7E41 81    2Ø         Ø181Ø COMP    CMPA   #$2Ø
7E43 25    Ø4         Ø182Ø         BLO    ENT
7E45 8Ø    2Ø         Ø183Ø         SUBA   #$2Ø
7E47 2Ø    F8         Ø184Ø         BRA    COMP
7E49 C6    6Ø         Ø185Ø ENT     LDB    #$6Ø
7E4B E7    8Ø         Ø186Ø         STB    ,X+
7E4D 8C    Ø6ØØ       Ø187Ø         CMPX   #$6ØØ
7E5Ø 26    Ø4         Ø188Ø         BNE    INCRE
7E52 8D    C5         Ø189Ø         BSR    SCROLL
7E54 2Ø    Ø7         Ø19ØØ         BRA    FINISH
7E56 4C               Ø191Ø INCRE   INCA
7E57 81    2Ø         Ø192Ø         CMPA   #$2Ø
7E59 26    EE         Ø193Ø         BNE    ENT
7E5B 9F    88         Ø194Ø         STX    $88
7E5D 35    34         Ø195Ø FINISH  PULS   B,X,Y
7E5F CC    ØAØ1       Ø196Ø         LDD    #$ØAØ1
7E62 39               Ø197Ø         RTS
7E63 35    Ø6         Ø198Ø NTSCN1  PULS   A,B
7E65 7E    7DC5       Ø199Ø NTSCRN  JMP    ORET2
                      Ø2ØØØ *INPUT FROM KEYBOARD OR
                      Ø2Ø1Ø *RS232 IF DEV-Ø
                      Ø2Ø2Ø *USE RSIN FOR REMOTE INPUT
7E68 B6    7DØ6       Ø2Ø3Ø IN      LDA    CURSOR
7E6B A7    9F ØØ88    Ø2Ø4Ø         STA    [$88]
7E6F ØF    7Ø         Ø2Ø5Ø         CLR    <FLAG
7E71 ØD    6F         Ø2Ø6Ø         TST    <DEV
7E73 26    5D         Ø2Ø7Ø         BNE    IRET2
7E75 32    62         Ø2Ø8Ø         LEAS   2,S
7E77 34    15         Ø2Ø9Ø         PSHS   B,CC,X
7E79 AD    9F AØØØ    Ø21ØØ IN1     JSR    [POLCAT]
```

```
7E7D 27    Ø2         Ø211Ø         BEQ    RSCHK
7E7F 2Ø    4F         Ø212Ø         BRA    NOCHNG
7E81 BD    7E88       Ø213Ø RSCHK   JSR    REMIN
7E84 27    F3         Ø214Ø         BEQ    IN1
7E86 2Ø    42         Ø215Ø         BRA    IRET1
7E88 F6    FF69       Ø216Ø REMIN   LDB    $FF69
7E8B C4    Ø8         Ø217Ø         ANDB   #$Ø8
7E8D 27    18         Ø218Ø         BEQ    ZEROA
                      Ø219Ø *NEW RS232 INPUT FROM CART
7E8F C4    Ø7         Ø22ØØ RSAN    ANDB   #$Ø7
7E91 26    14         Ø221Ø         BNE    ZEROA
7E93 B6    FF68       Ø222Ø         LDA    $FF68
                      Ø223Ø *STORE CHAR IN UPPER RIGHT
                      Ø224Ø *CORNER OF SCREEN
7E96 F6    7DØ1       Ø225Ø         LDB    CORNER
7E99 27    Ø3         Ø226Ø         BEQ    CHKBRK
7E9B B7    Ø41F       Ø227Ø         STA    $41F
                      Ø228Ø *CHECK FOR BREAK DISABLE
                      Ø229Ø *AND FOR BREAK SIGNAL
                      Ø23ØØ *FROM REMOTE TERMINAL
7E9E F6    7DØØ       Ø231Ø CHKBRK  LDB    BREAK
7EA1 27    Ø6         Ø232Ø         BEQ    NOTBRK
7EA3 81    Ø3         Ø233Ø         CMPA   #$Ø3
7EA5 26    Ø2         Ø234Ø         BNE    NOTBRK
7EA7 4F               Ø235Ø ZEROA   CLRA
7EA8 39               Ø236Ø         RTS
7EA9 F6    7DØ4       Ø237Ø NOTBRK  LDB    ICASE
                      Ø238Ø *CHECK IF CHAR SHOULD BE
                      Ø239Ø *UPPER- OR LOWERCASE AND
                      Ø24ØØ *CHANGE IT ACCORDINGLY
7EAC C1    Ø1         Ø241Ø CKCASE  CMPB   #$Ø1
7EAE 26    ØB         Ø242Ø         BNE    CMPB2
7EBØ 81    61         Ø243Ø         CMPA   #$61
7EB2 25    15         Ø244Ø         BLO    RET2
7EB4 81    7A         Ø245Ø         CMPA   #$7A
7EB6 22    11         Ø246Ø         BHI    RET2
7EB8 8Ø    2Ø         Ø247Ø         SUBA   #$2Ø
7EBA 39               Ø248Ø         RTS
7EBB C1    Ø2         Ø249Ø CMPB2   CMPB   #$Ø2
7EBD 26    ØA         Ø25ØØ         BNE    RET2
7EBF 81    41         Ø251Ø         CMPA   #$41
7EC1 25    Ø6         Ø252Ø         BLO    RET2
7EC3 81    5A         Ø253Ø         CMPA   #$5A
7EC5 22    Ø2         Ø254Ø         BHI    RET2
7EC7 8B    2Ø         Ø255Ø         ADDA   #$2Ø
7EC9 39               Ø256Ø RET2    RTS
7ECA C6    6Ø         Ø257Ø IRET1   LDB    #$6Ø
7ECC E7    9F ØØ88    Ø258Ø         STB    [$88]
7EDØ 35    95         Ø259Ø NOCHNG  PULS   B,CC,X,PC
7ED2 7E    7D71       Ø26ØØ IRET2   JMP    RET
7ED5 34    Ø4         Ø261Ø RSOUT   PSHS   B
7ED7 F6    FF69       Ø262Ø RSOUT1  LDB    $FF69
7EDA C4    1Ø         Ø263Ø         ANDB   #$1Ø
7EDC 27    F9         Ø264Ø         BEQ    RSOUT1
7EDE B7    FF68       Ø265Ø         STA    $FF68
7EE1 35    Ø4         Ø266Ø         PULS   B
7EE3 39               Ø267Ø         RTS
                      Ø268Ø *
7EE4                  Ø269Ø ZZZZ    ********
           7D32       Ø27ØØ         END    START

ØØØØØ TOTAL ERRORS
```

DISK CRASHES Part 2

continued from page 35

out.

I exclaimed "Golly Gosh" or words to that effect and dropped to the floor in the standard computer hacker's head-pounding position. All that time! But I did learn didn't I. Did I? Time will tell.

In closing I would like to stress that all information used in this series of articles has come from my gleanings of books and articles and from logical (well that's what I call it) deduction.

If you have detected errors of fact in these articles please write in to this illustrious publication so that all interested readers (particularly me) can benefit.

I am now going back to my semigraphics-8 programming... maybe I never do learn after all. But I am going to clear sufficient memory this time and I'm not going to do backups the lazy way. Bye!

(And thanks Ian for an exceptionally well written series. G.)

CoCo Conf'86 Tutorial Notes

GAME OF LIFE

Research Notes

by Bob Delbourgo

GRAHAM ASKED me to talk to you about the role of computers in education and, in a roundabout way, I will do just that; however let me bend the meaning of education and tell you about the uses of computing for scientific research -- really nothing more than a form of continuing education. I propose to tell you how I have benefitted from the use of a CoCo and a Model 100 in my job.

When we originally bought a 32K CoCo in 1982, I should confess that I knew nothing about computing and bought the CoCo mainly for the sake of Tino and Daniel, who, I thought, would have to confront the beasts in later life. Then I gradually became interested in the machine myself and soon realised its potential for my own research.

It wasn't long before we upgraded to 64K. Then after acquiring a Model 100 in 1984 (which is better at maths than CoCo) I found myself doing exciting new work in areas I would never have dreamed of touching ten years ago; and I continue to explore those areas these days. There is a message here: something which was bought as a "getting to know you" exercise can lead to plenty of other uses if you've got an inquisitive mind and a certain dogged persistence.

Our humble CoCo is one of the most versatile, reasonably-priced micros around and you should make the most of it. Just because you don't have all the latest accessories or because you don't possess the latest model and the fanciest software, don't lose interest in it! But, back to research...

When you open a scientific journal today you can't fail to notice that about one third of the published papers rely on computing in some form or another. Fifteen years ago the proportion was virtually nil. The reason is that micros were simply not around then.

When I went to university there was no formal training in computing. The younger generation here may find it difficult to believe, yet it's quite true and it hasn't prevented working scientists like myself from picking up the strands late in life and putting them to some use. Of course there are hundreds of applications of computers in research and obviously I cannot cover them all. What I intend to do is focus on two topics which I know a little bit about, which are relatively easy to understand and which I can program on the CoCo. These are CELLULAR AUTOMATA and CHAOS.

Let me begin with a "game" that some of you may have already come across, Conway's GAME OF LIFE. Incidentally, games are a legitimate way of learning new things. Don't feel sinful, ashamed or silly playing one of the many "computer games" on the market. There is often a lot behind them and if you analyse how the programs work you will learn much more than simply typing in the latest data into your personal files.

Anyway, returning to the game of life, many of you who have "played" it are likely to have only maintained a passing interest in it. In fact, Conway's invention is an example of cellular automata, not the simplest one, but perhaps the most popular. It involves a great deal of fascinating mathematical physics. You start at some instant with a two-dimensional "live" array of cells marked by little squares on your video screen. To see what happens at the next instant apply the following rules:

A cell is stable if it has 2 or 3 neighbours. A cell dies if it has fewer than 2 neighbours or more than 3 neighbours. An empty cell gives birth to a live one if it has 3 neighbours.

Given these rules of replication one can follow through the fate of successive generations of cells. But if you tried to work this out by straight brain-power, one cell at a time, it would take many days or even weeks of painstaking analysis; the computer can handle the problem in seconds especially when programmed in machine code.

I've brought along a SPECTRAL ASSOCIATES version of GAMELIFE where one can experiment at will. Observe how quickly CoCo runs through successive time frames... Also notice that the rules often lead to the total death of the cellular array; occasionally they lead to various stable configurations, to oscillating shapes or sometimes to crawling figures.

There are two important points I'd like to stress: (1) Very simple rules can lead to unimagined complexity. (2) The process is "irreversible" in that final states can arise from many initial states; in other words the same end can come from different beginnings or "you cannot go backwards in time".

Of course, you need nothing fancier than a Color Computer to be able to check all this for yourselves. Indeed let us simplify the problem and change the rules.

Instead of a two-dimensional array take a one-dimensional one and string the cells along a line (we can choose how many we want and their locations). Invent rules of birth and death so that they depend only on the state of the nearest neighbours. It turns out that, apart from six cases, the generation rules give uninteresting results.

The six rules can be formulated as follows: let A(i) = 1 or 0 represent whether cell i is alive or dead the previous generation and let B(i) stand for the condition of the i'th cell in the new generation. Then the relation between B(i) and the earlier A(i) is given by

```
 (1) B(I) = A(I-1) + A(I+1) MOD 2
 (2) B(I) = A(I-1) + A(I) + A(I+1) MOD 2
 (3) B(I) = A(I-1) + A(I) + A(I+1) + A(I-1)*A(I+1)*
    (1 + A(I)) MOD 2
 (4) B(I) = A(I-1) + A(I) + A(I+1) + A(I-1)*A(I)*
    A(I+1) MOD 2
 (5) B(I) = A(I-1) + A(I) + A(I+1) + A(I)*A(I+1) +
    A(I-1)*A(I+1) + A(I-1)*A(I) MOD 2
 (6) B(I) = (A(I-1) + A(I+1))*(A(I) + 1) + A(I-1)
    *A(I)*A(I+1) MOD 2
```

Remember that X MOD 2 signifies X - 2*INT(X/2) so the result for B(i) is always 0 or 1. Here is a little program on CELLULAR AUTOMATA, which Tino helped me program in ML, that shows you the development of cells by drawing successive generations one line at a time and for each of the six interesting rules.

Some of the shapes are quite fascinating and often repeat themselves. Again, the point is the same: simple rules produce complicated organisation and the system cannot be reversed in time. Cellular automata is a hot scientific topic at the moment and there are many eminent people studying variants of these objects in the hope that this will explain how systems evolve and whether this leads to more or less order.

Human beings or robots are regarded as examples of automata; they organise their environment by creating order nearby at the expense of creating more disorder overall -- in accordance with the laws of thermodynamics.

I'd like now to turn to the other topic, CHAOS. The subject was in a very confused state as recently as ten years ago -- I say recently because 10 years is not a long time in science. Then, suddenly, in the last decade there was a dramatic change in our understanding of how "chaos" or "turbulence" or "erratic behaviour" comes about.

I'd like to spend some time explaining the basic ideas to you, because they are not especially difficult. The exciting thing about the concepts is that they apply universally to all systems as we shall see!

Let's start with a simple model. We have a jar consisting of bacteria in a soup of nutrient. Let X stand for the proportion of the jar occupied by the bacteria (the rest being nutrient) at the beginning of a life cycle. Obviously X lies between 0 and 1.

Now during the next life cycle the bacteria will reproduce at a certain birth rate which we can describe by a variable L. The more bacteria the greater the progeny.

On the other hand, if there are too many bacteria there will not be sufficient food for them all and some will die. To take crude account of these conditions let us model the population of bacteria by the function

```
 L*X*(1 - X)    (with L=<4 to ensure that X does
   not exceed 1).
```

This formula tells you that if X is the population on some day, then the next day the population will be L*X*(1-X). That's simple enough.

The system is totally "deterministic" in that we can predict the next value of X if we know the starting value of X exactly and the birth rate L.

And, by the way, that is the way that all large natural systems go: they are simple and deterministic. However, please notice that two values of X can lead to the same population on the following day, so the process is again irreversible (there is no going back!)

The question is "What will the population X tend to after a long time?" Well let's set up a little program to find out:

```
 5 'Demonstration of emergence of chaos for selected
  inputs for the function X*L*(1-X)
 10 CLS:INPUT"enter birth rate L (>0 & <4)";L:IF
  L>4ORL<0THEN10
 20 INPUT"starting population X (>0 & <1)";X:
  IFX<0ORX>1THEN20
 30 C=1:PRINT
 40 PRINT"day";C;"population";X
 50 C=C+1:X=L*X*(1-X)
 60 IFINKEY$=""THEN40
 70 RUN
```

Before we RUN it let's make an educated guess at what will happen eventually. If the population does settle down to a final X-value it can only be when

```
  X = L*X*(1 - X) when X = 0 or X = 1 - 1/L.
```

Let's try RUNning now. Begin with the inputs L=.5 and X=.3 say. A few "iterations" or loops show that X does indeed settle down to 0; in other words the bacteria die out. This is not too surprising because the birth rate parameter L is small. L is called a "control parameter" of the system and we can study how the population depends on it.

Try inputting a higher birth rate L=2 with X=0.3. Sure enough the population settles down to a final X = 0.5 = 1 - 1/2. That's fine as it agrees with our algebra above.

Now increase the birth rate input L to 3.2 and start with X=.3. Here's a surprise! The final X settle down to two successive values, or a "two-cycle"; one day X=.799 , the next X=.513 , and then back again to .799 and so on. Quite astonishing for such a simple model! (The technical term is that the settled X "bifurcates" from the lower L case).

Now try L=3.5 and X=0.3; after a bit we find that X settles down to a succession of four values, .501, .875, .383, .827 ; in other words the system bifurcates again. It keeps on bifurcating until L reaches 3.7 at which point "chaos" has fully developed. What I mean here is that successive populations seem to bear no relation to one another!

Another feature of chaos is that if we begin with two population values that are very nearly the same, the final values can be widely different. "There is extreme sensitivity to the initial conditions".

Now let's go beyond the onset of bifurcation chaos and take L=3.75 . Again we see no rhyme and reason in the X's. Chaos reigns. But here's another surprise; try L=3.83.

You now find that the eventual population cycles between three values, .156, .505 and .957. We say that a "three-cycle window has been born". However it soon disappears as we increase L; already at L=3.86 it has gone. All of these properties (from L=2 to 4) can be displayed on a single diagram, (Figure 1) via this program

```
 5 'Demonstration of development of chaos as control
  parameter L is varied for the nonlinear map
  X->L*X*(1-X)
 10 CLS:PMODE4,1:PCLS:SCREEN1,0:LINE(0,0)-(255,191),
  PSET,B:'high-res screen
```

```
20 FORH=1TO254:L=2+2*H/254:X=.5
30 FORI=1TO50:X=L*X*(1-X):NEXTI:'get X to settle
 down after some 50 iterations
35 IFL<3.4THENFORI=1TO4:X=L*X*(1-X):V=(1-X)*191:
 PSET(H,V):NEXTI:GOTO50:'no need to iterate X much
 thereafter if L-values are prechaotic
40 FORI=1TO200:X=L*X*(1-X):V=(1-X)*191:
 PSET(H,V):NEXTI:'iterate X some 200 times near and
 after the chaotic range
50 PLAY"L255T255C":NEXTH
60 'your own screen dump routine here
70 GOTO70
```

In the screen printout (Figure 1) you can pick out the single fixed point X=1-1/L from L=2 to L=3 and then the ensuing bifurcations from L=3 to L=3.7. Then you notice chaos, broken by a few windows of stability, like the 3-cycle, which is the most noticeable of them.

The exciting thing about the whole affair is that one can prove that ANY similar model (functions of X like L*X*(1-X) with a single hump in the allowed region of X) has exactly the same behaviour. The order of events as chaos develops and thereafter is identical in all the models.

The other exciting thing is that the shrink rates of fork sizes and fork widths is also independent of the detailed model. They are "universal features" and "universal constants" of deterministic chaos.

The most telling thing is that these features have been observed in natural phenomena. A very wide variety of nonlinear, irreversible and therefore turbulent systems show up the period doubling process (and shrinkage rates) as the control parameter L is varied.

Discoveries in convection of fluids, in chemical reactions, in laser physics, in electronic circuits, in medicine (neurological processes such as heart rates, epilepsy?,..), in population dynamics, etc. have supported this general view of chaos. Someone has even dared to apply the ideas to the occurrence of wars, though his analysis is rather naive and unbelievable.

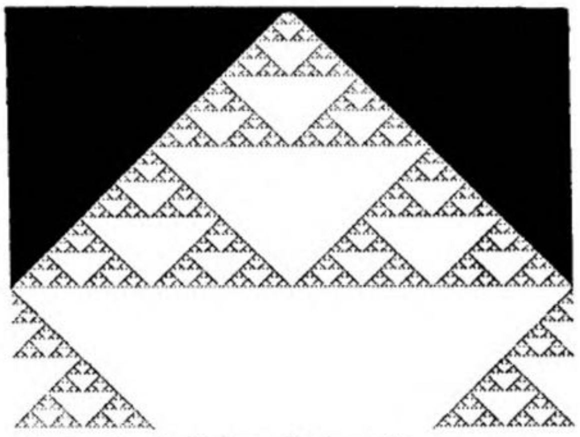
But it only goes to show you how seriously the concepts are being taken. My own work in this area -- helped no end by CoCo -- has concentrated on finding the relation between the fork length and fork width shrinkage rates, which up to now have been assumed to be independent.

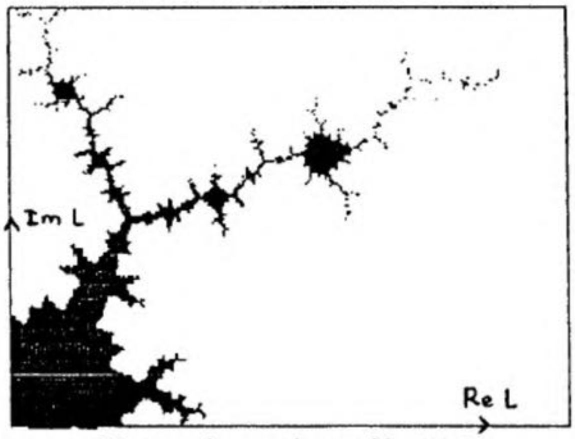
One can extend the iteration of a real variable X to the iteration of two real variables X and Y (along the horizontal and along the vertical say). One way is to unite X and Y into a complex variable Z=X+iY and study the iterations of Z through the complex function L*Z*(1-Z).

This is a fascinating subject with even more striking imagery (Figures 2 to 6). Let me end by showing you a few computer generated figures on "complex chaos" which I have saved on disk -- if you are interested in what they mean, please hang around and I'll explain them to you.

It's a good place to stop this "tutorial" and I do hope that I have stimulated your interest in the sort of things you can explore with a CoCo. I have some reprints of scientific articles of a general nature which some of you might care to glance at, if these topics have fired your imagination.

GAME OF LIFE


Rule 3. Cellular Automata



Filaments and cacti near
L = 2.7 + 1.1 i

The Listing:

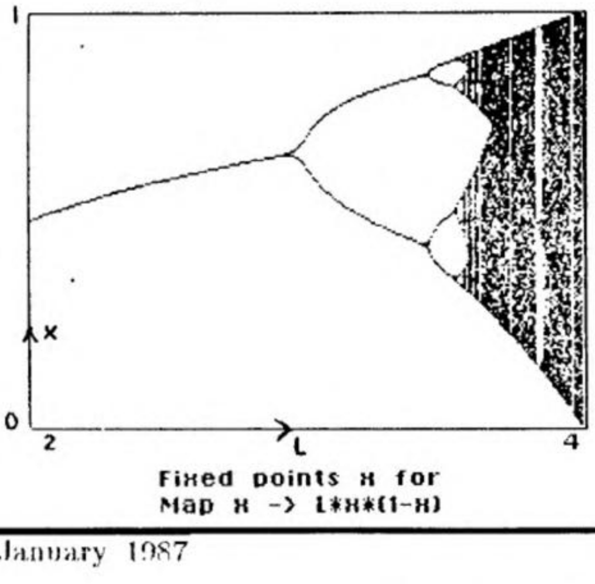
```
0 GOTO10
1 '******* GAME OF LIFE ********
3 SAVE"GAMELIFE:3":END
10 CLS
20 FOR I= 22841TO 24617
30 READ Q$:POKEI,VAL("&H"+Q$):NE
XTI
40 EXEC 24576
50 DATA 86,8,B7,FF,22,B7,FF,C0,B
7,FF,C2,B7,FF,C4,8E,3B,A,CE,4,0,
A6,80,27,4,A7,C0,20,F8,86,7,8E,0
,0,30,1F,26,FC,4A,26,F6,86,88,B7
,FF,22,B7,FF,C1,F,C0,F,C7,8E,8,2
0,9F,C4,8E,4,0,9F,C2,86,0,A7,80,
8C,8,0,26,F9,8E,A,0,6F
60 DATA 80,8C,1A,0,26,F9,20,59,D
C,C4,F,C1,44,56,6,C1,44,56,9,C1,
9,C1,D3,C2,DE,C0,1F,2,39,8D,2F,D
C,C4,83,0,40,25,2,DD,C4,8D,6,20,
35,8D,20,20,31,D,C7,26,19,C,C7,8
D,D0,A6,C9,3A,FA,43,A4,A4,97,C6,
A6,C9,3A,FA,A4,A4,AA,C9,3A,FE
70 DATA A7,A4,39,D,C7,27,FB,F,C7
,8D,B2,A6,C9,3A,FA,A4,A4,9A,C6,A
7,A4,39,AD,9F,A0,0,27,FA,81,43,2
7,BD,81,58,27,BD,81,45,10,27,FF,
72,81,5E,27,A4,81,A,27,1A,81,8,2
7,38,81,9,27,23,81,49,27,3D,81,4
4,27,50,81,47,27,6C,81,4B,27,5D,
20
80 DATA CA,8D,B5,DC,C4,10,83,F,C
0,24,87,C3,0,40,DD,C4,20,80,8D,A
4,DC,C4,10,83,F,FF,24,F4,C3,0,1,
DD,C4,20,ED,8D,93,DC,C4,83,0,1,2
5,E4,DD,C4,20,E0,D,C7,27,97,BD,3
9,8C,A6,C9,3B,2,97,C6,9E,C4,86,8
0,A7,89,A,0,20,84,D,C7,27,80
90 DATA BD,39,8C,A6,C9,3B,6,97,C
6,9E,C4,6F,89,A,0,20,E9,8D,E,AD,
9F,A0,0,27,F8,7E,39,E5,8D,3,7E,3
9,E5,17,FF,49,8E,A,0,86,80,A5,80
,27,16,6C,84,6C,1E,6C,88,3E,6C,8
8,3F,6C,88,40,6C,88,BE,6C,88,BF,
6C,88,C0,8C,1A,0,26,E1,8E,FF,FF
100 DATA 30,1,8C,10,0,24,42,A6,8
9,A,0,27,F3,81,3,27,1F,81,82,27,
2C,81,83,27,28,6F,89,A,0,1F,10,B
D,39,8E,A6,C9,3A,FA,A4,A4,AA,C9,
3B,6,A7,A4,20,D0,1F,10,BD,39,8E,
A6,C9,3A,FA,A4,A4,AA,C9,3B,2,A7,
A4,86,80,A7,89,A,0,20,B7,39,3F
110 DATA CF,F3,FC,40,10,4,1,C0,3
0,C,3,0,0,0,0,2A,2A,2A,2A,2A,2A,
2A,2A,2A,2A,2A,2A,2A,2A,2A,2A,2A
,2A,2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,60,60,60,60,60,
60,47,41,4D,45,60,4F,46,60,4C,49
,46,45,60,56,71,6E,71;60,60,60,6
0
120 DATA 60,60,60,2A,2A,60,60,60
,60,60,53,50,45,43,54,52,41,4C,6
```

```
0,41,53,53,4F,43,49,41,54,45,53,
60,60,60,60,60,60,2A,2A,60,60,60
,60,60,60,60,60,60,60,60,68,43,6
9,60,71,79,78,71,60,60,60,60,60,
60,60,60,60,60,60,2A,2A,60,60,60
,60,60,41
130 DATA 4C,4C,60,52,49,47,48,54
,53,60,52,45,53,45,52,56,45,44,6
0,60,60,60,60,60,2A,2A,60,60,60,
60,60,60,71,74,71,60,48,41,52,56
,41,52,44,60,41,56,45,60,60,60,6
0,60,60,60,60,60,2A,2A,60,60,60,
60,60,54,41,43,4F,4D,41,6C,60,57
,41,60,79
140 DATA 78,74,76,76,60,60,60,60
,60,60,60,60,60,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,2A,2A,2A,2A,2A,
2A,2A,2A,2A,2A,2A,2A,2A,2A,2A,2A
,2A,2A,2A,2A,2A,2A,0,0,0,0,0,0,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39
150 DA"A 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
160 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
170 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
180 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
190 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
200 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
210 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
220 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
230 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
240 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
250 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
260 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
270 DATA 39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39,39,39,39,39,39,39,39,3
9,39,39,39,39,39,39,39,39,39,39,
39,39,39,39,39,39,39,39,39,39,39
,39,39,39
280 DATA 39,39,39,39,39,39,39,39
,39,39,4F,B7,FF,40,86,34,B7,FF,3
,8E,4,0,86,80,A7,80,8C,6,0,2D,F9
,8E,59,39,10,8E,39,39,A6,80,A7,A
0,8C,5F,FF,2D,F7,F,71,7E,39,39
```

CELLBA

The Listing:

```
1 CLS0:DIMA(258),B(258):POKE6549
5,0
2 A(4)=RND(3):A(12)=RND(3):A(20)
=RND(3):A(28)=RND(3)
3 SET(7,2,A(4)):SET(23,2,A(12)):
SET(39,2,A(20)):SET(55,2,A(28))
4 FORJ=4TO30STEP2:A(0)=A(32):A(3
3)=A(1)
5 FORI=1TO32:C=A(I-1)+A(I)+A(I+1
):B(I)=C-4*INT(C/4)
6 IFB(I)<>0THENSET(2*I-1,J,B(I))
7 NEXTI:FORK=1TO32:A(K)=B(K):NEX
TK,J
8 PRINT"cellular automata:t.&r.d
elbourgo";:PRINT@480,"hobart, ta
smania, australia7005";
9 PLAY"O2L6CL12CP255CP255O1L6AGP
30L12GP255GP255O2CP255O1GO2CL6EP
10L6CL12P255CP255CP255O1L6AGP30L
12GO2L6D#L12DL6C":FORI=0TO33:A(I
)=0:NEXTI
10 CLS8:PRINT"        cellular au
tomata":PRINT@64," (1) LOW  RESO
LUTION - 2 COLORS":PRINT" (2) HI
GH RESOLUTION - 2 COLORS":PRINT"
 (3) LOW  RESOLUTION - 4 COLORS"
11 PRINT@256,"WHICH CHOICE #";:I
NPUTR:R=INT(R)
12 IFR<0ORR>3THEN11
13 ONR GOTO14,25,36
14 CLS5:PRINT"LOW RESOLUTION  -
2 COLOR MODE.":PRINT"128 CELL SI
TES, 96 GENERATIONS."
15 GOSUB49:GOSUB51
16 PRINT@224,"";:FORI=1TOS
17 INPUT"SEED POSITION (1-128)";
L:L=INT(L):L=L-128*INT(L/128)
18 A(L)=1:NEXTI
19 MO=0:AM=128:N=2:GOSUB47:FORI=
1TOAM:PSET(2*I-1,0,A(I)):NEXTI
20 FORJ=1TO96:A(0)=A(128):A(129)
=A(1)
21 FORI=1TO128:ONRU GOSUB53,54,5
5,56,57,58
22 PSET(2*I-1,2*J,B(I)):NEXTI
23 GOSUB48:NEXTJ
24 IFINKEY$=""THEN24ELSE10
```



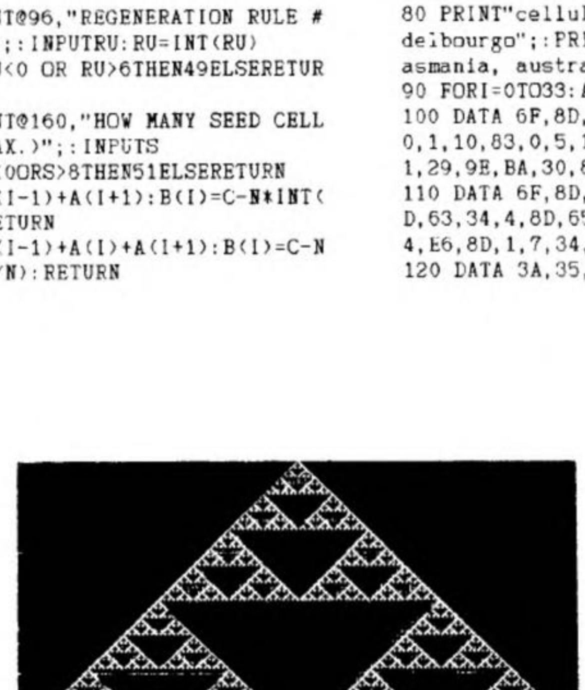
Fixed points x for
Map x -> L*x*(1-x)

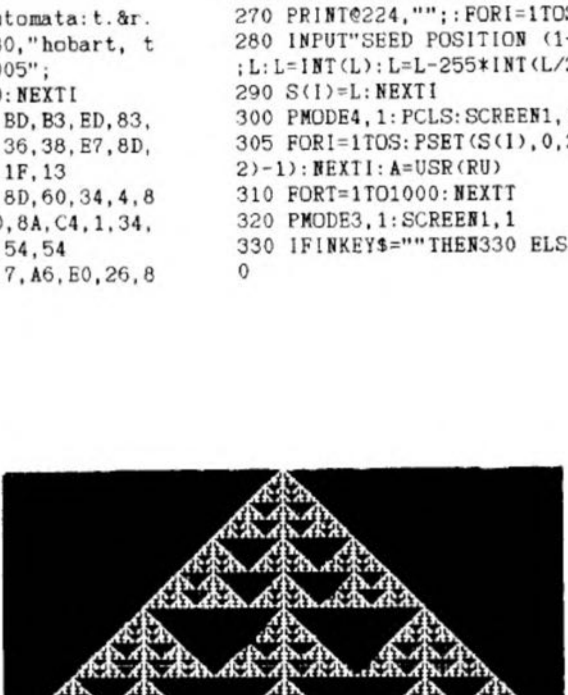
```
25 CLS0:PRINT"HIGH RESOLUTION -
2 COLOR MODE.":PRINT"256 CELL SI
TES, 192 GENERATIONS."
26 GOSUB49:GOSUB51
27 PRINT@224,"";:FORI=1TOS
28 INPUT"SEED LOCATION (1-256)";
L:L=INT(L):L=L-256*INT(L/256)
29 A(L)=1:NEXTI
30 MO=4:AM=256:N=2:GOSUB47:FORI=
1TOAM:PSET(I,0,A(I)):NEXTI
31 FORJ=1TO191:A(0)=A(256):A(257
)=A(1)
32 FORI=1TO256:ON RU GOSUB53,54,
55,56,57,58
33 PSET(I,J,B(I)):NEXTI
34 GOSUB48:NEXTJ
35 IFINKEY$=""THEN35ELSE10
36 CLS7:PRINT"LOW RESOLUTION - 4
 COLOR MODE.":PRINT"128 CELLS, 9
6 GENERATIONS."
37 GOSUB49:GOSUB51
38 PRINT@224,"";:FORI=1TOS
39 INPUT"SEED LOCATION (1-128)";
L:L=INT(L):L=L-128*INT(L/128)
40 A(L)=RND(3):NEXTI
41 MO=1:AM=128:N=4:GOSUB47:FORI=
1TOAM:PSET(2*I-1,0,A(I)+1):NEXTI
42 FORJ=1TO96:A(0)=A(128):A(129)
=A(1)
43 FORI=1TO128:ON RU GOSUB53,54,
55,56,57,58
44 PSET(2*I-1,2*J,B(I)+1):NEXII
45 GOSUB48:NEXTJ
46 IFINKEY$=""THEN46ELSE10
47 PMODEMO:PCLS(MO-4*INT(MO/4)):
SCREEN1,1:RETURN
48 FORK=1TOAM:A(K)=B(K):NEXTK:RE
TURN
49 PRINT@96,"REGENERATION RULE #
 (1-6)";:INPUTRU:RU=INT(RU)
50 IFRU<0 OR RU>6THEN49ELSERETUR
N
51 PRINT@160,"HOW MANY SEED CELL
S (8 MAX.)";:INPUTS
52 IFS<0ORS>8THEN51ELSERETURN
53 C=A(I-1)+A(I+1):B(I)=C-N*INT(
C/N):RETURN
54 C=A(I-1)+A(I)+A(I+1):B(I)=C-N
*INT(C/N):RETURN
55 C=A(I-1)+A(I)+A(I+1)+A(I-1)*(
1+A(I))*A(I+1):B(I)=C-N*INT(C/N)
:RETURN
56 C=A(I-1)+A(I)+A(I+1)+A(I-1)*A
(I)*A(I+1):B(I)=C-N*INT(C/N):RET
URN
57 C=(A(I)+1)*(A(I-1)+A(I+1))+A(
I)+A(I-1)*\(I+1):B(I)=C-N*INT(C/
N):RETURN
58 C=(A(I)+1      (I-1)+A(I+1))+A(
I-1)*A(I)*A(I+.):B(I)=C-N*INT(C/
N):RETURN
59 END
60 '*********** CELLBA **********
   ********* BOBO DELBOURGO ****
61 SAVE"CELLBA:3":END
```

CELL ML

The Listing:

```
0 GOTO5
1 '************ CELL ML *********
   ******* T&R DELBOURGO *******
5 CLEAR200,14840
10 CLS0:DIMA(33),B(33)
20 A(4)=RND(3):A(12)=RND(3):A(20
)=RND(3):A(28)=RND(3)
30 SET(7,2,A(4)+5):SET(23,2,A(12
)+5):SET(39,2,A(20)+5):SET(55,2,
A(28)+5)
40 FORJ=4TO30STEP2:A(0)=A(32):A(
33)=A(1)
50 FORI=1TO32:C=A(I-1)+A(I)+A(I+
1):B(I)=C-4*INT(C/4)
60 IFB(I)<>0THENSET(2*I-1,J,B(I)
+5)
70 NEXTI:FORK=1TO32:A(K)=B(K):NE
XTK,J
80 PRINT"cellular automata:t.&r.
delbourgo";:PRINT@480,"hobart, t
asmania, australia7005";
90 FORI=0TO33:A(I)=0:NEXTI
100 DATA 6F,8D,1,39,BD,B3,ED,83,
0,1,10,83,0,5,10,22,36,38,E7,8D,
1,29,9E,BA,30,88,20,1F,13
110 DATA 6F,8D,1,1D,8D,60,34,4,8
D,63,34,4,8D,65,16,0,8A,C4,1,34,
4,E6,8D,1,7,34,4,54,54,54
120 DATA 3A,35,4,C4,7,A6,E0,26,8
,86,7F,6A,8D,0,F4,20,2,86,80,5D,
27,10,6D,8D,0,E9,27,4,1A,1
130 DATA 20,2,1C,FE,46,5A,26,ED,
6D,8D,0,D9,26,4,AA,84,20,2,A4,84
,A7,84,1F,31,6C,8D,0,C8,27,2
140 DATA 20,A4,33,C8,20,1F,31,11
,93,B7,25,9A,E6,8D,0,B6,5A,20,B,
E6,8D,0,AF,20,5,E6,8D,0,A9,5C
150 DATA 30,88,E0,34,4,54,54,54,
3A,35,4,C4,7,A6,84,5D,27,4,48,5A
,26,F9,84,80,4D,26,3,5F,20,2,C6
160 DATA 1,1F,31,39,34,4,A6,8D,0
,81,27,74,81,1,27,67,81,2,27,4B,
81,3,27,38,81,4,27,1B,E6,E4,EB
170 DATA 62,34,4,E6,62,5C,A6,E0,
3D,34,4,EC,61,3D,A6,63,3D,EB,E0,
32,63,16,FF,43,A6,E4,E6,62,3D
180 DATA 34,4,E6,61,EB,63,A6,62,
3D,EB,E0,EB,E0,EB,E0,EB,E0,16,FF
,2A,EC,E4,3D,A6,62,3D,EB,E0,EB,E
0,EB,E0,16,FF,1B,EC,E4,3D,A6,62
190 DATA 3D,34,4,A6,61,E6,63,3D,
EB,E0,EB,E0,EB,E0,EB,E0,16,FF,3,
35,4,EB,E0,EB,E0,16,FE,FA,35,4,E
B,61,32,62,16,FE,F1,0,0,0
200 FORI=0TO319:READD$:POKE14848
+I,VAL("&H"+D$):NEXTI:DEFUSR=148
48
210 DIMS(8):CLS8:PRINT" cellular
 automata - HIGH RES."
220 PRINT@64,"256 CELL SITES, 19
2 GENERATIONS.";
230 PRINT@96,"REGENERATION RULE
# (1-6)";:INPUT RU:RU=INT(RU)
240 IFRU<0 OR RU>6 THEN230
250 PRINT@160,"# SEED CELLS (8 M
AX.)";:INPUTS
260 S=INT(S):IFS<0ORS>8THEN250
270 PRINT@224,"";:FORI=1TOS
280 INPUT"SEED POSITION (1-255)"
;L:L=INT(L):L=L-255*INT(L/255)
290 S(I)=L:NEXTI
300 PMODE4,1:PCLS:SCREEN1,1
305 FORI=1TOS:PSET(S(I),0,2*RND(
2)-1):NEXTI:A=USR(RU)
310 FORT=1TO1000:NEXTT
320 PMODE3,1:SCREEN1,1
330 IFINKEY$=""THEN330 ELSERUN21
0
```


Rule 1. Cellular Automata


Rule 2. Cellular Automata

HARDWARE PROJECT

The CoCo ROS, Part One

Add excitement to computing with your own Robot Operating System

By Dennis H. Weide

Last year at the New Mexico state fair, I had the opportunity to see a computer exhibit that had an IBM-compatible computer connected to a robot arm. The arm was controlled by BASIC's LPRINT command (PRINT#-2 in the CoCo) which could be executed from the keyboard or from a BASIC program. These commands were sent to an interface unit as big as the computer itself where they were processed into signals that the arm could use. After a few minutes of experimentation, I was able to program the arm to pick up small objects and move them. I inquired about the price of the robot arm and interface, and was told that it cost more than $5,000.

It was a little out of my price range, so I decided to build something that resembles a robot of sorts and interface it to the CoCo. I had no idea where to begin so I just started reading everything I could find on the CoCo.

After a lot of research and experimentation, I came up with what I call the CoCo ROS (Color Computer Robot Operating System). While it is somewhat primitive, it gives those people interested in experimenting with their CoCo and robotics a chance to do so without spending a fortune.

What's a ROS?

The CoCo ROS is a hardware circuit you build and a machine language program I've written to program it. It plugs into the cartridge port, but can be used with the multipack interface.

The circuit board plugs into the CoCo ROM port and allows you to connect other equipment to the CoCo. Many such circuits have already been designed and built, but I think you'll find this one is easier to program and it allows more I/O ports than most of the others. If you're new to computers or don't have much experience in circuit building, you may want to wait until you've read and studied all three parts of this article before you begin.

The ROS software consists of one BASIC program for cassette functions and one machine language program for writing, testing, modifying and executing the ROS program you write. The machine language program was written using *Deft* PASCAL and the compiled version will be available on RAINBOW ON TAPE for those who don't have *Deft* PASCAL.

ROS Defined

By now, you're probably asking what the ROS can do. Well, the ROS allows you to interface peripheral (external) equipment to the CoCo and program the computer to control external functions. There are many robot construction kits and radio-controlled toys currently on the market that can be controlled from your computer. Model railroad buffs will find the ROS a suitable computer-controlled system for use with their railroad layouts. The ROS is not designed for use with radio-controlled equipment using proportional controls.

Figure 1 is a partial list of some of the kits and toys that can be interfaced to the CoCo using ROS. I chose the Robotix R-2000 construction kit from the Milton Bradley Company over the others because it had gripper arms and four-geared, reversible motors. In about 30 minutes, I built a robot arm for testing the system. The kit is actually a toy and accuracy of movement leaves a lot to be desired, but I was able to overcome the limitations of the kit by adding microswitches to the robot arm and connecting them to the inputs on the circuit board. I'd be interested to hear about your successes (and failures) with ROS.

Let's Get Started

Next month, I'll introduce you to the interface circuit and give you tips on how to build it. And in the last installment I'll show you the software and explain how to program the robot you build. For now, let's look at how to build a simple logic tester that'll be useful for testing the ROS circuit when you build it.

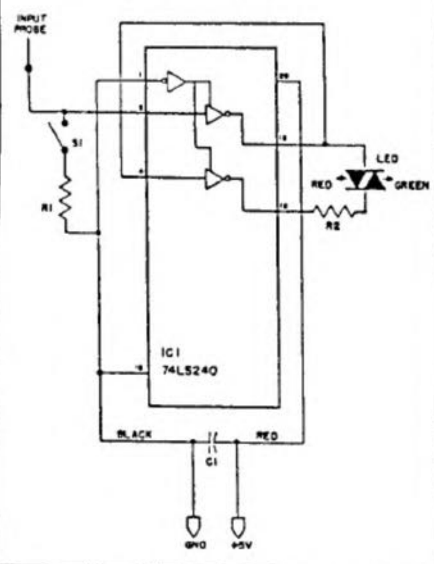
While the ROS circuit is not difficult or complicated to build, you'll need to test it thoroughly after you build it to ensure there are no wiring errors.

Figure 2 is a schematic diagram of a simple logic tester that's both inexpensive and easy to construct. It uses a 74LS240 Octal Buffer/Line Driver IC which is fully TTL and CMOS compatible. If you're experienced with digital circuit design, you'll notice that many

Figure 1: Robotic Toy List

Name	Manufacturer	Price
Capsela 2000	Play Jour	$89.95
Robotix R-1500	Milton Bradley	$49.95
Robotix R-2000	Milton Bradley	$59.95
Digger Dan's Site Crane	Revell	$29.99
Digger Dan's Colossal Crane	Revell	$42.99
Erector Set	Ideal	$31.97
Robostrux	Tomy	$19.99
Chatbot	Tomy	$59.99
Verbot	Tomy	$59.99
Omnibot	Tomy	$279.99
Omnibot 2000	Tomy	$499.00

Figure 2: Logic Tester



other ICs could have been used in place of the one chosen. Feel free to try other types for your logic tester. I chose the 74LS240 because it's used extensively in the ROS circuit and can handle 40 milliamps of current, enough to drive the LEDs. The LEDs are the bi-color type that change color according to the direction of current flow through them.

Because the ROS circuit is digital, we're only concerned with high and low states when testing it. When the logic tester is connected to the circuit under test, the LED lights, indicating whether the point under test is high or low. Because of the simplicity of the tester, switch S1 is used to condition the input probe before reading. When the tester is connected to a +5-volt power source and switch S1 is open, the green LED should light. Closing S1 should change the LED to red. If your tester colors are reversed, reverse the polarity of the LED to correct the problem. If you can't find the right type of LED, use two LEDs connected in parallel but be sure only one conducts at a time. Resistor R2 is used to limit current through the LED.

To use the tester, connect the red lead to +5 volts and the black lead to ground of the circuit under test. Close S1 so that the red LED is on. Resistor R1 holds the input at ground potential until an incoming high overrides it. If the point under test is high, the green LED lights. If the red LED stays on, remove the probe from the circuit under test and open S1. This causes a high on the input to the test circuit and the green LED lights. If the point under test is low, the red LED lights. You'll have to switch S1 back and forth to verify the test results. If the point under test doesn't change the color of the LED after switching S1, then it's open.

I recommend you use a socket for the IC so you can easily replace it if the need ever arises. When you build the logic tester, decide in advance what type of case you're going to use to house it. I used a round plastic tube for the housing and a blunted cross stitch needle for the input probe. Be sure to make a good electrical connection on the input probe to prevent erroneous indications. You can use one of Radio Shack's micro-test clips (#270-370) instead of the probe, or you can make a one-piece, hand-held tester. Wiring isn't critical but be sure to use a .1 mfd capacitor across the +5-volt supply lead to prevent switching errors.

The parts I used for the logic tester can be purchased from almost any electronic supply store, but may be expensive. I purchased all my parts from a mail-order parts house (Jameco Electronics, 1355 Shoreway Road, Belmont, CA 94002) and saved a substantial amount of money. The only problem is that the minimum order is $20. Figure 3 is the parts list showing prices at Jameco. If you're going to build the ROS circuit, wait until next month to see the parts list. You'll be able to place one order for all parts.

Figure 3: Logic Tester Parts List

Designation	Part	Quantity	Price
IC 1	74LS240	1	.69
C1 .1mfd	* 272-1432	1	.49
R1 470 ohm	* 271-1317	1	5/.39
R2 10 ohm	* 271-1301	1	5/.39
LED	XC5491	1	4/1.00
Clip lead	* 270-370	2	2/1.39
	Input Probe	1	N/A
	Case	1	N/A
S1 SPST	* 275-406	1	2/.69
Misc. 30 guage & hookup wire			N/A

Note: * Denotes Radio Shack parts

Trip Tallying

By Malvin Thomas

Mile Log prints a log sheet for keeping track of mileage and gallons of gasoline used on a trip, whether for business or pleasure.

Sample Printout

```
DATE :- - - - - - - - - - - - - - -
MILES END OF DAY: - - - - - - - - -
MILES START OF DAY: - - - - - - - -
  TOTAL MILES :- - - - - - - - - -

GAS,NO. OF GALLONS: - - - - - - - -
MILES PER GALLON :  - - - - - - - -
```

The listing: MILE LOG

```
Ø ' MILEAGE LOG
1 CLS3
2 INPUT "NO. OF PAGES";C
3 FOR P=1 TO C
1Ø CLS4:PRINT@168,"W O R K I N G
";
2Ø PRINT#-2,"":PRINT#-2,""
4Ø FOR X=1 TO 7
5Ø K=2:L=K+L
6Ø PRINT#-2,""
7Ø PRINT#-2,"DATE :- - - - - - -
 - - - - - - - - -  : DATE :- - -
- - - - - - - - - - - - - -"
8Ø PRINT#-2,"MILES END OF DAY: -
 - - - - - - - - -  : MILES END OF
 DAY: - - - - - - - - - - - "
9Ø PRINT#-2,"MILES START OF DAY:
 - - - - - - - - -  : MILES START
OF DAY: - - - - - - - - - - "
1ØØ PRINT#-2,"  TOTAL MILES :- -
 - - - -  - - - - -  :  TOTAL MILE
S- - - - - - - - - - - - - -"
11Ø PRINT#-2,""
12Ø PRINT#-2,"GAS,NO. OF GALLONS
: - - - - - - - - -  : GAS, NO. OF
 GALLONS: - - - - - - - - -"
13Ø PRINT#-2,"MILES PER GALLON :
 - - - - - - - - -  : MILES PER G
ALLON : - - - - - - - - - - -"
14Ø PRINT#-2,". . . . . . . . . .
. . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . ."
15Ø PRINT@264,"NO. "L" DONE";
16Ø NEXT X
17Ø PRINT#-2,""
175 NEXT P
18Ø PRINT@324,"* * DONE WORKING
* * ";
19Ø GOTO 19Ø
```

Advanced BASIC

by Mike Turk

(During CoCoConf, we were fortunate to have a number of experts who provided tutorials on various subjects. Our friend Mike Turk has made his Tutorial notes available for distribution here. G.)

A. Introduction

In this article I will be looking at some Advanced BASIC techniques. When Alex (Hartmann) and I were asked to give the sessions on Advanced BASIC at CoCo Conf 86 we decided to tackle different topics. I dealt with code optimisation and maintainability and Alex dealt with the technical tricks of the trade - specific techniques, PEEKS, POKES and advanced language features.

B. Maintainability v Optimisation

There is a distinct trade-off between maintainability and optimisation. Usually whatever you do to make a program more readable and maintainable causes it take up more space and to run more slowly. Similarly, whatever you do to make a BASIC program run faster or take up less space makes it less readable and harder to maintain. You must decide what your purpose is.

A program is maintainable if it is easy to add extra functions to or to fix when things go wrong. There are two distinct types of optimisation:

1. Optimisation to gain increased speed,

and 2. Optimisation to reduce space. You tend to want things to go faster and to take up less space.

Oddly (for reasons which will become plain soon), techniques that reduce program space tend also to speed things up.

I strongly suggest that you make it your aim to write well documented, easy-to-follow and easy-to-fix programs. The only time you should consider optimisation is when a particular part of the program runs unbearably slowly or when you just do not have enough space to fit the program into memory.

C. Code Optimisation

In this section I will look at optimisation in detail. But before I do this I need to explain about how the BASIC interpreter works so that you can see why these techniques are valid.

1. Inside the Interpreter

BASIC code is tokenised whenever you enter each line. This means that space is saved and the interpreter can look up, via a table, which machine code subroutine is to be used to carry out your instructions. If you PEEK at your tokenised code the tokens will be displayed as graphic symbols on your screen. Some tokens are two bytes long and others are one byte long.

The Interpreter picks up the start address of your BASIC code from locations 25 and 26. Your code starts at address = (256*PEEK(25) + PEEK(26)). Similarly there are pointers to the start of variables and to the start of arrays at 27, 28 and 29, 30..

Each BASIC line has a four byte header consisting of a two byte pointer to the start of the next line and a two byte line number. Then follows your tokenised code. Finally there is a byte containing ASCII 0, the end of line marker. The interpreter knows when it has reached the end of your code because it strikes two bytes containing 0 for the address of the next line and then one byte containing 0 for the end of line marker. The start of variables pointer points directly after this.

Try this little program:

(Program 1 - Save it, you will need it again later).

```
10 A = 10 : B$ = "TEST"
20 X = PEEK(25) * 256 + PEEK(26)
30 FOR Y = X TO 32000
40 PRINT Y , PEEK(Y) ; CHR$(PEEK(Y))
50 FOR Z = 0 TO 500 : NEXT
60 NEXT
```

Hit pause (Shift + @) when you get about a page full. The first column shows you where you are looking at in memory, the second the contents of that address as an ASCII code and the third the printable character that it represents. You should be able to see the pointer to the next line (what is 256 * the first byte + the second), the line number (10) and then "A" (ASCII 65), a space (ASCII 32) and then a graphics character (ASCII 179) - the token for "=".

When you are happy that this makes sense press a key and continue until the address you calculated comes into view and stop the program again. Just prior to the address that you worked out you should see one byte containing ASCII 0 - the end of line marker. Now change the program and edit line 20 to use locations 27 and 28. Now we will look at how the variables are stored. Stop when you have a screen full.

Each simple variable takes up seven bytes. The first is "A". (Can you see it there?). The first two bytes contain the first two characters of the variable name. The second byte is left empty if the varaiable has only one letter. If it is a string variable then 128 is added to the second character (see, just below the "B"). The next five bytes either contain the value of the numeric variable in the machine's floating point format (yuk), or contains the fields used to find the string in the area that you CLEARed to hold strings.

In both cases VARPTR points to the first byte of this five byte area. The string variable's field is useful - the length of the string is kept in the first of the five bytes and the address is kept in the third and fourth.

What do we do with the second and fifth bytes? Nothing. O. K. now let us use this information to see how we can optimise our code.

2. Use Variables Not Constants

When the interpreter sees something that is not a token, special symbol or variable name it has to translate the number (or point to the string). This translation from ASCII to a floating point number takes time. If however you assign a value to a numeric variable (eg. A = 10), the interpreter needs only to search for the variable and then to load it into its internal floating point register. If you must use constants, then use Hexadecimal constants.

Try this experiment: (Program 2 - Save it, you will need it again later).

```
10 TIMER = 0
20 FOR X = 0 TO 200
80 NEXT
90 PRINT TIMER
```

Write down the timer value (13). Now add line

```
        30 A=X*1.2345
```

and check the timer value (255). Add line

```
        5 B = 1.2345
```

and change line 30 to read

```
        30 A=X*B (I now get 66).
```

This is nearly four times faster. Now change line 30 to do the following:

```
        POKE16384,0 (98),
```

then to

```
        POKE &H4000,0 *(48).
```

Then set B to 16384 in line 5 and try POKE B,0 (43). Finally also set A to 0 in say line 7 and try POKE B,A (30). You can see just over a threefold increase in speed here.

3. Initialise Critical Variables First

When a variable name is detected by the interpreter, the interpreter classifies it as either a simple (scalar) variable or an array variable. It is also classified as a numeric or string variable. As I mentioned earlier, The interpreter picks up the address of the start of variables (from locations 27 and 28) and then searches for the simple variable.

If you initialise a variable early (In a DIM statement! 10 DIM A, B, C, X(5) etc), it occurs early in the list so that the interpreter does not have to search so far (nor does it have to convert the number to its floating point format if it has been set up). Make up a program based on program 2 and run it with your variable inside the loop say the 20th and then the first in a list and see the difference.

4. Use short variable names

If space is tight then use single and double letter variable names. The fact that a variable has only a one or two byte name also speeds up interpretation by the scanner as it works its way through the line.

5. Reuse Variables

If you reuse variables then there are less to search through and less space is taken. Reused early declared variables can be put to good effect in loops.

6. Remove REMs and 's

These not only take space but they slow things down. By the way, REM occupies only one byte but ' occupies two (Why? The extra :).

7. Remove Spaces

Removing spaces also speeds up interpretation. You can also get more code in a line and save space.

Using the EDIT command you can actually fit more into a line than can be displayed. This can cause problems when LISTing or LLISTing. I have seen utilities around to remove spaces and comments and to join lines. The only time a space is critical is when a space must be put between a numeric variable and a keyword. IFX>YGOTO10 gives a different result to IFX>Y GOTO10, (Why?)

8. Use PRINT @ Rather Than Spaces

If you put lots of space in your PRINT statements then it takes longer to display and takes more space. POKE is fast for some screen-oriented applications. TAB and PRINT@ are also faster. Why?

9. Remove Redundant THENs and GOTOs

Use IF ..... GOSUB <line number> and IF .....THEN <line number>. You do not need THEN in the first case and do not need GOTO in the second. These also add speed.

10. Multistatement Lines

I have already mentioned multistatement lines. Each line saved occupies four less bytes (not five - why? Hint ":"). More importantly it is one less line for the interpreter to search through when it obeys a GOTO or GOSUB instruction, (see later).

11. Use POKE

POKE has (with PEEK) the distinction of having one of the shortest machine code interpretation paths to follow. These are fast. They can be used to store small positive integers (whole numbers) or to keep your own file management system in numeric arrays and the space saving is enormous. Simply use VARPTR to point to the array start X = VARPTR(Z(0)) and use PEEK and POKE. Ie. POKE X+I,N where I is the index or offset to the element of your single byte field length array and N is the value you are POKEing into it. N must of course be in the range 0-255.

You could modify this to handle numbers in the range 0-65536 if you used two bytes per integer.

. When I needed space to write useful programs on my MC-10, I used this technique. In one case I kept two sets of 400 numbers using an array initialised with DIM A(159) (The extra five bytes came from the zeroth element), and in another program I kept 10000 bytes of text in an array set up using DIM A(2000).

There is a trap for the unwary here. Remember that I said that when new variables occurred they were added to the end of the variables. Well, each time this occurs, the arrays are shifted up seven bytes to make space. So you should initialise all of the simple variables before setting up your VARPTR.

12. Use FOR ... NEXT

Do not explicitly initialise a loop variable, then increment or decrement it and test each loop using IF etc. Make the interpreter work for you by using FOR ... NEXT. Not only must the interpreter re-interpret each test and GOTO but it must also do the calculation and the comparison in a non-optimal fashion. FOR ... NEXT uses the stack and a faster calculation.

13. Use Small Line Numbers

A small line number takes less space and therefore is faster. Renumber from 0 incrementing by 1 after you have done most other things.

14. Place Subroutines Wisely

When the interpreter sees a GOTO or GOSUB it has to find the line referred to. If the line number is less than the line being interpreted it searches from the start of the program chaining down the address list checking line numbers. If the number is greater than the line being interpreted it searches on towards the end of the program. I have not checked whether the interpreter is smart enough to go straight back to the current line number in cases such as an INKEY$ loop.

So, put your subroutines either right at the front or close after the critical pieces of code, or in some cases repeat the code in-line. (Watch for the trade off in space and search time for other branches).

This also explains why multi-statement lines are a good idea. There are less lines to be chained down when searching for line numbers.

15. Remove Calculations From Loops

If you use a value that does not change within a loop do not re- calculate it each time. If necessary, nest the loop, splitting the calculation. Routines that scan a multi-dimensional array or look up angles for later uses in formulae are commonly re-interpreted and re- calculated time and time again. Here is an example: (Program 3)

```
10 TIMER = 0
20 FOR X = 1 TO 20
50 FOR Y = 1 TO 100
60 A = X * 10 + Y
70 NEXT
80 NEXT
90 PRINT TIMER
```

Now try make these changes:

```
5  C = 10
30 B = X * C
60 A = B + Y
```

You should almost double the speed.

16. Avoid EXP, * and /, but..

Multiplication and division are not the strong points of most microcomputers. John Redmond knows how how to avoid these using FORTH. Repeated addition and subtraction and shifting are used by assembler programmers to get around these.

If you must raise something to a power it is still often quicker to multiply several times than to use EXP. Reserve that for non-integral powers.

Use program 2 and try these combinations in line 30. A = X EXP(2), A = X EXP(5), A = X * X, A = X * X * X * X * X.

Note the fact that EXP is much slower than * (how many times must you multiply a number by itself to take as long as EXP?). However EXP seems to take the same time.

17. Watch Strings

String space is another subject altogether. By assigning a large string space you can avoid garbage compaction. (That is when the computer runs out of string space and compresses everything). My Hot CoCo Spelling checker happily waits for 30 or so seconds every few screens while the checkee sits (im-)patiently).

You can use strings held within your BASIC code; VARPTR, PEEK and POKE; the extended form of MID$, LSET and RSET to avoid compaction problems since these tend not to alter the length of strings as long as the string manipulated does not create a string longer than the one that is the subject of MID$, LSET or PSET. Look these up in your Extended Color BASIC and disk manuals.

18. Optimise I/O

Why and how would you do this (especially with tapes)? Hint: By using large blocks and doing all your blocking and unblocking within your program. After all, you can do a lot of calculating and shifting data in the time it takes to save or read data to or from your tape or disk. Optimise disc I/O using the techniques given in your disk manual.

19. Testing

You can speed up comparisons significantly by using embedded flags or switches. TRUE is -1 (in most cases any non-zero integer will do) and FALSE is 0. To switch from true to false and back use SW = ABS(SW) - 1.

What do these mean?

A = 9 : PRINT A<0 , A>0 IF A THEN PRINT A A = PEEK(65280) = 126 : PRINT A

This sets A to be TRUE (-1) if location 65280 contains 126 and FALSE (0) if it does not. B = -((A>99) + (A>200) + (A>300)) This sets B to 1 if A is in the range 100 to 200; 2 if A is in the range 201 to 300 and to 3 if A is greater than 300.

20. TIMER

To test how long things take insert TIMER=0 into your code at the start of the critical section and print its value at the end. We have already done this. You can also use a dynamic timer in each critical block of code. Here is how.

0 T1=0:T2=0:C1=0:C2=0.... - 800 TIMER = 0 - Program block 1 - 900 T1=T1+TIMER:C1=C1+1:TIMER=0 - Program block 2 - 2000 T2=T2+TIMER:C2=C2+1:TIMER=0 - - 9000 PRINT#-2,"Block 1"T1/C1,"Block 2"T2/C2....

This gives you the average time used by each block in the program, but it also interferes slightly with the timing of your program, (Heisenberg strikes again).

21. Other Tools

There are three other useful tools that you can use to look inside your code. TRACE, PRINT and the Platinum Worksaver. Use TRACE to see which lines your program is spending its time in. Use PRINT to keep track of what is happening to your critical variables.

Use The Platinum Worksaver to hack about your code - its full screen editor with the ability to cut and paste lines and to scroll, together with its built-in error trapping and its (wait for it), ability to let you edit your code without loosing your data, makes it an ideal aid. However do not forget to detatch the keyboard table (Plat attaches a table of key definitions at the end of your program), before you finally save your program.

D. Maintainability

When you develop a program - even a very small one, there is every chance that one day you will want to modify it. You may want to fix a bug or change one of its functions by modifying, deleting or adding code. Maintainability is about making this process easier.

If the program is long or even if the program is short but contains tricky code, you may find that it takes a long time to understand what you were originally doing before you could make the changes. Here are some general principles that make for easier maintenance.

1. Use Comments and Space

The first thing to do is to put some readable comments into your code. Use REM or '. Take a couple of lines and make them stand out. If possible format them so that they sit well on a 32 column screen. Make them meaningful. Do not say "add 1 to X" say "point to the next page". You should mark the start and end of each major module of code and summarise its function. You could also remark on tricky code, the entry and return parameters for subroutines etc.

At the start of each program put details about authorship, date, title, description, modification history, peculiarities etc. This magazine prefers you to put the standard line 3 SAVE routine at the start too.

Written documentation, filed with a program listing may be justified with a large program. A pretty printing routine like DOCULIST will format your printed code and make it much more readable by putting statements on separate lines and by spacing nicely.

2. Write Modular Code

If you write modular programs - using pieces of understandable code - then programs are easy to develop, test, debug and maintain.

I usually have a main loop which then either uses GOTO or GOSUB to call the specific subroutines that do the actual work. This means that in a menu driven program you can write your top level and initialisation code and use dummy routines for the lower levels. Then you can add bits as you wish a module at a time, debugging and testing as you go.

You may care to have the odd PRINT statement kept in the code or in a subroutine to track elusive bugs. Leave it in and comment it out later.

3. Use Meaningful Variables

Use meaningful variable names. What is more readable? To use A = A + B or TOTCOST = TOTCOST + BILLS

E. Other Topics

Other topics dealt with during the talk covered the things that made programs friendlier like: a) Menus, b) Anticipation, c) Help Screens, d) Prompts, e) Documentation, and f) Error Trapping.

We also touched on the following very briefly: a) Analysis, b) Functional Design, c) The Basic Constructs, d) Program design, and e) Testing and Debugging.

However, space does not permit me to tackle these in this article.

F. Summary

During the conference we looked in some detail at how the BASIC interpreter works, how we could use this knowledge to make our programs run faster and how we could save space. Remember that when you use these optimisation techniques that your code becomes harder to fix. So, make a copy of an easy-to-read maintainable version of your code before you start hacking at it.

G. Acknowledgements

There were two articles that I used as resources for the talk:

Timer Tips, by Philip McLaughlin, Hot CoCo, November 1984, p64.

Optimize Your Code, by Robert McTernan, 80 Micro, January 1983, p270.

GRAPHICS 4K

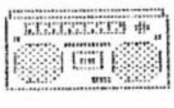
The Boogie Box

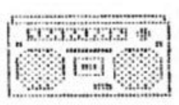
By Michael Berenz

This program prints a very small version of a jam box on any Radio Shack dot matrix.

The Listing: RADIO

```
10 REM *****************
20 REM *      RADIO      *
25 REM * BY MIKE BERENZ *
30 REM *****************
40 PRINT#-2,CHR$(18)
50 PRINT#-2,CHR$(128);CHR$(224);
CHR$(184);
60 FOR A=1 TO 38
70 PRINT#-2,CHR$(168);
80 NEXT A
90 PRINT#-2,CHR$(184);CHR$(224)
100 READ B:IF B=999 THEN PRINT#-
2,"":GOTO 130
110 PRINT#-2,CHR$(128+B);
120 GOTO 100
130 PRINT#-2,CHR$(128);CHR$(255)
;CHR$(128);CHR$(252);
140 FOR C=1 TO 5
150 PRINT#-2,CHR$(170);CHR$(213)
;
160 NEXT C
170 READ D:IF D=999 THEN 200
180 PRINT#-2,CHR$(128+D);
190 GOTO 170
200 FOR E=1 TO 5
210 PRINT#-2,CHR$(170);CHR$(213)
;
220 NEXT E
230 PRINT#-2,CHR$(170);CHR$(252)
;CHR$(128);CHR$(255)
240 READ F:IF F=999 THEN PRINT#-
2,CHR$(30):END
250 PRINT#-2,CHR$(F+128);
260 GOTO 240
270 DATA 0,127,0,48,48,0,79,77,7
3,79,73,77,75,13,9,15,73,77,75,7
7,73,79,73,77,75,77,73,79,9,13,1
1,79,64,64,70,79,79,70,0,48,48,0
,127,999
280 DATA 42,124,1,1,125,5,53,53,
53,53,5,125,1,1,124,999
290 DATA 0,127,64,67,69,74,85,90
,85,90,85,90,85,74,69,67,64,64,6
5,65,65,65,65,89,89,89,88,88,67,
69,74,85,90,85,90,85,90,85,74,69
,67,64,127,999
```





FILTER/PAGER

by Ross McKay

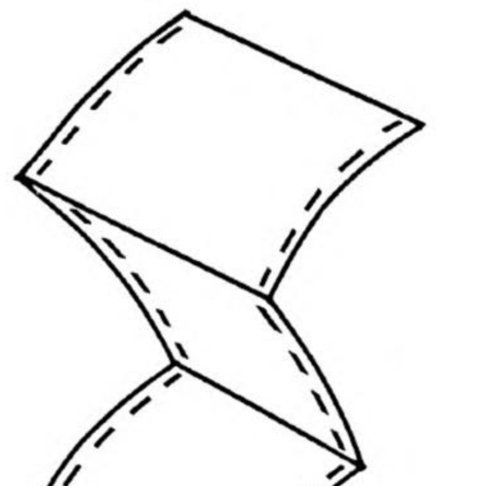
Recently I wanted to pass more than one param--eter to one of by BASIC09 programs, but I didn't want to do it all the time. Now, BASIC09 expects ALL parameters to be passed all the time.

When OS9 passes parameters to a program it gives them as a single string. The program then has to split the string up into the required parameters, eg. Dir looks for an "e" by itself, then the name. If you leave out the "e" it just doesn't list.

It is quite easy to see from here that all you have to do to have a varialbe number of params is read everything as a single string, and split it up! The best command for this is SUBSTR, which works much the same as the BASIC command INSTR.

To demonstrate this, here is a little program to list text files in pages, with margins and page numbers. The optional parameter is page heading. If you just give the program a filename, it will not print a heading.

```
PROCEDURE pager
(* Pager v1.1
(* by courtesy of
(* -- Rosko ! --

(* usage: pager("filename") >/dev
(* pager("filename heading") >/dev
(* EG: pager("document Statement for month of July") >/p

PARAM paramstring:STRING[95]

DIM pathnum,numpages,numlines,position:INTEGER
DIM heading:STRING[64]
DIM filename:STRING[30]
DIM line:STRING[255]

(* constants *)
DIM cr:STRING[1]
cr:=CHR$(13)
DIM form:STRING[1]
form:=CHR$(12)
DIM margin:STRING[12]
margin:="            "

(* BEGIN *)
position:=SUBSTR(" ",paramstring)
IF position=0 THEN
filename:=paramstring
ELSE
filename:=LEFT$(paramstring,position-1)
heading:=RIGHT$(paramstring,LEN(paramstring)-position)
ENDIF

numpages:=0
OPEN #pathnum,filename
REPEAT
numpages:=numpages+1
PRINT cr+cr
PRINT USING "s80B,STR$(numpages)
PRINT USING "s80B,heading
PRINT
numlines:=0
REPEAT
numlines:=numlines+1
READ #pathnum,line
WHILE LEN(line)>64 DO
PRINT margin; LEFT$(line,64)
numlines:=numlines+1
line:=RIGHT$(line,LEN(line)-64)
ENDWHILE
```



```
PRINT margin; line
UNTIL numlines=54 OR EOF(#pathnum)
PRINT form;
UNTIL EOF(#pathnum)
CLOSE #pathnum
END
```

Remember a little BASIC09 program from a while back, which filtered all garbage characters from a file? Well since I use it quite alot, I rewrote it in ASSEMBLER. Since I was doing it again, I decided to update it a bit.

```
Microware OS-9 Assembler RS Version 01.00.00
       06/29/86 22:51:56 Page 001

Filt - Garbage filter for text files

              * Filt  -- by courtesy of --Rosko !--
              * v1.1 29-6-86
              *
              * usage: filt [filename]
              * will filter out unwanted characters from files,
              * and convert TAB's into the appropriate # of spaces.
              *

                                   nam   Filt
                                   ttl   Garbage filter for text files

                                   ifp1
                                   use   /d0/defs/os9defs
                                   endc

        0000 87CD00AE              mod   F.END,F.NAM,PRGRM+OBJCT,REENT+1,F.ENT,F.MEM

              *******************
              * Constants section
              *
        000D 46696CF4     F.NAM    fcs   "Filt"
        0011 20202020     tabstr   fcc   "        "
        0400              bufsiz   equ   $400
        0001              stdout   equ   1

              **********************
              * Data storage section
              *
      D 0000              inpath   rmb   1          input path #
      D 0001              colnum   rmb   1          column counter for setting correct TAB's
      D 0002              buflngth rmb   2          length of buffer
      D 0004              bufend   rmb   2          end of buffer
      D 0006              buffer   rmb   bufsiz     input buffer
      D 0406              stack    rmb   $80
      D 0486              F.MEM    equ

              *****************
              * Program section
              *
        0019 0F00         F.ENT    clr   inpath     set for stdin
        001B 8601                  lda   #$1
        001D 9701                  sta   colnum     set column counter to 1
        001F A684                  lda   ,x         look for paramater
        0021 810D                  cmpa  #$0d
        0023 2702                  beq   F.ENT01    no params
        0025 8D77                  bsr   F.OPEN     yes, go open file

        0027 3046         F.ENT01  leax  buffer,u
        0029 30890400              leax  bufsiz,x
        002D 9F04                  stx   bufend     find end of buffer
        002F 8E0400                ldx   #bufsiz
        0032 9F02                  stx   buflngth

        0034 9600         F.READ   lda   inpath     read into buffer
        0036 3046                  leax  buffer,u
        0038 109E02                ldy   buflngth
```

```
003B 103F89                 OS9   I$read
003E 2533                   bcs   READERR
0040 1F20                   tfr   y,d
0042 DD02                   std   buflngth
0044 3046                   leax  buffer,u
0046 308B                   leax  d,x
0048 9F04                   stx   bufend

004A 3046       F.TEST      leax  buffer,u
004C A684       F.TST01     lda   ,x          get a char
004E 810D                   cmpa  #$0d        is it a CR?
0050 2728                   beq   CR          yes, go adjust column counter and write line
0052 8109                   cmpa  #$09        is it a TAB?
0054 2728                   beq   TAB         yes, convert it
0056 8120                   cmpa  #$20        is it a printer control?
0058 250F                   blo   F.TST05     yes, ignore
005A 8180                   cmpa  #$80        is it non-ASCII?
005C 240B                   bhs   F.TST05     yes, ignore
005E 108E0001   F.TST03     ldy   #$01        set for 1 char
0062 8601                   lda   #stdout
0064 103F8C                 OS9   I$writln    write it
0067 253F                   bcs   ERROR
0069 0C01       F.TST05     inc   colnum      add 1 to column counter
006B 3001                   leax  1,x         point at next character
006D 9C04                   cmpx  bufend      is the buffer finished?
006F 26DB                   bne   F.TST01     no, go for more
0071 20C1                   bra   F.READ

0073 C1D3       READERR     cmpb  #E$eof      is it end of file?
0075 2631                   bne   ERROR       no, do error
0077 5F                     clrb
0078 202E                   bra   ERROR

007A 0F01       CR          clr   colnum      set colnum to 0 (new line)
007C 20E0                   bra   F.TST03     write CR

007E D601       TAB         ldb   colnum      get column number
0080 C407                   andb  #%00000111 do columns MOD 8
0082 50                     negb
0083 CB08                   addb  #$08        subtract from 8 (add -B to 8)
0085 4F                     clra              make D contents of B only
0086 1F02                   tfr   d,y         put number to write in to Y
0088 3410                   pshs  x           preserve buffer pointer
008A 308DFF83               leax  tabstr,pcr point to spaces
008E 8601                   lda   #stdout
0090 103F8C                 OS9   I$writln
0093 2513                   bcs   ERROR
0095 3510                   puls  x           retrieve pointer
0097 5A                     decb
0098 DB01                   addb  colnum      update the column counter
009A D701                   stb   colnum
009C 20CB                   bra   F.TST05     continue

009E 8601       F.OPEN      lda   #read.      read access only
00A0 103F84                 OS9   I$open      open file
00A3 2503                   bcs   ERROR
00A5 9700                   sta   inpath      input path number
00A7 39                     rts               continue

00A8 103F06     ERROR       OS9   F$exit

00AB 1F1B4E                 emod
00AE            F.END       equ   *
                            end

00000 error(s)
00000 warning(s)
$00AE 00174 program bytes generated
$0486 01158 data bytes allocated
$1F33 07987 bytes used for symbols
```

If you don't feel like typing these in, and you have a modem, you will be able to download them from the OS9 section of the Sydney CoCo Connection, (02-618-3591. And while you're there, drop me a line!)

OZZIE OS9

by Fred Bisseling

FIRST A LITTLE history into how this article came about. A little over twelve months ago I purchased a copy of OS9, and my first reaction was what have I let myself in for. After reading, reading and more reading and then running the system up my next reaction was- Now what do I do? This resulted in a string of phone calls to Bob T., Kevin Holmes and others. Everytime I had a problem I'd be on the phone. Nett result- I started to get just a small glimpse of what OS9 was all about. But the phone bill was literally hundreds of dollars.

My next question was- Why doesn't somebody write an article with HINT'S & TIPS on OS9? Then I wouldn't have to spend all my time and money on the phone. Here I was, guilty of the very thing I'd accused others of doing- always expecting other people to do things for them. So to cut a long story short, here it is.

The basis of these monthly articles will be what you send me. If you have had a problem and found the solution to that problem, or have a tip on OS9 software/hardware then I want to hear from you. No matter how small or trivial you may believe it to be, please put pen to paper and send it to me c/- of

Fred Bisseling
P.O. BOX 770,
COOMA 2630

For the very item you send me might be the solution to someone elses problems. You can also MESSAGE me on the OS9 BOARD on GOLDLINK, or send a private message to my VIATEL Mailbox number 648232630. Messages and mail (self addressed envelope please) will be answered promptly.

I know that a lot of people gave OS9 away because they didn't have the help they needed to get them through. With your help, whether you are a novice or a veteran, we can help others to understand OS9. In doing so, these people will become more proficient in the use of OS9 and they in turn can contribute as well. The end result being a lot more users of OS9 and the increased availability of knowledge and software. This can only benefiet all of us. If what you send me isn't suitable for my articles, then I'll pass it onto Greg for his articles. With the release of CoCo3, OS9 is becoming even more important. Designed as an OS9 machine, the maximum benefit of CoCo3 will only be realised by the use of OS9.

I'm not going to pretend that I'm an expert on OS9; just someone who is interested in learing more and helping others to do the same. So if you find a mistake (heaven forbid) in one of my articles, don't keep it to yourself, write to me and let me know. I'll be more than happy to print the correction in my next article.

If you have written some software, or intend doing so, why not contribute it to the NATIONAL OS9 USER GROUP. (Contact Graeme Nichols or myself) But in doing so, you are probably the only person who knows how your program works. So in order to simplify things, and save a user from tearing his/her hair out, why not add a few simple notes on what your masterpiece does and how it works. This should include any special commands etc.

It seems a few people have had the following problem. Having tried to load a module into memory, it came up with an error code #205. This error: ILLEGAL MODULE HEADER- module not loaded because its sync code, header parity, or its CRC is incorrect.

This problem can arise in a number of ways- usually by changing something within the module after it has been constructed. See page 22 of the OS9 Technical Information Manual, heading 'Executable Memory Module Format'. There are other causes, and other error codes relating to the same problem, I'll cover these shortly. This section begins with a table of Relative addresses and their uses. On the right is the Check Range. So if you look at the table it shows that error #205 would occur if you changed any one or more of these values. CRC (checksum error) will occur even if you change the Attributes of the original file. So I'll use this example on how to update the CRC and make the file usable.

First use the ATTR Command to set all the Attributes (page 62 of the red book) on the file with the exception of the S and D attributes (unless of course the file is both a Directory and Shareable); ie. ATTR /D1/PROGRAM R W E PR PW PE <ENTER>. This would set the Attributes owner/public, read, write and execute permissions.

Use the DIR E /D1/program name to see that the ATTRibutes have in fact been set. We must now update the file by using the VERIFY Command. This is important, and if done incorrectly will still leave you with the CRC error. (see page 115 of red manual) This will calcuate a new set of values set in the CRC Check value. ie, VERIFY U </D1/programname >D1/newprogramname ENTER.

By using the 'U' option with the VERIFY command, it copies the old module with the new and verified CRC value to a new module. The CRC value is calculated by OS9 and added to the table of newprogramname. By using DIR E /D1/newprogramname you probably see that the ATTRibutes have been reset to that of the original file attributes set by USER0. Before KILLing/DELeting the old file ensure that newprogramname LOADs correctly without the dreaded #205 error. You can then RENAME newprogram to the old name if you wish.

The VERIFY command is a very powerfull tool and should be used often when building new software or editing others, saving yourself a lot of headaches.

Now to errors #232 and #243. Error #232 is 'INCORRECT MODULE CRC-module has bad CRC Value.' Once again this problem can be solved by regular use of the VERIFY Command.

Error #243 is an entirely different problem. The most likely causes of this error are: 1) faulty disk, or 2) bad housekeeping. Although a faulty disk is possible, it's highly unlikely. A faulty disk is more likely to show up during FORMATting.

If you are like me, when you first purchased

continued to page 54

TUTORIAL

Readable Equivalents to C

By Calvin Dodge

Calvin Dodge is a self-employed programmer/consultant. He enjoys OS-9 and C and loves his CoCo. His wife, Elsi, teaches emotionally and behaviorally disturbed children and uses the CoCo to write assignment sheets, reports, and so on. They live in Wheatridge, Colorado.

When I first began programming in C, I had a problem remembering which conditional operators did what (like "!=" for not equal rather than "< >"). Fortunately, the C pre-processor statement "#define" made it easy to rename things and help my memory. I created a file in /d1/DEFS called logic.h containing the following:

```
#define TRUE 1
#define FALSE Ø

#define EQUALS ==
#define NOTEQUALTO !=

/* bitwise operations */
#define AND &
#define NOT ~
#define XOR ^
#define OR |

/* logical operations */
#define LAND &&
#define LOR ||
#define LNOT !

#define MOD %

/* end of "logic.h" */
```

Now, in every program I have a line near the beginning that says `#include<logic.h>`. When I type `if(a EQUALS b)`, the compiler knows I mean `if(a == b)`. This makes it easier to avoid typing `if(a=b)`, which means "make a equal to b, then see if a is non-zero."

I hope this technique helps make your programming easier and your programs more readable. A C program can use all the readability it can get!

(Questions about this tip may be directed to the author at 4490 North Yukon Court 2A, Wheatridge, CO 80033, 303-420-9758. Please enclose an SASE when writing.)

OZZIE OS9

continued from page 53

OS9, you used the BOOT disk to test the speed of your disk drives, and haven't done so since. (See getting started with OS9). Os9 isn't as forgiving as Basic, and the speed of the disk drives are critical (295.5 to 304.5 rpm for standard Tandy drives). Don't forget to put a disk into all your drives before testing or the system will Hang. If the disk drive is very noisy or the speed is incorrect then adjustment/cleaning are obviously required. If you are prepared to do your own adjustments then there are several articles on the subject published in a variety of magazines. However if you are one of the faint hearted, then I suggest you take the drives to Tandy or a reputable service agent. If you don't already own dust covers for your CoCo, drives, printer etc., then I strongly suggest you do so. Having purchased them, don't forget to remove them when equipment is turned ON. Heat cannot disipate and will result in costly repair bills.

Other than dust, drive speed and head alignment, there are a couple of other things to check. Keep your cable connectors and ROMpak connectors clean of oxidants. Avoid repetitive plugging and unplugging of paks and cables as much as possible. Some connectors are fragile and easily damaged. Some of you CoCo veterans are probably saying, yes we know all that. So OK, send me some of your ideas and hard earned knowledge to pass onto less experienced users. I won't complain if you send in items for more experienced users as well.

Finally, (at last you say!) this column can only succeed with your help. So see you next month.

Living on Rainbow Time

By Greg L. Zumwalt

I have been using the CoCo 3 for quite some time now. I must say, my enthusiasm for this machine hasn't waned a bit. I've seen a variety of benchmarks comparing machine X with machine Y using "mflops," "FFTs," etc. Well, I've got my own benchmark for computers that reads: The power of a computer is directly proportional to the length of time it holds my attention.

And that's the reason for *Rainbow-Time*. *RainbowTime* is a real-time analog clock that remains on the CoCo 3 screen while I work (to remind me that after 12-14 hours of CoCo 3 experimenting, its time to sleep).

In our previous window discussion, "The Color Computer 3 Does Windows, and More" (September 1986, Page 20), we created examples of windows using text only output. This time, we'll put the CoCo 3 into high gear and experiment with the graphics displays.

OS-9 Level II Graphics

The OS-9 Level II windowing system supports six screen types summarized in Figure 1.

Using the DWSET command (the "create a device window" command), the user has the choice of any of the six available screen types.

Notice that screens five, six, seven and eight are graphics screens. In our last discussion on windows, we used Screen 7, a graphics screen, as a text display by merging a graphics character set with it. This time, we're using Screen 8, again merged with a character set, but also including the use of some of the OS-9 Level II graphics commands.

BASIC09 and Chains

RainbowTime is written as four separate BASIC09 programs. The first three build the "static" portion of the clock; the last is the running portion. Each program performs a specific task, and when finished, calls the next. This process continues until the fourth program is called and the clock begins running.

In BASIC09, the process of calling programs in this manner is called chaining. The advantage of chaining is that when a program has completed its task, it can be erased, and the next program loaded in its place. In our example, once the clock face has been created, the programs that created it are no longer required. By using chaining, these programs are removed from memory, leaving more memory for other tasks.

Getting Started

We need to create a type eight device window. Remember the OS-9 build command? We will use it to build a procedure file to create the window. At the OS-9 prompt, enter the following:

```
build window1
iniz w1
merge sys/stdfonts >/w1
display 1b 20 08 00 00 28 18 07
  00 00 >/w1
shell i=/w1&
```

After the last line, press ENTER twice. When the OS-9 prompt appears, continue by typing:

```
window1
```

When the OS-9 prompt appears this time, press the CLEAR key, and Device Window type eight appears on the screen. We are now ready for BASIC09.

The four BASIC09 programs that create *RainbowTime* are entered and saved on the disk individually. To do this, we must first enter BASIC09. From the OS-9 prompt on Window 1 type:

```
basic09 #12k
```

This loads BASIC09 into the CoCo 3 and allocates 12,000 bytes of memory for us to use.

Now we will enter the *clock1* program. At the BASIC09 prompt on Window 1 type:

```
e clock1
```

This tells BASIC09 the name of the program to edit, *clock1*. BASIC09 responds with a PROCEDURE clock1 followed by the BASIC09 edit prompt. Proceed from here by carefully typing in the *clock1* program shown in Listing 1 (the programs contain many REM statements that are not necessary for

Number	Size	Color	Memory	Type
01	40 by 24	8 & 8	1600	Text
02	80 by 24	8 & 8	4000	Text
05	640 by 192	2	16000	Graphics
06	320 by 192	4	16000	Graphics
07	640 by 192	4	32000	Graphics
08	320 by 192	16	32000	Graphics

Figure 1: Screen Types

An independent programmer and computer designer, Greg Zumwalt is one of the select few writing Tandy software for the new Color Computer 3. He owns ZCT Software of Tulsa, Oklahoma.

RainbowTime to run; eliminating them will save typing time). After you enter the last line, type:

```
q
```

This puts you back in the BASIC09 command mode. Continue by typing:

```
save
kill
```

The *clock1* program is now on the disk, and BASIC09 is ready for *clock2*. Follow the same procedure for entering the *clock2, clock3* and *clock4* programs substituting *clock2, clock3* and *clock4* as the name of the program to edit.

Running *RainbowTime*

Now that the four *RainbowTime* programs have been entered and saved on the disk, we start the clock by typing in the following:

```
kill
load clock1
run
```

You see the clock being built, one section at a time, and as one program chains to the next, you see the name of the next program appear on the screen. When *clock4* begins, three hands appear (hours, minutes and seconds) with the second hand moving as each second passes.

Each of the four *RainbowTime* programs use OS-9 Level II graphics commands in drawing the various parts of the clock. As you can see in the program listings, there are a variety of methods available to send these commands (SHELL, PRINT CHR$, etc). The *RainbowTime* programs intentionally use the various methods for the purpose of illustration. Examine each method carefully, as each has its own advantages and disadvantages.

The clock face is created by the programs *clock1, clock2* and *clock3*.

Clock1 starts the drawing of the clock face by setting the screen background to blue, drawing a white circle, then filling it. It then draws seven smaller concentric circles, filling each in a different color. Finally, the BAR command is used to erase the lower half of the seven smaller circles, leaving the upper half as the rainbow. Notice that *clock1* uses SHELL "display . . . ." to send graphic commands. *Clock1* finishes by performing the chain to *clock2*.

Clock2 draws the tick marks on the clock face. After initialization, a dual FOR/NEXT loop is entered. The two loops draw four short tick marks and one long tick mark (for the hours) around the outside edge of the clock face, 12 times. Notice that *clock2* uses the command RUN GFX2 to set the foreground color and draw the tick mark lines. This method of issuing graphic commands uses the BASIC09 graphics interface module GFX2. The end result is the same; however, the GFX2 method requires the GFX2 module in memory, thus leaving less for your programs. *Clock2* finishes by performing the chain to *clock3*.

Clock3 draws Roman numerals at the 3, 6, 9 and 12 o'clock positions using LINE DRAW commands. The data statements contain the starting and ending points for each line in each Roman numeral. A FOR/NEXT loop reads the data statements and sends them to GFX2 as line drawing commands. *Clock3* finishes by performing the chain to *clock4*.

Time Keeping for *RainbowTime*

Clock4 is the run time program of *RainbowTime*. After declaring and initializing variables and setting up for screen drawing, a loop is entered that performs the time keeping function.

The WHILE statement waits for the time to change by comparing the time with the string variable GETTIME. If the current time is different from GETTIME, the current time is copied to GETTIME and a conversion process from string to integer begins.

The integer time is saved in the three-element array, TIME. The three elements in the array represent seconds, minutes and hours, respectively. The seconds and minutes are simply converted from string to integer. However, the hours are modified. The hours are first multiplied by five to place them in the range of five to 60, like the seconds and minutes (this is so the hand drawing subroutine at Line 10 can use the same equation for determining where the hands should point). Then, the value (minutes/12) is added to hours. This gives the hour hand five distinct positions between the hours tick marks.

The time array is then compared element by element with a second three-element array, LASTTIME. If a difference is found, the hand associated with the difference (seconds, minutes and/or hours) is erased from the LASTTIME position and drawn at the LASTTIME =TIME position.

The subroutine at Line 10 converts the zero to 60 value passed to it in the LASTTIME(COUNT) element to the zero to 360 degree angle, ANGLE. ANGLE, using the BASIC09 SIN and COS functions, defines the X and Y endpoints (X2 and Y2) of the clock hand using radius as the vector length. The origin of the hand (X1 and Y1) is always in the center of the clock. Note that radius is set to 120 if COUNT equals three, otherwise it is set to 160. (When COUNT equals three, the hours variable is being redrawn, so the hand is shorter.)

So there you have it — *RainbowTime*. What if you decide it would be nice to have *RainbowTime* on the screen while you work on other programs in another window, but on the same screen? Well, this requires a scaled down version of *RainbowTime*, designed for a smaller window.

What? You don't want to type in the entire program again changing each and every data value? I don't blame you. Refiguring all of those values for a smaller window would take hours!

Well, relax, OS-9 Level II and the windowing system come to the rescue. The windowing system's scaling mathematics convert the 640-by-192 screen coordinates of *RainbowTime* into whatever size window you define, automatically! Whenever you design a program that generates graphics, always base the graphic coordinates on the 640-by-192 screen size. Then, OS-9 will properly scale the coordinates for any size window your program is run on.

Experiment With *RainbowTime*

I hope you enjoy *RainbowTime*. It was designed to illustrate the use of a wide variety of the OS-9 Level II graphics commands.

Try running *RainbowTime* on a different size window, or even a different graphics screen type. Experiment with the colors and the color palette. Try adding the remaining hour numbers. Add an alarm function by comparing GETTIME with a date/time string the user can enter.

Remember, *RainbowTime* is running as *one* task on a *multi-tasking* computer system. This means that *RainbowTime* can run while you do other programming. By experimenting, you will see how easy it is to create your own powerful multi-tasking environment using Microware's OS-9 Level II, BASIC09, windows and graphics on the Tandy CoCo 3. □

Editor's Note: To modify these routines for operation on a 128K CoCo 3, follow these steps:

1) **Delete the** `merge` **and** `shell` **lines from** ***window1*** **and do not use the procedure file to create window 1 yet.**
2) **Start BASIC09 and enter the listings. On** ***clock1, clock2*** **and** ***clock3*****, however, do not enter the** `chain` **command lines which appear at the end of each of these listings.**
3) **Use** `pack` **to place each of the four clock listings in your CMDS directory.**
4) **Type** `chd/d0/cmds` **and press ENTER. Then enter the following command line:**

```
merge clock1 clock2 clock3
clock4 runb>/d0/clocks ENTER
```

Then type `chd/d0` **and press ENTER.**

5) **Enter the following command line:**

```
attr /d0/clocks r e w pr pe pw ENTER
```

6) **Type** `window1` **and press ENTER to create window 1.**
7) **Enter** `load /d0/clocks` **and then enter the following commands while still in the normal window:**

```
clock1>/w1 ENTER
clock2>/w1 ENTER
clock3>/w1 ENTER
clock4>/w1& ENTER
```

8) **Now you may use the CLEAR key to enjoy your CoCo clock. Keep in mind, because of the graphics memory requirements, you will not have any working memory left while the clock is running.**

Listing 1: `clock1`

```
ØØØØ       REM ******************************
ØØ21       REM * RainbowTime
ØØ31       REM *
ØØ35       REM * Programmed by Greg L. Zumwalt
ØØ57       REM *
ØØ5B       REM ******************************
ØØ7C       REM *
ØØ8Ø       REM * clock1
ØØ8B       REM * draw the clock face background
ØØAE       REM *
ØØB2       REM * set the drawing mode to "store"
ØØD6       REM *
ØØDA       SHELL "display 1b 2f ØØ"
ØØEE       REM *
ØØF2       REM * set the foreground color to "white"
Ø11A       REM *
Ø11E       SHELL "display 1b 32 Ø7"
Ø132       REM *
Ø136       REM * set the background color to "blue"
Ø15D       REM *
Ø161       SHELL "display 1b 33 Ø4"
Ø175       REM *
Ø179       REM * clear the screen
Ø18E       REM *
Ø192       SHELL "display Øc"
Ø1AØ       REM *
Ø1A4       REM * place the draw pointer in the center of the screen
Ø1DB       REM *
Ø1DF       SHELL "display 1b 4Ø Ø1 4Ø ØØ 5f"
Ø1FC       REM *
Ø2ØØ       REM * the following "circle" sequences
Ø225       REM * draw a circle, then fill it,
Ø246       REM * then change the foreground color
Ø26B       REM * for the next circle
Ø283       REM * circle 1
Ø29Ø       REM *
Ø294       SHELL "display 1b 5Ø ØØ bf"
Ø2AB       SHELL "display 1b 4f"
Ø2BC       SHELL "display 1b 32 Ø6"
```

```
Ø2DØ        REM *
Ø2D4        REM * circle 2
Ø2E1        REM *
Ø2E5        SHELL "display lb 5Ø ØØ 77"
Ø2FC        SHELL "display lb 4f"
Ø3ØD        SHELL "display lb 32 Ø5"
Ø321        REM *
Ø325        REM * circle 3
Ø332        REM *
Ø336        SHELL "display lb 5Ø ØØ 66"
Ø34D        SHELL "display lb 4f"
Ø35E        SHELL "display lb 32 Ø4"
Ø372        REM *
Ø376        REM * circle 4
Ø383        REM *
Ø387        SHELL "display lb 5Ø ØØ 55"
Ø39E        SHELL "display lb 4f"
Ø3AF        SHELL "display lb 32 Ø3"
Ø3C3        REM *
Ø3C7        REM * circle 5
Ø3D4        REM *
Ø3D8        SHELL "display lb 5Ø ØØ 44"
Ø3EF        SHELL "display lb 4f"
Ø4ØØ        SHELL "display lb 32 Ø2"
Ø414        REM *
Ø418        REM * circle 6
Ø425        REM *
Ø429        SHELL "display lb 5Ø ØØ 33"
Ø44Ø        SHELL "display lb 4f"
Ø451        SHELL "display lb 32 Ø1"
Ø465        REM *
Ø469        REM * circle 7
Ø476        REM *
Ø47A        SHELL "display lb 5Ø ØØ 22"
Ø491        SHELL "display lb 4f"
Ø4A2        SHELL "display lb 32 ØØ"
Ø4B6        REM *
Ø4BA        REM * circle 8
Ø4C7        REM *
Ø4CB        SHELL "display lb 5Ø ØØ 11"
Ø4E2        SHELL "display lb 4f"
Ø4F3        SHELL "display lb 32 Ø7"
Ø5Ø7        REM *
Ø5ØB        REM * this section draws a white "bar"
Ø53Ø        REM * over the bottom half of all the
Ø554        REM * circles, leaving the upper half as
Ø57B        REM * the "rainbow."
Ø58E        REM *
Ø592        SHELL "display lb 4Ø ØØ c9 ØØ 5f"
Ø5AF        SHELL "display lb 4a Ø1 b7 ØØ 9d"
Ø5CC        REM *
Ø5DØ        REM * finished with the background,
Ø5F2        REM * "chain" to the ticker
Ø6ØC        REM *
Ø61Ø        CHAIN "ex basicØ9 clock2"
```

Listing 2: clock2

```
ØØØØ        REM *
ØØØ4        REM * clock2
ØØØF        REM * draw the tick marks around the face
ØØ37        REM *
```

```
ØØ3B        REM * define the variables to use
ØØ5B        REM *
ØØ5F        DIM angle,color,x1,y1,x2,y2,x3,y3,a,b:INTEGER
ØØ8A        REM *
ØØ8E        REM * initialize the variables
ØØAB        REM *
ØØAF        x1=32Ø
ØØB7        y1=95
ØØBE        color=1
ØØC5        DEG
ØØC7        REM *
ØØCB        REM * set the drawing mode to "XOR"
ØØED        REM *
ØØF1        SHELL "display 1b 2f Ø3"
Ø1Ø5        REM *
Ø1Ø9        REM * draw the ticks
Ø11C        REM *
Ø12Ø        FOR a=Ø TO 11
Ø13Ø          FOR b=Ø TO 4
Ø14Ø            angle=a*3Ø+b*6
Ø152            x2=SIN(angle)*191
Ø161            y2=COS(angle)*95
Ø17Ø            IF b=Ø THEN
Ø17C              x3=SIN(angle)*165
Ø18B              y3=COS(angle)*82
Ø19A            ELSE
Ø19E              x3=SIN(angle)*175
Ø1AD              y3=COS(angle)*87
Ø1BC            ENDIF
Ø1BE            color=color+1
Ø1C9            IF color=8 THEN
Ø1D5              color=1
Ø1DC            ENDIF
Ø1DE            RUN gfx2("color",color)
Ø1FØ            RUN gfx2("line",x1+x2,y1-y2,x1+x3,y1-y3)
Ø21C          NEXT b
Ø227        NEXT a
Ø232        REM *
Ø236        REM * end of ticker, "chain" to numbers
Ø25C        REM *
Ø26Ø        CHAIN "ex basicØ9 clock3"
```

Listing 3: clock3

```
ØØØØ        REM *
ØØØ4        REM * clock3
ØØØF        REM * draw the hour numbers
ØØ29        REM *
ØØ2D        REM * define the variables to use
ØØ4D        REM *
ØØ51        DIM x1,y1,x2,y2,a:INTEGER
ØØ68        REM *
ØØ6C        REM * set drawing mode to "store"
ØØ8C        REM *
ØØ9Ø        SHELL "display 1b 2f ØØ"
ØØA4        REM *
ØØA8        REM * set the foreground color to "black"
ØØDØ        REM *
ØØD4        SHELL "display 1b 32 ØØ"
ØØE8        REM *
ØØEC        REM * the following data statements
Ø1ØE        REM * define the roman numerals for
Ø13Ø        REM * the 3, 6, 9 and 12 o'clock
```

```
Ø14F       REM * positions
Ø15D       REM *
Ø161       DATA 296,16,312,32
Ø173       DATA 312,16,296,32
Ø185       DATA 32Ø,16,32Ø,32
Ø197       DATA 336,16,336,32
Ø1A9       DATA 292,16,34Ø,16
Ø1BB       DATA 292,32,34Ø,32
Ø1CD       DATA 44Ø,87,48Ø,87
Ø1DF       DATA 44Ø,1Ø3,48Ø,1Ø3
Ø1F1       DATA 444,87,444,1Ø3
Ø2Ø3       DATA 46Ø,87,46Ø,1Ø3
Ø215       DATA 476,87,476,1Ø3
Ø227       DATA 3ØØ,159,332,159
Ø239       DATA 3ØØ,175,332,175
Ø24B       DATA 3Ø4,159,312,175
Ø25D       DATA 312,175,32Ø,159
Ø26F       DATA 328,159,328,175
Ø281       DATA 16Ø,87,192,87
Ø291       DATA 16Ø,1Ø3,192,1Ø3
Ø2A1       DATA 164,87,164,1Ø3
Ø2B1       DATA 172,87,188,1Ø3
Ø2C1       DATA 172,1Ø3,188,87
Ø2D1       REM *
Ø2D5       REM * start drawing
Ø2E7       REM *
Ø2EB       FOR a=1 TO 21
Ø2FB         READ x1,y1,x2,y2
Ø3ØC         RUN gfx2("line",x1,y1,x2,y2)
Ø32C       NEXT a
Ø337       REM *
Ø33B       REM * finished, "chain" to run
Ø358       REM *
Ø35C       CHAIN "ex basicØ9 clock4"
```

Listing 4: clock4

```
ØØØØ       REM *
ØØØ4       REM * clock4
ØØØF       REM * run the clock
ØØ21       REM *
ØØ25       REM * define the variables to use
ØØ45       REM *
ØØ49       DIM x1,y1,x2,y2:INTEGER
ØØ5C       DIM count,erase,radius,angle:INTEGER
ØØ6F       DIM time(3),lasttime(3):INTEGER
ØØ84       DIM gettime:STRING
ØØ8B       REM *
ØØ8F       REM * initialize the variables
ØØAC       REM *
ØØBØ       x1=32Ø
ØØB8       y1=95
ØØBF       erase=Ø
ØØC6       DEG
ØØC8       REM *
ØØCC       REM * set the drawing color to "cyan"
ØØFØ       REM * and the drawing mode to "XOR,"
Ø113       REM * producing "red" hands on the
Ø134       REM * white portion of the clock face
Ø158       REM *
Ø15C       SHELL "display 1b 32 Ø6"
Ø17Ø       SHELL "display 1b 2f Ø3"
Ø184       REM *
```

```
Ø188        REM * this is the actual time-keeping
Ø1AC        REM * loop
Ø1B5        REM *
Ø1B9        REM *
Ø1BD        REM * wait for the time to change
Ø1DD        REM *
Ø1E1        LOOP
Ø1E3          WHILE DATE$=gettime DO
Ø1EE          ENDWHILE
Ø1F2          REM *
Ø1F6          REM * get the time, convert to integer, and save in time array
Ø233          REM *
Ø237          gettime=DATE$
Ø23D          time(1)=VAL(MID$(gettime,16,2))
Ø24F          time(2)=VAL(MID$(gettime,13,2))
Ø261          time(3)=5*VAL(MID$(gettime,1Ø,2))+time(2)/12
Ø281          REM *
Ø285          REM * determine whether hours, minutes and/or
Ø2B1          REM * seconds changed, if so, redraw the hand
Ø2DD          REM *
Ø2E1          FOR count=1 TO 3
Ø2F1            IF time(count)<>lasttime(count) THEN
Ø3Ø4              IF erase<>Ø THEN
Ø31Ø                GOSUB 1Ø
Ø314              ENDIF
Ø316              lasttime(count)=time(count)
Ø325              GOSUB 1Ø
Ø329            ENDIF
Ø32B          NEXT count
Ø336          erase=1
Ø33D        ENDLOOP
Ø341        REM *
Ø345        REM * this subroutine uses an input value
Ø36D        REM * of Ø to 6Ø to draw a hand on the clock
Ø398        REM * at the Ø to 36Ø degree positions
Ø3BD        REM *
Ø3C1 1Ø     angle=36Ø/6Ø*lasttime(count)
Ø3D6        IF count=3 THEN
Ø3E2          radius=12Ø
Ø3E9        ELSE
Ø3ED          radius=16Ø
Ø3F4        ENDIF
Ø3F6        x2=x1+SIN(angle)*radius
Ø4ØB        y2=y1-COS(angle)*(radius/2)
Ø423        RUN gfx2("line",x1,y1,x2,y2)
Ø443        RETURN
```

It's AMAZING

DS DD 5 1/4" Disks...

$20 A Box

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	$25.00
	With instructions and sound	$35.00
The Probe: A temperature measuring device which attaches to the joystick port of your CoCo or T1000, or to the joystick port of your CoCo Max. Comes with programs to start you thinking, and is supported monthly in Australian CoCo magazine.		$39.95
SOFTWARE		
Magazines: Australian Rainbow Magazine — THE magazine for advanced CoCo users! Australian CoCo Magazine — THE magazine for the new user of a Tandy computer. Also suits owners of CoCos, MC 10s, Tandy 1000s, 100s, 200s & 2000s.	Australian Rainbow 1986	$4.95
Back Issues: Australian Rainbow Magazine. (Dec '81 to now.) **Please Note:** Some months out of stock.	1982 — 1985	$2.50
Australian CoCo Magazine. (Aug '84 to now.) **Please Note:** Some months out of stock.	Australian CoCo 1986	$3.75
	Sept 1984 — 1985	$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	$10.00
	Subscription, 6 months	$42.00
	12 months	$75.00
Back issues of CoCoOz are always available	Each DISK	$10.95
	Subscription on disk, 12 months	$118.26
Rainbow on Tape, or Disk: Australian. The programs you see listed in Australian Rainbow Magazine are available on tape. A boon if you don't understand the language!	Each Tape	$15.00
	Subscription, 12 months	$144.00
American. We also supply the programs found in American Rainbow on tape. Please specify either Australian or American.	**NEW** for 1986 ONLY Each DISK	$15.50
	Subscription on disk, 12 months	$172.00
MiCoOz: The programs in the MiCo section of Australian CoCo Magazine. (For MC 10 computers only) Back issues of CoCoOz and MiCoOz are always available	Each Tape	$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.

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

Best of CoCoOz #9. 32K GAMES.

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

Best of CoCoOz #10. EDUCATION2.

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

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

GOLDSOFT

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

ORDER FORM

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

Or charge my credit card monthly
TAPE/DISK ONLY

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

Additional Requirements:

....

....

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

Name: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Address: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐

☐☐☐☐☐☐☐☐☐☐☐☐☐ **P.C.** ☐☐☐

Phone No.: ☐☐☐☐☐☐☐☐☐☐☐☐☐

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

VIDEOTEX EQUIPMENT SOLUTIONS

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

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

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

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

VISIONHIRE BUSINESS SYSTEMS
A Division of Visionhire Australia Pty Ltd
36 Brookes Street, Bowen Hills

Computer Wizardry

PO Box 979,
Gosford. N.S.W. 2250

* Communicating Software & Hardware
* Agents for Computer Hut Software
* Agents for Speech Systems
* Prompt Courteous Service
* Education Software

043 24 7293

* Phone or write for Catalogue

Bankcard or Visa Card Welcome!!

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

NT:

DARWIN	BRENTON PRIOR	089.81.7766

QLD:

BRISBANE:		
BIRKDALE	COLIN NORTH	07 824 2128
BRASSALL	BOB UNSWORTH	07 201 8659
CLAYFIELD	JACK FRICKER	07 262 8869
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	07 375 3161
SANDGATE	MARK MICHELL	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	GLEN HODGES	070 54 6583
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	STEVEN YOUNGBERRY	
TOOWOOMBA	LEN GERSEKOWSKI	076 35 8264
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
WHITEROCK	GLEN HODGES	070 54 6583

SA:

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

TAS:

DEVONPORT	JEFF BEST	004 24 1850
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	MORRIS GRADY	051 66 1331
WONTHAGGI	LOIS O'MEARA	056 72 1593
YARRAWONGA	KEN SPONG	057 44 1488

WA:

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

CANADA - CoCo:

Ontario	Richard Hobson	416 293 2346

JOHN GRIGSBY	945872030
BOB KENNY	665122050
JUDY RUTLEDGE	285350000
ARTHUR SLADE	262289400
ALLAN THOMPSON	838155830
DARCY O'TOOLE	755105770

special interest groups

BUSINESS:

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

OS9 GROUPS:

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

MC-10 GROUPS:

LITHGOW	DAVID BERGER	063 52 2282
ORANGE	DAVID KEMP	063 62 2270
PORT LINCOLN	BILL BOARDMAN	086 82 2385
SYDNEY	GRAHAM POLLOCK	02 603 5028
WARRNAMBOOL	GARY FURR	055 62 7440

TANDY 1000 / MS DOS:

QLD:		
BRISBANE		
NORTH	BRIAN DOUGAN	07 30 2072
SOUTH	BARRY CAWLEY	07 390 7946
GOLD COAST	GRAHAM MORPHETT	075 51 0015
VIC:		
MELBOURNE	TONY LLOYD	03 882 4664
NSW:		
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY WEST	ROGER RUTHEN	047.39.3903
WYONG	JOHN WALLACE	043 90 0312

FORTH:

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

ROBOTICS:

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

CHRISTIAN USERS' GROUP:

COLLIE	RAYMOND L. ISAAC	097 34 1578

300 BAUD BULLETIN BOARDS

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

1200/75 BAUD TANDY INFORMATION

GOLDLINK	VIATEL *642#
VTX 4000	03 329 2936

OTHER TANDY USERS ON VIATEL

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

ALLAN BEALE	726353300
FRED BISSELING	648232630
JACK FRICKER	726288690

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
642
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 !!